Transfer learning of convolutional neural networks for texture synthesis and visual recognition in artistic images

Nicolas Gonthier

université
PARIS-SACLAY

Transfer Learning of Convolutional Neural Networks for Texture Synthesis and Visual Recognition in Artistic Images

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Traitement du signal et des images
Unité de recherche: LTCI, Télécom Paris, Institut polytechnique de Paris, France
Référent: Faculté des sciences d'Orsay

Thèse présentée et soutenue à Palaiseau, le 31 mars 2021, par

Nicolas GONTHIER

Agnès DESOLNEUX Directrice de recherche CNRS ENS Paris-Saclay, France	Présidente
Mathieu AUBRY Chercheur (HDR), ENPC, France	Rapporteur & examinateur
Javier PORTILLA Chercheur (HDR), CSIC, Espagne	Rapporteur & examinateur
Thomas HURTUT Professeur, Polytechnique Montréal, Canada	Examinateur
Sabine SÜSSTRUNK Professeure, EPFL, Suisse	Examinatrice
Yann GOUSSEAU Professeur Télécom Paris, Institut polytechnique de Paris, France	Directeur de thèse
Olivier BONFAIT Professeur des Universités Université de Bourgogne, France	Codirecteur de thèse
Saïd LADJAL Maître de Conférence Télécom Paris, Institut polytechnique de Paris, France	Co-encadrant

Thèse de doctorat

NNT: 2021UPASG024

A Alice, pour son amour et son soutien.

All models are wrong, but some are useful.

George E.P. Box

Statisticians, like artists, have the bad habit of falling in love with their models.

George E.P. Box

Acknowledgements

Je tiens tout d'abord à exprimer mes remerciements à Yann et Saïd (mes deux encadrants à Télécom Paris). Ils furent de véritables mentors, collègues et soutiens durant ces 4 années que j'ai passé à Télécom Paris. Ils surent à la fois me donner des indications de recherche, m'apporter de l'aide lorsque cela était nécessaire tout en me laissant une immense liberté. Yann, merci d'avoir cru en mon projet de thèse un peu original. Merci de m'avoir partagé ta passion pour la synthèse de texture et la modélisation des images. Saïd, merci pour ta rigueur mathématique et ton fourmillement d'idées. J'ai beaucoup appris à vos côtés, autant sur le plan humain et que scientifique. Nos discussions de recherche ont toujours été passionnantes et souvent bien trop courtes. J'espère que nous aurons à nouveau l'occasion de travailler ensemble et à échanger sur diverses problématiques scientifiques.

Je remercie aussi Olivier d'avoir contribué à ce projet de thèse interdisciplinaire, de nous avoir apporté son expertise, de m'avoir fait rencontrer ses doctorants et doctorantes, de m'avoir mis en relation avec différents acteurs du monde de l'histoire de l'art même si tout n'a pas forcément abouti. Merci à toi aussi d'avoir cru dans ce projet qui cherchait à développer de nouveaux outils pour l'histoire de l'art.

Then, I would like to warmly thank the jury for agreeing to review my work. Your comments have truly improved the quality of this thesis. Special Thanks to Mathieu Aubry and Javier Portilla for reviewing the manuscript despite its size.

Merci aussi aux membres de mon jury à mi-parcours pour leur retours et leur remarques constructives.

Je tiens à remercier Télécom Paris ainsi le département IDS, son personnel administratif pour son accueil et son aide. Je remercie ensuite mes collègues que j'ai pu rencontrer entre avril 2017 et mars 2021. Tout d'abord, merci à mon premier co-bureau Ivan. Merci pour ces discussions animées et ces bons moments passés ensemble. Merci à mes autres co-bureaux de Paris ou de Palaiseau: Arthur, Vincent, Hélène et Antoine[1]. Merci à Clément[2] pour son aide et ses super blagues. Merci à Mateus pour son soutien et pour être encore plus motivé que moi pour refaire le monde 3 fois par jour. Merci à Sylvain pour son humour. Merci à Emanuele pour ses conseils sur le café, à Giammarco pour les descriptions idylliques de la Sicile. Merci à Kamélia d'avoir accepté d'être ma binôme au conseil d'école, à Alasdair pour ses super théories. Je remercie également tous les autres membres de l'équipe IMAGES, enseignants-chercheurs, post-docs, doctorant(e)s et stagiaires pour leur accueil convivial et chaleureux. Merci ainsi à Bastien[3], Xiangli, Gang, Nicolas G., Flora, Sylvain, Matthis, Alban, Raphaël,

[1] Même si c'était plutôt un bureau rien que pour moi à Palaiseau, mais je te comprends Antoine.

[2] A bientôt pour la révolution.

[3] #SpookyGroup et pour les autres aussi.

Alessandro, Nicolas C., Robin, Pierrick, Lara, Alessio, Rodrigo, Adrien, Hadrien, Weiying, Xu, Trinh, Pierre, Elie, Alireza, Chenguang, Xiaoyi, Corentin, Gilles, Guillaume, Christophe, Julien[4], Clément LL[5]. Merci à Florence et Isabelle d'avoir toujours été à l'écoute, à Henri pour avoir été un "sparkling partner", à Pietro, Kiwon, Jean-Marc, Tamy, Jean-Marc, Michel, Stéphane, Marco, Philippe.

Merci et bon courage pour tous les futur · e · s docteur · e · s que j'ai pu croisé durant ces années de thèse. Même à ceux et celles que je n'ai que brièvement croisé · e · s. Comme on m'a répété plusieurs fois : "La meilleure thèse c'est celle qui est soutenue".

Merci à tout les participant · e · s du groupe de lamentations sur les problèmes de convergence de CNN et d'update de pilote Cuda (aka le deep learning working group).

Je tiens aussi à remercier le personnel administratif de l'Université Paris-Saclay[6] (tout particulièrement de la bibliothèque des sciences) ainsi que le comité d'attribution des bourses de thèse qui m'a accordé sa confiance en 2017.

Je tiens à remercier tous les participant · e · s au test perceptuel sur les textures, sans vous, nous n'aurions rien pu conclure. Merci pour votre temps.

Je remercie aussi toutes les personnes que j'aurai été amené à rencontrer lors de conférences, séminaires, écoles thématiques et avec qui ce fut un plaisir de discuter : Valentin, Claire, Vincent, Arthur et tant d'autres.

Merci à tous ceux que j'ai pu rencontrer sur Twitter et avec qui j'ai pu avoir des échanges scientifiques sur des sujets très variés. J'espère qu'une telle communauté continuera à exister et à prospérer, car la recherche se doit d'être collaborative, ouverte et réflexive. Merci à Joh, Benoît, Alexander, Shona, Christophe et tous les autres qui ont pu m'apporter tellement de soutien et d'encouragements lors de ces années de doutes et de remises en question.

Je tiens à remercier tout particulièrement Antoine Courtin sans qui cette thèse n'aurait tout simplement jamais vu le jour. Je le remercie pour avoir répondu à mon mail en 2015 dans lequel je lui faisais part de ma volonté de faire de la recherche sur un sujet mêlant les mathématiques, l'informatique avec l'histoire de l'art ou l'analyse d'art. Merci à lui de m'avoir encouragé dans cette voie et de m'avoir mis en contact avec Olivier, mais aussi d'autres personnes pour échanger sur ces nombreux sujets.

Merci à Anca de m'avoir transmis son goût pour la recherche, il y a si longtemps. Merci à Daniel de m'avoir fait découvrir une autre manière de faire de la recherche ainsi que mes collègues de CiNet : Akiko-san, Eiji-san, Wada-san, Yamagishi-san, Thomas, Tristan, Daniela. Je tiens également à remercier Emmanuel Rachelson, Christophe Garion et Denis Matignon pour leurs encouragements par rapport à mon projet de thèse, à la fin de mon école d'ingénieur.

Merci à mes colocs d'Avenue d'Italie pour leur aide aussi : merci à Margot[7], Pierre, Geneviève et Matthieu. Merci à tout Pizza Inc : Diane, Alexis, Lio, Clément, Manon, Bertrand, Clément, Clément et Spock. Merci à Valentin mon binôme d'avoir supporté mes jérémiades. Merci à mes autres camarades de Supaéro déjà docteurs : Sofien, Florent, Ayoub, Kévin, Raphaël, etc. Merci aux anciens de la MPSI-B : André et Agathe.

[4]Pour m'avoir entrainé avec lui dans sa lutte contre l'injustice.
[5]Pour avoir repris la relève au conseil d'école.
[6]Mon établissement de rattachement administratif.
[7]Merci d'avoir discuté tant de fois avec moi du statut des doctorants.

Je voudrais enfin remercier mes parents pour leur soutien, leur éducation et pour m'avoir encouragé dans mes différents choix de carrière, même si vous aviez un peu peur que je fasse comme Tanguy[8].

Merci aussi à Maxime[9] et Magali pour leur soutien au cours de ses années de thèse.

Et surtout merci à ma future femme Alice[10] de m'avoir soutenu toutes ses années durant, d'avoir été enthousiaste lorsque j'ai évoqué pour la première fois l'idée de travailler sur des sujets portant sur l'histoire de l'art (en 2015 à Grenoble). Merci de m'avoir supporté quand je disais que rien n'allait jamais fonctionner, de m'avoir toujours aidé à donner le meilleur de moi-même. Merci d'être là dans ma vie tout simplement car c'est cela le plus important.

Merci aussi à tous ceux que j'oublie aussi évidemment, tête en l'air que je suis.

This research was supported in part by the "IDI 2017" project funded by the IDEX Paris-Saclay, ANR-11-IDEX-0003-02.

J'ai été acceuilli durant ces 3 années et demi de thèse à Télécom Paris, Institut Polytechnique de Paris. Je tiens à remercier cette école pour son soutien technique, matériel et administratif.

[8] Surtout durant la rédaction.

[9] Tu es le prochain docteur de la famille normalement :)

[10] Inférieur à 3.

ABSTRACT

This last decade, there has been a resurgence of concern over Convolutional Neural Networks (CNNs) because these models reach high performances in numerous tasks from the fields of computer vision, machine learning or image processing. Nevertheless, theses performances have been obtained at the price of large supervised learning processes. Several works have shown that pretrained CNNs provide generic and useful representations for images manipulation or analysis and that their internal feature maps can be transferred to other tasks and domains.

In this thesis, we study the transfer of CNNs trained on natural images to related tasks. We focus on two topics: texture synthesis and visual recognition in artworks.

The first one consists in synthesizing a new image given a reference sample. The new synthesis must be perceptually similar to the reference image while being different. Most of the recent methods are based on enforcing the Gram matrices of ImageNet-trained CNN features. We develop a simple multi-resolution strategy to take into account large scale structures. This strategy can be coupled with long-range constraints either through a Fourier frequency constraint, or through the use of feature maps autocorrelation. This multi-resolution scheme allows excellent high-resolution synthesis and combining it with additional constraints largely improves the synthesis of regular textures. We compare our methods to alternative ones on various texture samples and corroborate our visual observations with quantitative and perceptual evaluations.

In the second part, we focus on transfer learning for artistic image classification, which consists in predicting if a visual category is present in the image. First, we compare several approaches for the transfer of already learned CNNs. Indeed, pretained CNNs can be used straightforwardly without any training when the classes are similar between domains. Otherwise, they can be used as off-the-shelf feature extractors or initialization points for a new training. We illustrate the advantage of the last solution for learning efficient classifiers on painting datasets. Second, we use feature visualization techniques, CNNs similarity indexes and quantitative metrics on the maximal activation images to highlight some characteristics of the fine-tuning process.

Another possibility is to transfer a CNN trained for object detection. This task consists in detecting instances of semantic objects of a certain visual category and providing a bounding box around each instance. We propose a simple multiple instance method using off-the-shelf deep features and box proposals, for weakly supervised object detection. At training time, only image-level annotations are needed. Variants of the proposed approach including multi-layer networks and polyhedral classifiers are also considered. We experimentally show the interest of our models with respect to state-of-the-art methods, on six non-photorealistic datasets, with extreme domain shifts.

Synthèse en français

Les réseaux de neurones à convolution (en anglais CNN pour *Convolutional Neural Networks*) ont connu un fort regain d'intérêt, ces dernières années. En effet, ces modèles ont permis d'obtenir des performances inédites dans de nombreux domaines de la vision par ordinateur, de l'apprentissage automatique ou du traitement d'images. Cependant, ces performances ont été obtenues au prix d'apprentissages supervisés massifs. Plusieurs travaux ont montré que les CNNs pré-entrainés, fournissent des représentations génériques et utiles pour la modification ou l'analyse d'images. Ces représentations internes aux CNNs peuvent être transférées à d'autres tâches ou domaines.

Dans cette thèse, nous avons étudié le transfert de CNNs pré-entrainés sur des images naturelles vers des tâches proches. Nous avons principalement travaillé sur deux axes de recherche : la synthèse de texture et la reconnaissance visuelle dans les images d'œuvres d'art.

Le premier axe consiste à synthétiser une nouvelle image à partir d'une image de référence. L'image synthétisée doit être perceptiblement similaire à l'image de référence tout en étant différente. La plupart des méthodes récentes sont basées sur l'utilisation des matrices Gram des cartes de caractéristiques issues de CNNs entrainés sur ImageNet. Nous avons développé une stratégie multirésolution simple pour prendre en compte les structures à grande échelle. Cette stratégie peut être couplée à des contraintes à grande distance, telle qu'une contrainte basée sur le spectre de Fourier, ou bien avec l'utilisation de l'autocorrélation des cartes de caractéristiques. Cette stratégie multirésolution permet d'obtenir d'excellentes synthèses en haute résolution et la combiner avec des contraintes supplémentaires améliore les résultats dans le cas des textures régulières. Nous avons comparé nos méthodes aux méthodes alternatives sur divers exemples de texture et avons corroboré nos observations visuelles par des évaluations quantitatives et perceptuelles.

Pour le second axe, nous nous sommes concentrés sur le transfert d'apprentissage pour la classification des images d'art. La classification consiste à prédire si une catégorie visuelle est présente dans une image à partir de ses pixels. Dans un premier temps, nous avons comparé plusieurs approches de transfert de CNNs pré-entrainés. En effet, les CNNs pré-entrainés peuvent être utilisés directement sans aucune modification lorsque les classes sont similaires entre les domaines. Autrement, ils peuvent être utilisés comme des extracteurs de caractéristiques ou alors comme initialisation pour un nouvel entrainement. Nous avons mis en avant la supériorité de cette dernière solution pour la classification d'images artistiques. De plus, nous avons employé des techniques de visualisation de caractéristiques, des indices de similarité ainsi que des

métriques quantitatives sur les images d'activation maximale pour mettre en évidence certaines caractéristiques du processus de paramétrage fin (fine-tuning en anglais).

Une autre possibilité que nous avons étudié est le transfert de CNN entrainé pour faire de la détection d'objets. Cette tâche consiste à détecter des instances sémantiques d'une certaine catégorie visuelle et à fournir une boîte englobante autour de chaque instance. Nous avons proposé une méthode simple pour répondre à la tâche de détection faiblement supervisée, c'est à dire avec uniquement des annotations au niveau de l'image lors de l'entrainement. Cette méthode est basée sur un apprentissage à instances multiples, l'utilisation de caractéristiques figées et de propositions de boîtes issues d'un CNN pré-entrainés. Des variantes de cette approche, notamment un réseau multicouche et un classificateur polyédrale, ont également été étudiées. Nous avons expérimentalement démontré l'intérêt de nos modèles par rapport aux méthodes état de l'art, sur six jeux de données non photoréalistes, avec des changements de domaines extrêmes.

PUBLICATIONS

The material reported in this thesis was the subject of the following publications:

- Gonthier N., Gousseau Y., Ladjal S. *An analysis of the transfer learning of convolutional neural networks for artistic images*; Workshop on Fine Art Pattern Extraction and Recognition, ICPR, 2020.
- Gonthier N., Gousseau Y., Ladjal S. *Transfert d'apprentissage et visualisation de réseaux de neurones pour les images artistiques*; The Measurement of Images. Computational Approaches in the History and Theory of the Arts, DHNord 2020.
- Gonthier N., Gousseau Y., Ladjal S., Bonfait O. *Weakly Supervised Object Detection in Artworks*; Workshop on Computer Vision for Art Analysis, ECCV, 2018.

The following publications are under review:

- Gonthier N., Gousseau Y., Ladjal S. *High resolution neural texture synthesis with long range constraints*; [Submission at Journal of Mathematical Imaging and Vision].
- Gonthier N., Ladjal S., Gousseau Y. *Multiple instance learning on deep features for weakly supervised object detection with extreme domain shifts*; [Submission at Computer Vision and Image Understanding].

Contents

1

Introduction

Contents

1.1 Context

The field of image analysis has experienced an extraordinary growth in recent years, thanks to the revival of relatively old tools of artificial intelligence, notably artificial neural networks [Goodfellow et al. 2016]. In particular, Convolutional Neural Networks (CNNs) have allowed unprecedented performances in terms of visual recognition, in the setting of large scale supervised learning [Krizhevsky et al. 2012; Russakovsky et al. 2015]. They have been applied in numerous domains such as medical imaging, autonomous cars, astronomy, social network monitoring or art analysis. The main goal of such a model is to predict if a visual category is present in the image according to its raw content. To train machines that are able to recognize the visual content of an image, the research community has developed models with complex and useful internal representations.

The success of modern CNN models can be explained by the availability of a large amount of labeled data and computational resources (with the use of Graphics Processing Units (GPUs)) but certainly also by many research works since Fukushima [1980] and LeCun et al. [1989]. A CNN is composed of several layers (or stages) of transformations to an input image, where each layer transforms its input representation for the next one. Most layers are composed of convolutions and non-linear activation functions. The parameters of these layers are learned from labeled data by minimizing a task-specific objective function. Thus, a trained CNN provides a mapping between the raw image pixels and visual categories (such as objects, scenes, digits or diseases).

CNNs seem to learn a hierarchy of representations of the visual input with an increasing level of abstraction [Zeiler et al. 2014; Olah et al. 2017]. Then, for most computer vision tasks, CNNs replaced methods based on Bag of Features [Csurka et al. 2004] or part-based models [Felzenszwalb et al. 2010] and handcrafted features such as SIFT [Lowe 2004] and HOG [Dalal et al. 2005].

To face the lack of available training data, the research community pay particular attention to the transfer learning of neural networks. This machine learning paradigm consists of training a machine learning model on a new task while exploiting knowledge that the model has already learned on a previously related task. Yet, the internal representations from CNNs are somehow generic and can be reused for a different task the one on which the network has been originally trained [Donahue et al. 2014]. The transfer learning of trained CNNs became the default choice for most visual recognition tasks. This idea of transferring CNNs has also been successfully applied to heterogeneous domains, ranging from medical imaging [Tajbakhsh et al. 2016], galaxy detection [Ackermann et al. 2018] to art analysis [Crowley et al. 2014] replacing other methods. The main research questions are how to effectively transfer a CNN, how the CNN is modified during the transfer and why does transfer learning work so well.

Moreover, these internal representations also contain a localization information that can be used to point out the object of interest [Oquab et al. 2015]. This opens the path to precise object localization without localization information during training. Most works that tackled this problem are also built on pretrained CNNs and focus on natural images. Nothing excludes the possibility of looking for the same kind of algorithms for art images.

In a different direction, pretrained CNNs provide more efficient image representations than traditional filtering methods (e.g. wavelets) for image synthesis and especially texture synthesis by example [Gatys et al. 2015b]. Nevertheless, this image representation is costly and maybe over-parametrized with almost 200k parameters for the method based on Gram matrices. The task of texture synthesis by example consists in synthesizing a new image that is perceptually similar to a reference example. The statistics-based methods propose to do this by enforcing a set of statistics computed on a given image representation through an optimization process. CNN based methods renew this research field which investigates the question of statistical characterization of perceptually identical images, initiated by Julesz [1962].

A particular field in which the visual recognition performances of CNNs can be exploited is the Digital Humanities[1] and especially for the sub-field of Digital Art History. Indeed, art and culture history has greatly benefited in recent years from the results of fine art digitization campaigns. These campaigns include acquiring digital scans and photographs of artworks (mainly done by museums [Gorgels 2013; The Metropolitan Museum of Art 2017] and other cultural institutions[2]), scanning archive

[1]This field can be traced back to 1946 with the Roberto Busa's project of creating a computer-generated concordance to Thomas Aquinas's writing. A reader interested in the subject of history of the Digital Humanities is referred to [Schreibman et al. 2004].

[2]All these cultural institutions in charge of collecting and maintaining cultural heritage for the public can be regroup under the term of *GLAM* for "galleries, libraries, archives, and museums".

photographs [Seguin et al. 2018] and sale or exhibition catalogues [Joyeux-Prunel et al. 2015]. Digital collections allow the preservation and remote access to cultural heritage in particular with the recent rise of the IIIF community. Such collections, even when available online, can only be fruitfully browsed through the metadata associated with images. However, browsing them using the visual content of images may provide a wider user experience [Gordea et al. 2017]. For instance, image search algorithms may rely on perceptual features (as color or texture), local features (as SIFT [Resig 2014]), geometric ones [Hurtut et al. 2011] or spatial organization [Hurtut et al. 2008].

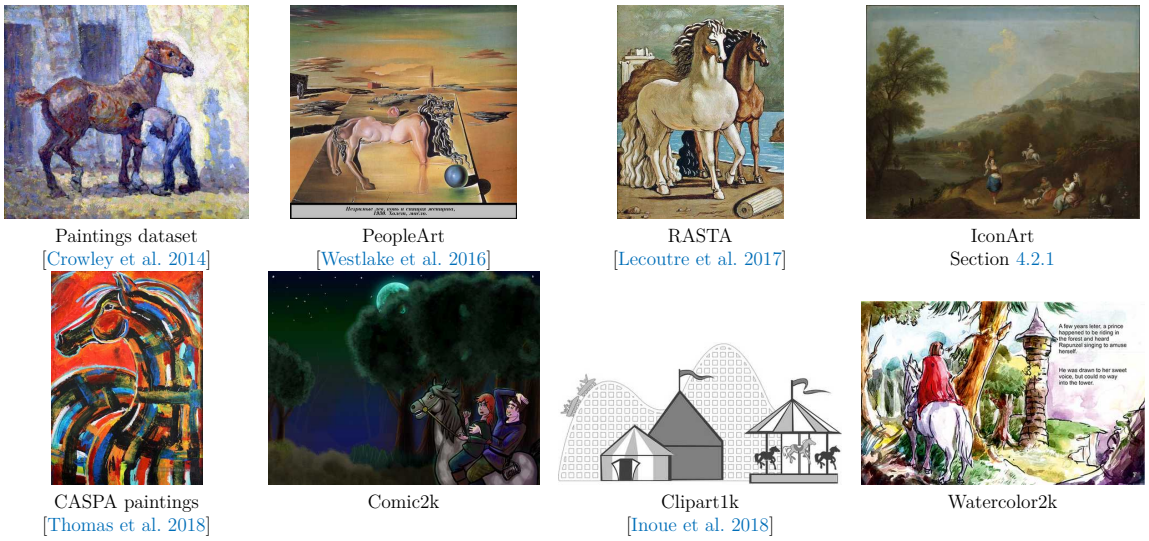
In recent years, several research teams developed search engines based on CNNs and dedicated to fine arts for different recognition tasks: Replica [Seguin 2018] for visual similarity search, ArtPI [Artrendex 2018] for style, artist and genre recognition, Oxford Painting Search [Crowley et al. 2018] for semantic recognition of arbitrary objects. Their models offer the opportunity to make large art collections searchable with complex queries about the semantic or stylistic content or even pattern similarity. They make it possible to analyze the digital images of patrimonial artifacts with data mining. Moreover, these new tools may lead to a paradigm shift across social sciences and humanities or will at least facilitate access to large amount of data, enable a wider scope and open new questions [Kitchin 2014]. Indeed, text analysis has long been privileged by the Digital Humanities, for instance, with the "distance reading" paradigm of Moretti [2000]. Moreover, the computational methodologies from Digital Humanities have been most recently integrated to the field of art history. It might now be possible to process large collections of art images in a "distant viewing" manner [Bender 2015]. Computational methods may allow to come back with a new sight to traditional questions of art history (creative process, circulation of visual patterns, iconography, etc.) and flourishing new research questions.

1.2 Motivations

In this manuscript, we are interested in the transfer of CNNs for various tasks: high resolution texture synthesis by example, classification and Weakly Supervised Object Detection (WSOD) in artwork images. These tasks require a generic image representation. At present, CNNs trained on large scale natural image datasets provide useful representations for image analysis and synthesis. However, there are many challenges to the raw application of these networks.

Lack of annotated art images Large number of annotated natural images are available in curated datasets [Deng et al. 2009; Kuznetsova et al. 2020]. These annotations even contain localization information of the object of interest [Everingham et al. 2010; Lin et al. 2014; Kuznetsova et al. 2020]. This allows for learning powerful models that recognize a wide range of visual categories and localize objects within the image. Such various, accurate and numerous annotations do not exist for artwork images, especially paintings. Most of the time, annotations are scarce and without localization information. This presents two challenges: the first is how to cope with limited annotations to learn efficient classifiers and the second is how to learn to localize objects with only image-level annotations during training.

Challenging visual content Objects depicted in paintings or other artworks can vary considerably. Such variations may have occurred for technical or stylistic reasons. Artists from Impressionism or Cubism movements depicted objects in some specific and highly abstract way compared to what can be found in photography. Such depiction diversity can be seen for a horse in Figure 1.1. These variations may also be very far from the real aspect of an object. Moreover some visual categories simply cannot be found in natural images because they emerged out of the mind of the artist or myths or religions.



Paintings dataset [Crowley et al. 2014] | PeopleArt [Westlake et al. 2016] | RASTA [Lecoutre et al. 2017] | IconArt Section 4.2.1

CASPA paintings [Thomas et al. 2018] | Comic2k | Clipart1k [Inoue et al. 2018] | Watercolor2k

Figure 1.1: Example images from different art datasets. Various interpretations of a horse. The origin of each image is mentioned below.

Understanding CNNs Although widely used and extremely effective for visual recognition, CNNs are still poorly understood. They are powerful approximation functions defined by millions of parameters. Once trained, their internal representations are pretty generic. Thus the transfer of CNN parameters from one task or domain to another became the *de facto* solution, from natural images to artworks for instance. In this context, particular attention is drawn to the transfer learning of CNNs.

Texture Synthesis Texture modeling and texture synthesis are excellent ways to study mathematical representation of images and highlight some of their limitations. The image representations used for texture synthesis go from Markov Random Fields [Cross et al. 1983] to wavelet decompositions [Portilla et al. 2000] or non-parametric Markovian modeling [Efros et al. 1999]. Since Gatys et al. [2015b], CNNs are the main image representation provider. This work proposes to use second order statistics of features from a pretrained CNN instead of a wavelet decomposition. Though these methods yield state-of-the-art results, one limitation is the difficulty to efficiently capture large scale structures for high resolution synthesis.

1.3 Contributions and Outline

In brief, we propose methods to transfer CNNs more efficiently and make a better use of their internal representations. We improve the current texture synthesis method based on Gram matrices of CNN features to tackle high resolution synthesis and long-range dependency. We study different schemes of transfer learning from natural images to art ones. Then, we use features visualization and diverse distances between CNNs for getting a better understanding on fine tuning process. Finally, we propose a simple but efficient multiple instance perceptron model, learned on top of deep features from a CNN pretrained on a detection task. We believe that such a method could be used for automatic recognition and localization of iconographic elements in large artwork collections.

Texture Synthesis Chapter 3: In this chapter, we present several texture synthesis methods based on CNNs that significantly improve the ability to preserve the large scale organization of textures. We explore the benefits of a multi-resolution framework to account for long range structures and allow the synthesis of high resolution images. In this multi-resolution framework, we show that additional constraints are useful in the case of regular textures. These constraints rely on the Fourier spectrum of the input image or on the full autocorrelation of the CNN features. The proposed methods are experimentally tested and compared to alternative approaches, both in a quantitative way and through a user study.

Analysis of Transfer Learning Chapter 4: Here, we examine the different transfer learning schemes applied to art classification tasks. We introduce a new dataset with iconographic labels from which to learn classifiers, and evaluate the different transfer learning techniques. We show that there is a clear advantage of fine-tuning a pretrained model compared to using off-the-shelf features extraction or training from scratch. Moreover, a double fine-tuning involving a medium-size artistic dataset can improve the classification on smaller datasets, even when the task changes. We also use feature visualization techniques to qualitatively demonstrate some properties of the transfer learning processes. We corroborate these properties with a quantitative evaluation based on characterizations of the set of maximal activation images and CNN comparisons.

MIL Model for WSOD in Artworks Chapter 5: This chapter presents a weakly supervised learning method for detecting objects in artworks. This method utilizes a simple multiple instance approach applied on pre-trained deep features and yields excellent performances on non-photographic datasets, possibly including new classes (e.g. angel or the character Mary that are intrinsically absent from photographic datasets). We investigate several flavors of the proposed approach, some including a multi-layer network and a polyhedral classifier.

Before presenting our contributions, we provide in Chapter 2 relevant background for the various but related themes that are considered in this thesis, including image representation with CNN, transfer learning, CNN understanding, texture synthesis

with CNN and deep learning for art analysis. In Chapter 6, we finally summarize our contributions and suggest directions for future work.

2

Related Work

Contents

The goal of this chapter is to provide a literature review of all the key topics covered by our work. In this thesis, we are interested in the use of Convolutional Neural Networks (CNNs) for several image processing tasks such as artistic image classification, weakly supervised object detection in art images and texture synthesis. We first present CNNs and the paradigm of transfer learning (Section 2.1). We then introduce standard tools for CNN analysis methods such as feature visualization and feature similarity (Section 2.2). Next, we present the main works about CNN models in the context of art analysis (Section 2.3). In Section 2.4, we give an overview of Weakly Supervised Object Detection (WSOD) and Multiple Instance Learning (MIL) strategies to automatically learn the locations of the objects with image-level labels only. It is a specific application of CNN model. Finally, we present the texture synthesis problem which consists in creating new images perceptually similar to a reference one (Section 2.5). We first give a broad overview of the field and previous works, before giving a more detailed account of CNN-based methods, that are closely related to the contributions of this manuscript.

2.1 Convolutional Neural Network and Transfer Learning

In this section, we present the CNN model. This kind of model became the state-of-the-art model for image classification this last decade and the *de facto* solution to most of the computer vision problem. Then we will define the transfer learning paradigm with a focus of the transfer of CNN for image-related tasks.

2.1.1 Learning Visual Representations With Convolutional Neural Networks

In 2012, AlexNet, a deep CNN architecture proposed by Krizhevsky et al. [2012], has emerged as an efficient method for classifying large scale image datasets, by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al. 2015]. In this section, we first describe the main building blocks of CNNs and then some of the classical CNN architectures.

2.1.1.1 Convolutional Neural Network

A *supervised learning* problem can be defined by a triplet:

- an input space $\mathcal{X}$
- an output space $\mathcal{Y}$
- a probability distribution $p(x, y)$.

The goal is to find the "best" prediction function $f_\omega : \mathcal{X} \to \mathcal{Y}$, parametrized by ω based on example input-output pairs. In other words, we are looking for the optimal function f_ω that minimizes the expectation over $p(x, y)$ of a given loss[1] ℓ between its prediction

[1]It expresses the discrepancy between the predictions of the model being trained and the actual ground truth labels.

and the correct output. This expectation is called risk:

$$E_{(x,y)\sim p(x,y)}\{\ell(y, f_\omega(x))\}. \tag{2.1}$$

This function f_ω is built from a finite number of labeled training data consisting of a set of training examples. We note $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \ldots, N\} \in (\mathcal{X} \times \mathcal{Y})^N$ the training dataset containing N samples. Thus, one can only estimate an empirical risk:

$$R_{emp}(f) = \frac{1}{N} \sum_i L\left(y_i, f_\omega\left(x_i\right)\right). \tag{2.2}$$

We will denote $f_\omega(x) = \hat{y}$ the prediction for the input x.

CNN is a machine learning solution among others to tackle this problem. A CNN is a type of feed-forward artificial neural network that uses at least one *convolutional layer*. A feedforward neural network is a non-linear function approximator that can be decomposed in a succession of mathematical operations (or functions), called *layers*. Each layer takes as input the data from the previous layer[2]. There are mainly three types of operation:

- Combination function
- Activation function
- Pooling layer

First, the combination function combines in different ways the output of the previous layer to calculate the values of the next layer. The two main combination layers are the fully connected one and the convolutional one. The fully connected layer is a linear combination (used for multiple layers perceptron type networks). The convolutional layer is a convolution product between data from the previous layer and a certain number of convolution kernels. Often the result of this convolution product is rectified by adding a bias. The convolution exploits spatially local correlation by enforcing a local connectivity pattern between adjacent pixels. Let us actually mention that mathematically a cross-correlation is computed in the mostly used deep learning frameworks such as Tensorflow or Pytorch. The convolutional layer can be seen as a regularized version of the fully connected one.

The second operation is an activation function; a non-linear, thresholding function (for instance it can be the positive part of the input[3] or a sigmoid function[4]). The activation function can be seen as a contractive non-linear pointwise operator [Mallat 2016]. It permits to learn more complex decision boundaries than a sole linear layer.

Finally, the last operation corresponds to a pooling of the outputs of the previous layer using a subsampling. In the case of CNN, the pooling layer is most of the time a spatial subsampling. The spatial pooling layer permit to reduce the dimension of the current layer, to control the computational complexity of the architecture and grants a degree of translational invariance. The most used pooling layer computes a max or an average.

Then, a feedforward network with n layers can be written as:

$$f_\omega(x) = f_{\omega_n}\left(f_{\omega_{n-1}}\left(\ldots f_{\omega_2}\left(f_{\omega_1}(x)\right)\right)\right), \tag{2.3}$$

[2]Or from several of the previous layers.

[3]Also called a Rectified Linear Unit (ReLU) and equal to $f(x) = \max(0, x)$.

[4]Equal to $f(x) = \frac{1}{1+e^{-x}}$

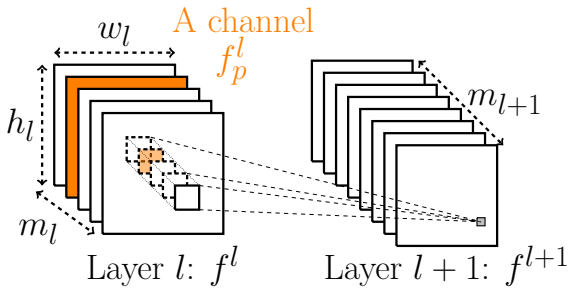


Figure 2.1: Diagram illustrating the passage from the l-th layer to the $l+1$-th layer through convolution.

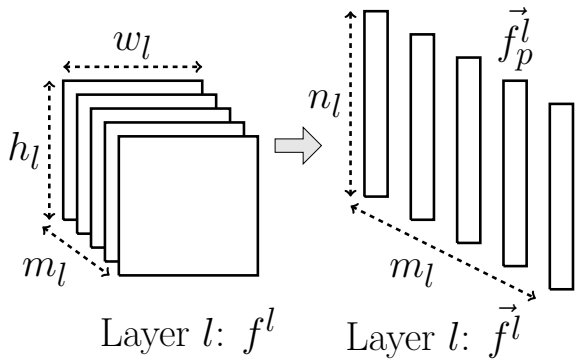


Figure 2.2: Vectorized version of the features map from layer l.

where x is the input, $\omega = [\omega_1, \ldots, \omega_n]$ is the vector of CNN parameters, f_{ω_l} is the l-th layer and ω_l is the vector of parameters of the l-th layer. We note $f^l = f_{\omega_l}(f^{l-1}) \in \mathbb{R}^{h_l \times w_l \times m_l}$ the output of the l-th layer for a given input image $I \in \mathbb{R}^{h \times w \times m_0}$. In the following work, we note h_l and w_l the spatial dimensions and m_l the number of channels of the l-th layer (or the feature maps) as in Figure 2.1. In the case of a convolutional layer, the number of channels m_l is equal to the number of convolutional kernels/filters. We use the term of *channel* denoted f_p^l to refer to the p-th 2D output according to the "filters dimension" of the layer f^l whereas the term *neuron* refers to an individual position/pixel i and is denoted: $f_p^l(i)$. For the input image, h and w are the spatial dimensions and m_0 the number of channels ($m_0 = 3$ for the RGB image). The spatially vectorized version of f^l is noted $\vec{f}^l$ and can be seen in Figure 2.2.

The term "feedforward" model is used because information flows strictly in the forward direction, from the input layers, through hidden layers, to output layers on the contrary to the Recurrent Neural Network (RNN) that include feedback connections [Rumelhart et al. 1986].

Historically, the *perceptron* of Rosenblatt [1958] may be considered as the ancestor of the neural network[5]. Fukushima [1980] proposes a neural network architecture named *Neocognitron* for pattern recognition explicitly motivated by Hubel and Wiesel's hierarchical model of the cat or monkey visual cortex [Hubel et al. 1962; Hubel et al.

[5]The reader may refer to the recent overview of deep neural network history by Schmidhuber [2015].

1968]. His model alternates shift invariant non-linear transformation and down sampling thus incorporating the core of CNN without naming it. The parameters of this model are obtained by a layer-wise unsupervised clustering algorithm. *Deep Learning* is a paradigm that consists in simply stacking many layers one after the other.

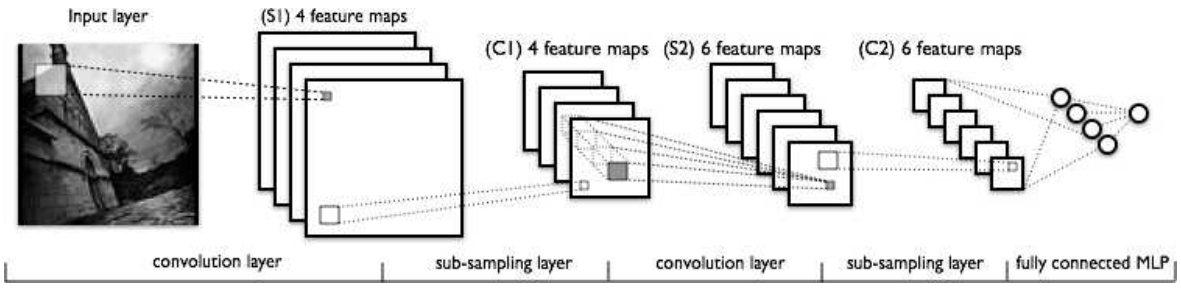


Figure 2.3: LeNet-5 architecture - Figure from [LeCun et al. 1998].

In Figure 2.3, one can see the standard architecture of a CNN. It has been designed for handwritten digit recognition in the seminal work of LeCun et al. [1998]. In this schematic, the convolution and activation layers are grouped under a single label. This network consists of two convolution layers (the first one made up of 4 different convolutions and the second one of 6) and two sub-samplings. The last layers of this network are fully connected layers. They allow to obtain an output of the network in the form of the different classes of the classification task. The model has around 60000 parameters.

On the contrary to Fukushima [1980], the CNN parameters can be learned by a minimization of the empirical risk (mentioned in equation (2.2)), most of the time combined with a regularization term. In 1989, LeCun et al. [1989] used back-propagation [Rumelhart et al. 1986] to learn the convolution kernel coefficients directly from images of hand-written numbers. Thanks to this gradient based method, learning was thus fully automatic, and can be used in a broad range of image recognition problems and image types. Stochastic Gradient Descent (SGD) and its variants (such as ADAM [Kingma et al. 2017]) are nowadays the main optimization algorithms for machine learning in general and for deep learning in particular. The gradient descent method is a common way to minimize an objective function of parameters ω by updating the parameters in the opposite direction of the gradient of the objective function with respect to the parameters. Computing the exact gradient is expensive because it requires evaluating the model on every example in the entire dataset. To ease this problem, the SGD method estimates the gradient by randomly sampling a small amount of training examples (called a minibatch). More information about deep learning and CNN can be found in the excellent book of Goodfellow et al. [2016].

We have to mention that two active research fields propose methods to learn model parameters without the pairs of training examples and associated labels. These are unsupervised learning [Fukushima 1980; Ranzato et al. 2006; Erhan et al. 2010] and self-supervised learning [Zhang et al. 2018a; Zoph et al. 2020]. These fields are out-of-the-scope of this manuscript.

2.1.1.2 Classical Convolutional Neural Network Architectures for Image Classification

We now present a brief history of classical CNN models. Since 2012, CNNs have become popular with the large win of AlexNet (in the competition ILSVRC) over handcrafted features model such as HOG [Dalal et al. 2005] or SIFT[6] [Lowe 2004]. Since then all the SOTA models for large scale visual recognition tasks are based on CNNs. New architectures, layers or building blocks have been proposed to reach better, quicker and more stable learning.

AlexNet consists of 8 layers (5 convolutional and 3 fully connected) as it can be seen in Figure 2.4a. The number of parameters is 60 millions, its training has been made within a reasonable time thanks to the use of GPU [Krizhevsky et al. 2012]. Simonyan et al. [2014] show that the depth of the network is an important component for good performance. They introduced several variants of a "very deep" convolutional network including *VGG19*, a 19 layers CNN architecture (see Figure 2.4b). This model was widely reused, thanks to the sharing of its learned weights to the community. A drawback of this model is its number of parameters (144M) and inference cost. Szegedy et al. [2015] introduce a deeper model named *InceptionV1* (originally named GoogleNet) who won the ILSVRC 2014 classification challenge. The number of parameters is reduced to 7 millions thanks to the introduction of Inception modules and the use of only one last fully-connected layer for the classification (compared to the three used by VGG-like models). This architecture of 22 layers can be seen Figure 2.4c. He et al. [2015] introduce *Residual Network* (ResNet) incorporating residual blocks to learn deeper CNN, in addition to the use of batch normalization [Ioffe et al. 2015]. Indeed, the residual blocks permit to overcome the vanishing/exploding gradient problem [Glorot et al. 2010] that appears when adding too many layers. Then models could have more than hundreds of layers (see Figure 2.4d). The ResNet model got the first places at the ILSVRC 2015 detection, localization, MS COCO 2015 [Lin et al. 2014] detection and segmentation competitions, and has had an extraordinary impact in computer science this last five years.

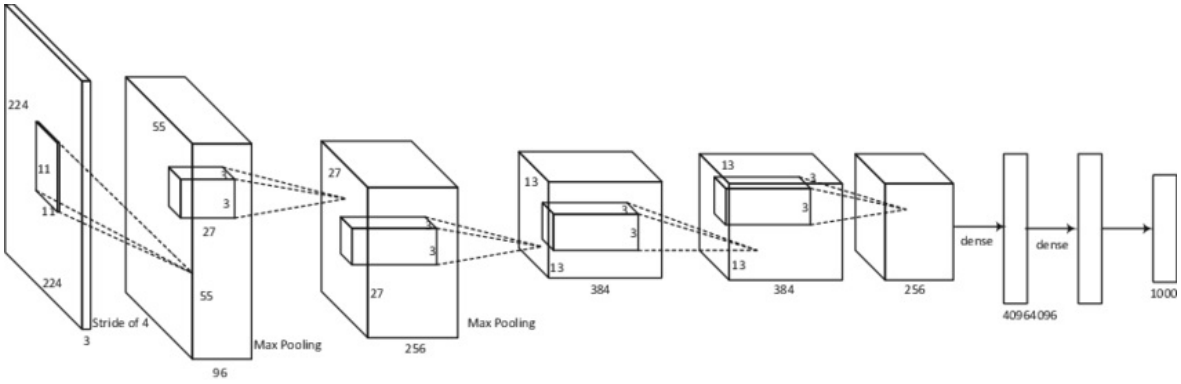
The extremely detailed version of these three architectures can be found in Annex, Chapter D.

Next, research works about CNN for large scale image classification have been conducted in several directions. While there are too many to list here, following are a few examples:

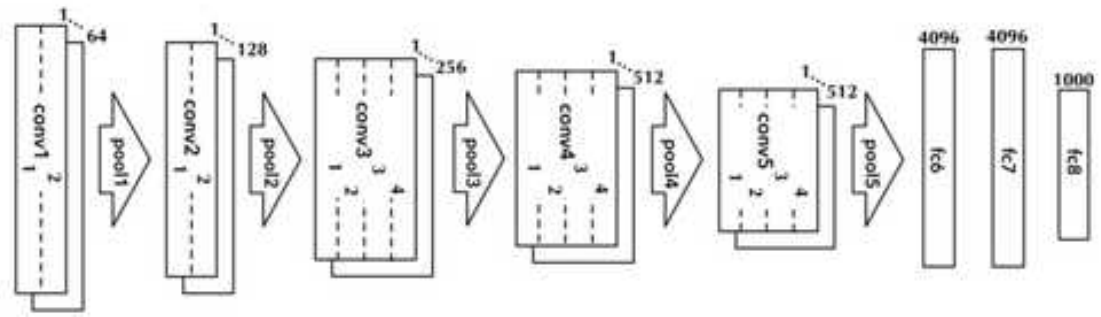
- Combining Inception modules with residual ones [Szegedy et al. 2017]
- Providing lighter CNN [Howard et al. 2017]
- Providing new efficient building blocks such as the "Squeeze-and-Excitation" block, a channel-wise recalibaration feature responses [Hu et al. 2018]
- Working with bigger input images [Tan et al. 2019]
- Fixing the train-test resolution discrepancy [Touvron et al. 2020]

Nevertheless, computer vision has moved from feature engineering to architecture design such as AlexNet, VGG19, InceptionV1 and ResNet. Meta-learning may be the next paradigm shift. This may take the form of meta-architecture design. For instance,

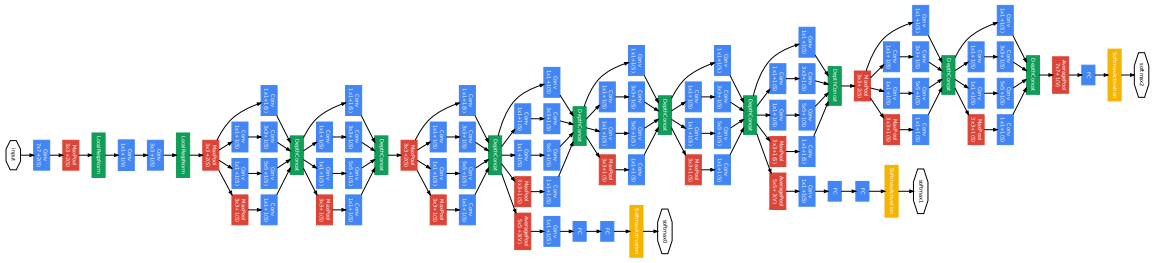
[6] Both based on local histograms of orientations.
[7] Courtesy [He et al. 2016].



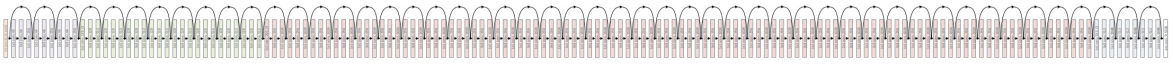
(a) AlexNet architecture, 8 layers [Krizhevsky et al. 2012]



(b) VGG19 Architecture[7], 19 layers [Simonyan et al. 2015]



(c) InceptionV1 (or GoogLeNet) Architecture, 22 layers [Szegedy et al. 2015]



(d) ResNet 152 Architecture, 152 layers [He et al. 2015]

Figure 2.4: Several different CNN architectures for image classification.

Neural Architecture Search takes a novel approach to meta-learning architectures by using a RNN trained with Reinforcement Learning to design architectures that result in the best accuracy [Zoph et al. 2017a; Zoph et al. 2017b; Tan et al. 2019]. This can also consist in learning the data augmentation strategy [Cubuk et al. 2019] or both at the same time [Wei et al. 2020]. However, these methods are costly because they require an optimization process in a space with an exponentially increasing number of dimensions.

As the SOTA changes very regularly, we invite the reader interested in these subjects to refer to the "Papers with codes" website, mostly up to date.

2.1.2 Transfer Learning

We start by giving a brief formal definition of transfer learning, then present three specific techniques for CNN transferring, and finally give examples of applications to visual recognition.

2.1.2.1 Deep Transfer Learning

Transfer Learning is a general paradigm broadly used in Machine Learning. Transfer learning is defined as the situation where two supervised learning problems are available: a source problem: $(\mathcal{X}_\mathcal{S}, \mathcal{Y}_\mathcal{S}, p_\mathcal{S}(x, y))$ and a target one $(\mathcal{X}_\mathcal{T}, \mathcal{Y}_\mathcal{T}, p_\mathcal{T}(x, y))$ which are related but not identical (i.e. at least $\mathcal{X}_\mathcal{S} \neq \mathcal{X}_\mathcal{T}$, $\mathcal{Y}_\mathcal{S} \neq \mathcal{Y}_\mathcal{T}$, or $p_\mathcal{S} \neq p_\mathcal{T}$). For instance, if we transfer a model learned on RGB images to multispectral ones, we are in the case of $\mathcal{X}_\mathcal{S} \neq \mathcal{X}_\mathcal{T}$. Transferring a model from a flower species classification task to a flower colors one, means $\mathcal{Y}_\mathcal{S} \neq \mathcal{Y}_\mathcal{T}$. Transferring a model from an offline academic setting to a production online one may cause $p_\mathcal{S} \neq p_\mathcal{T}$. Transfer learning consists in exploiting the source data $\mathcal{D}_\mathcal{S}$ together with the target one $\mathcal{D}_\mathcal{T}$ to potentially find a better model in terms of empirical loss on the target test set than the one obtained when using only the target data. Among other things, transfer learning is useful when source data is abundant while target one is not.

Depending on the difference between the source and target domain and tasks, Pan et al. [2010] define several transfer learning settings. In the *inductive transfer learning* setting, the target task is different from the source task, no matter the source and target domains are the same or not. In the *transductive transfer learning* setting, the source and target tasks are the same, while the source and target domains are different. In addition, Weiss et al. [2016] define more details settings such as the *homogeneous* and *heterogeneous* transfer learning settings. In the first case, only one kind of data is used whereas in the second one several are considered (images and texts for instance). Moreover, they draw a distinction, among other things, between *symmetric* and *asymmetric feature-based* setup. This latter consists in mapping the features of the source domain into the target feature space whereas in the first setup, the features from both domains are treated in a similar way. Readers are referred to these two surveys [Pan et al. 2010; Weiss et al. 2016] for the general framework on transfer learning. The common framework of transfer learning of artificial neural network has been introduced in [Pratt et al. 1991], whereas the first work on transfer learning of perceptron have been conducted during the 1970s, by Bozinovski and published in Croatian according to [Bozinovski 2020].

Designing and training an entire CNN from scratch (with random initialization) on a large dataset is difficult, because this task requires an accurate knowledge of training and engineering tricks, huge sets of labeled training images and computational resources (GPU). For instance, it takes about one week to train a standard CNN on ImageNet with one consumer GPU. Nevertheless, CNNs seem to learn representations on a large scale dataset that are both discriminative and generic (more information about it can be found later in Section 2.2). Thus, CNNs have been successfully recycled for other computer vision tasks. They can be efficiently transferred to other (small) datasets. As already mentioned, the models from the ILSVRC competitions [Russakovsky et al. 2015] and trained on ImageNet became the *de facto* solution for most of the computer vision problems. As we assume in the following work that the input space are identical (RGB images) but the output space and probability differ, ImageNet classification against art classification for instance. Hence, we focus on the inductive, homogeneous and asymmetric feature-based transfer learning setup in Chapters 4 and 5. There are three settings in Transfer Learning of CNNs mainly used:

1. *Off-the-shelf* feature extraction
2. *Fine-Tuning* a pretrained model
3. Training the same CNN architecture *from scratch* (with random initialization) on the new task

Off-the-shelf feature extraction This strategy is to use the fixed feature representations of a CNN pre-trained on a large dataset to train a classifier (Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), etc.) for the new dataset. It can be the feature representations from the penultimate layer but also mid-level ones. These features are often called *deep features*. Donahue et al. [2014] have shown that the AlexNet architecture of Krizhevsky et al. [2012] can be used (without fine-tuning) as a blackbox feature extractor, yielding excellent performance on several recognition tasks including scene classification, fine-grained sub-categorization, and domain adaptation. Other contemporaneous works propose transferring image representations from this architecture trained on ImageNet dataset. They investigate transfer to other visual recognition tasks such as Caltech256 image classification [Zeiler et al. 2014], object and action classification [Oquab et al. 2014] or object localization [Sermanet et al. 2013a].

Fine-Tuning Before 2014, one conventional solution to tackle the difficulty of training deep neural networks was to use unsupervised pre-training, followed by supervised fine-tuning. We can find an example for MLP in [Hinton 2007], for Deep Belief Network and autoencorder in [Erhan et al. 2010] and one for pedestrian detection with CNNs in [Sermanet et al. 2013b]. In [Erhan et al. 2010], they mentioned the fact that unsupervised pre-training can be seen as a regularizer and that it works better than random initialization.

In the seminal work about the Region Convolutional Neural Network (R-CNN) model, Girshick et al. [2014] run a supervised pre-training on the ImageNet image-level annotation classification task before fine-tuning their model on Pascal VOC dataset for the object detection task. They show that this two-step method with an auxiliary dataset is an effective paradigm for learning high-capacity CNNs when data is scarce. Since then, ImageNet-pretrained networks are used as backbone models for object

detection and semantic segmentation tasks. Using a pretrained CNN as initialization point can also be seen as a regularization method.

Training from scratch The last solution is to train a proven architecture designed for a related visual task but with a random initialization of the learnable parameters. This method is named training from scratch. We can also argue that the bare architecture of a successful network is in itself a form of transfer learning, as this architecture has proven its relevance to the task of image classification. Whether the architectures are hand-crafted or learned by hyper-optimization, they contain some priors about the input space or the task from the source domain. These priors can be useful for the target task if it is related to the source one.

Note that it is also possible to transfer only some part of the pre-trained weights of the network while keeping the same architecture, as in [Yosinski et al. 2014].

Another hybrid approach of transfer learning is proposed by reusing only the scaling of the pretrained weights but not the weights themselves, see [Raghu et al. 2019].

Pertinence of deep transfer learning A large number of studies have compared the performance of the off-the shelf and the fine-tuning approaches [Chatfield et al. 2014; Girshick et al. 2014; Kornblith et al. 2018]. Fine-tuning typically achieves higher accuracy, especially for larger datasets or datasets with a larger domain mismatch from the training set [Yosinski et al. 2014; Chu et al. 2016a; Huh et al. 2016; Cheplygina 2019; Romero et al. 2020]. Yosinski et al. [2014] show that the first layer of AlexNet can be frozen when transferring between subsets of ImageNet without performance drops, but freezing later layers produces a substantial drop in performance. Huh et al. [2016] also study the AlexNet model. They report that at a fixed budget, training with fewer classes but more images per class performs slightly better at target tasks than training with more classes but fewer images per class. Moreover the number of images per class and the number of classes have a positive (but small) influence on the performance of the transferred model. Chu et al. [2016a] provide a complete illustration that the standard practice of fine-tuning a pretrained AlexNet model is almost always beneficial for seven different datasets (from birds to plankton images). They show that depending on the size of the target set and the distance to the source dataset, it can be a necessity to freeze the transferred layers. Some works highlight that it exists an optimal number of layers to freeze depending on the source and target set (see [Lecoutre et al. 2017; Cetinic et al. 2018] for deep transfer for art analysis). Other works even tried to automatically find this optimum depending on the target set thanks to Bayesian optimization [Basha et al. 2020a; Basha et al. 2020b].

Nevertheless, some works have recently shown that ImageNet pretrained models are maybe not as generic as previously thought. The Kornblith et al.'s work [2018] try to evaluate the implicit hypothesis in computer vision research that models performing the best on ImageNet necessarily perform the best on other vision tasks. In order to prove this, they compare the performance of 13 CNN on 12 image classification tasks in the three previously mentioned settings. In the case of off-the-shelf feature extraction, ResNet models outperform the other ones and the performance on ImageNet is a poor

proxy for the performance on the target set. Networks yield better performance when they are fine-tuned. In this case, they observe a stronger correlation between accuracy on ImageNet and accuracy on target tasks. Moreover, ImageNet pretraining accelerates convergence. For some fine-grained classification datasets with small amounts of data[8], training from scratch perform competitively with fine-tuning but in the other cases, the fine-tuning approach achieves higher performance.

Depending on the discrepancy between the source and target tasks or the specificity of the target domain, developing a specific architecture may lead to better performance. Indeed, it has been shown on two large scale medical imaging tasks that transfer learning of pretrained models offers little benefit to performance compared to a lightweight CNN [Raghu et al. 2019]. ImageNet accuracy is not correlated to accuracy on medical datasets. Transfer learning using X-rays images from the same anatomical region outperforms every other methods whereas using images from other anatomical regions provides similar performance to using ImageNet as source set [Romero et al. 2020]. In the recent survey by Cheplygina [2019], twelve papers with various sources and/or target datasets from different medical domains are compared. The author urges the medical imaging community to conduct larger systematic comparisons into this important topic because there is not a clear answer about the superiority of using a source set from medical domains or not. The same kind of meta-analysis should be conducted for other image domains such as radar or artistic one. In Section 4.3.2, we will compare the different setups of transfer learning for art classification, in addition to the use of natural or artistic images as source set.

The guest of universal feature representation On the one hand, some recent works push this pre-training paradigm further by pre-training on datasets that are 300 times (in-house JFT-300M dataset [Sun et al. 2017]) and even 3000 times (in-house Instagram dataset [Mahajan et al. 2018]) larger than ImageNet. These works tend to learn a universal feature representation. This last work demonstrates significant improvements on image classification transfer learning tasks, the improvements on object detection being relatively small (on the scale of plus 1.5 AP on MS COCO). Recently, the Open Images V4 dataset, of 9.2M images have been made public [Kuznetsova et al. 2020] and could replace ImageNet in the future. The marginal benefit from this kind of very large scale pre-training data maybe be questionable, or for ethical reasons, since the non-openly available datasets come from micro-workers or uninformed platform users [Tubaro et al. 2019], either for ecological reasons [Strubell et al. 2019]. Hence the emergence of the weakly supervised learning paradigm to extract more information from the training sets (see Section 2.4).

Besides, He et al. [2019] show a surprising result: ImageNet pre-training has limited impact on MS COCO object detection and segmentation tasks. The model[9] trained from scratch with enough iterations reaches the same performance as the fine-tuning method even with a small portion of the training set, down to 10k images[10]. ImageNet pre-training speeds up convergence early in the training, but it does not necessarily

[8] A few thousand labeled examples.

[9] A Mask R-CNN model [He et al. 2017] with a ResNext backbone is studied.

[10] However, one must keep in mind that, within a fixed number of images, more bits of information are provided to the model when we train it for object detection or segmentation task compared to classification task. Bounding boxes or pixel labels are more informative that image-level labels.

provide regularization or improve the final target task accuracy as mentioned before. As also mentioned before the use of a given architecture can be seen as a kind of transfer learning.

Domain adaptation In the following work and more precisely in Chapter 4, we apply the most classic technique: we transfer a model pre-trained on the source domain and extract fixed deep features or fine-tune it on the target data.

More effective techniques from the *domain adaptation* field consisted in fine-tuning on both domains at the same time [Sun et al. 2016a; Sun et al. 2016b; Inoue et al. 2018]. Moreover these techniques belong to the transductive setup [Pan et al. 2010] and the symmetric feature-based transfer learning setup [Weiss et al. 2016]. Indeed source and target are transformed into a common features space, sometimes called domain-invariant features space. In the framework of domain adaptation, the task (output space) is the same whereas the input space is not. For instance, it consists in recognizing the same objects in DSLR and webcam images [Sun et al. 2016a; Sun et al. 2016b], in natural images and artworks [Li et al. 2017a] or in sunny and foggy conditions [He et al. 2020]. Readers are referred to the domain adaptation for visual applications survey by Csurka [2017] for a broader description of this field.

Nevertheless the off-the-shelf or fine-tuning methods are computationally more efficient especially when training sets are different in size.

2.1.2.2 Applications of Convolutional Neural Networks

As previously recalled, CNN based models yield state-of-the-art performances in many areas of computer vision such as object detection [Girshick et al. 2014; Girshick 2015; Ren et al. 2015; Redmon et al. 2016; Lin et al. 2017; Liu et al. 2020], semantic segmentation [He et al. 2017], visual question answering [Wu et al. 2017], pose estimation [Güler et al. 2018], image manipulation [Viazovetskyi et al. 2020], etc. CNNs also provide state-of-the-art in numerous other domains such as medical images (for tumor segmentation for instance [Vakanski et al. 2020]), astronomy images (for galaxy detection [Ackermann et al. 2018]) or artistic ones (e.g. for object recognition in paintings [Crowley et al. 2014]). New kinds of artificial neural networks are emerging those years and may soon dethrone the CNNs, including the Capsule Networks [Sabour et al. 2017] and Vision Transformers [Dosovitskiy et al. 2020].

Object Detection Among these possible applications, object detection is maybe one of the most studied. Object detection consists in recognizing a given visual category within an image but localizing it (i.e. be able to output a bounding box around this object). This task is important for several reasons. First, it can be the first step needed before a fine-grain classification of the object of interest[11]. Second, it permits providing relative spatial information between objects[12]. Finally, object detection may be a way to reduce shortcut learning [Geirhos et al. 2020]. The object detection task is difficult due to viewpoint and scale variations, deformation, occlusion, illumination conditions,

[11] For instance, it is common to first run a face detection algorithm before running an identification model.

[12] This information may be needed for action recognition or relation inference.

background clutter and intra-class variation as other visual recognition tasks. However, the need of bounding the object increases the difficulty as it requires to learn exactly what belongs to the object of interest within the image. Nevertheless, this task is easier than semantic segmentation[13].

2.2 Understanding Convolutional Neural Networks

The deep learning community has provided a plethora of experimental tools for trying to better understand deep neural networks. These tools include feature visualization [Erhan et al. 2009; Zeiler et al. 2014; Olah et al. 2017], pixel attribution [Simonyan et al. 2014; Selvaraju et al. 2017; Smilkov et al. 2017; Sundararajan et al. 2017] and network comparison [Raghu et al. 2017; Kornblith et al. 2019], besides theoretical intends [Paul et al. 2015; Mallat 2016]. In Section 2.2.1, we present first one branch of feature visualization based on optimization. Then we introduce some of the tools used for network comparison (Section 2.2.2).

2.2.1 Visualizing Convolutional Neural Networks

The main idea of the visualization of CNNs is to represent them in the image domain to have a better interpretation or explanation of their behavior.

On the one hand, there is a "dataset centric" approach[14]. The most basic technique is to select images from either the training or test sets which maximize certain channel activations. *Attribution* or saliency[15] methods consist in identifying pixels of the input image that contribute the model decision [Simonyan et al. 2014; Selvaraju et al. 2017; Smilkov et al. 2017; Sundararajan et al. 2017]. This method is mainly based on inspecting the magnitude of the derivative of the score for a given class with respect to the input image. Another solution is to look at the reconstruction of the input image from the deep features of a given layer [Mahendran et al. 2015].

On the other hand, *feature visualization* by optimization (or activation maximization) is said "network centric". It consists in synthesizing an image that maximizes the activation of some part of the network [Erhan et al. 2009; Simonyan et al. 2014; Olah et al. 2017]. Thus, this part of the network may be selective for the synthesized visual patterns.

2.2.1.1 Feature Visualization via Activation Maximization

We here focus on the popular interpretability method of feature visualizations via activation maximization.

Definition Feature visualization answers questions about what a deep network is responding to by generating examples that yield maximum activations for a given

[13] Semantic segmentation consists in classifying each pixel of the object of interest.
[14] "Image centric" is maybe a more relevant name.
[15] Also called sensitivity.

neuron, channel or layer. First introduced by Erhan et al. [2009], optimization based methods of feature visualization maximize the mean[16] activation of a particular channel by doing a gradient ascent on this mean activation with respect to the input image. The goal is to obtain an image that will maximize the activation of a given channel:

$$\hat{I} = \arg\max_{I} \; mean\, f_k^l. \tag{2.4}$$

Starting from a random initialization, the gradient ascent will create iteratively an image $\hat{I}$ that maximizes the activation of the channel k. The synthesized image will be named *Optimized Image*[17] in this manuscript. The seminal work of [Erhan et al. 2009], apply this method to Deep Belief Network [Hinton et al. 2006]. They have shown that the first layer channel obtained by training on natural images are "Gabor-like features". The second layer corresponds to corners detectors. Higher-level layers are quite complex but interpretable in some sense. When the network is trained on MNIST (a hand-written digits dataset), some optimized images look like pseudo-digits. Higher-level layers learn meaningful combinations of lower level features. Moreover most random initializations yield the same prominent visual pattern.

This method is applied for the first time to CNN by Simonyan et al. [2014], to visualize the optimized images corresponding to the class neurons. They show which kind of stimulus, those neurons respond to. For each class, the visual pattern within the images seems to belong to the class considered. Nevertheless, to achieve results that are visually acceptable and understandable by human, it is a necessity to add some constraints to the optimization, thus avoiding getting adversarial structured noise [Szegedy et al. 2014] or the fooling images investigating in Nguyen et al. [2015]. Constraints can be L_2-regularization [Simonyan et al. 2014], blurring [Nguyen et al. 2015], jitter and multi-resolution [Mordvintsev et al. 2015], GAN generator parametrization [Nguyen et al. 2016], stochastic geometric transformation or color-decorrelated based on Fourier transformation [Olah et al. 2017]. Nevertheless, a recent paper from Engstrom et al. [2019] show that adversarially robust models provide good feature visualizations without enforcing any priors or regularization. Thanks to these regularizations, Olah et al.'s work [2017] became a milestone because its illustrates the hierarchical aspect of the internal representations of a CNN trained on ImageNet with stunning feature visualizations, as it can be seen in Figure 2.5. It is important to say that gradient-based feature visualization only represents one instance of the set of the possible images that can fire a given channel. This has been illustrated by the diversity loss term used in [Olah et al. 2017], based on the Gram matrix of the feature maps[18].

Many researchers considered that feature visualizations are interpretable and that "features can be rigorously studied and understood" [Olah et al. 2020b]. As defined in [Olah et al. 2018], some channels of the deep models can be considered as a *detector* of a specific visual pattern more or less complex (curve, floppy ears, etc.). Visualization

[16] Other statistics that the mean can be used, for instance Mordvintsev et al. [2015] use the Frobenius norm of the whole layer.

[17] Also known as "Numerically computed image" [Simonyan et al. 2014] or "optimized example" [Olah et al. 2017].

[18] This diversity term is inspired by the texture synthesis with CNN method of Gatys et al. [2015b]. We will introduce it later, in Section 2.5.2.1.

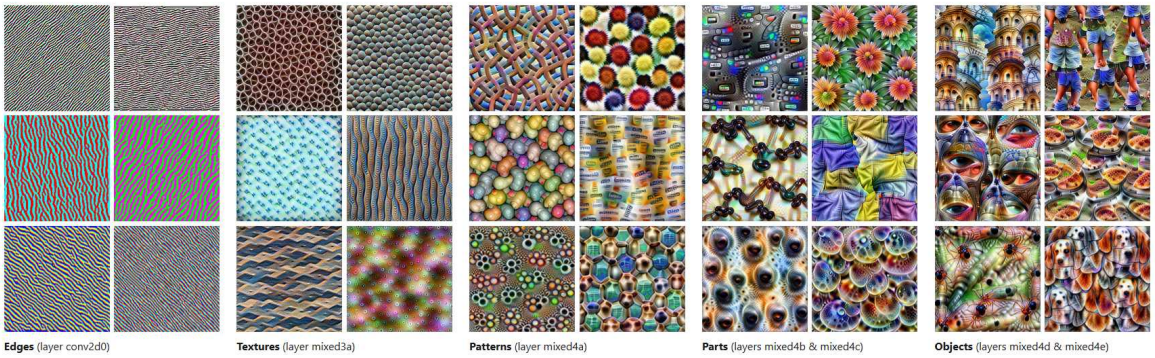


Figure 2.5: Optimized images for different layers of InceptionV1 CNN trained on ImageNet. Courtesy of [Olah et al. 2017].

of the optimized images also permits regrouping the filters of the first layers of an InceptionV1 model in some comprehensible groups [Olah et al. 2020a]. The same team proposed reverse engineering of one of the most common patterns in visual content: curve [Cammarata et al. 2020]. They propose a rigorous protocol to illustrate both the role and characteristics of a certain number of channels detecting curve and the hierarchical relations between them. This research team also claims in [Olah et al. 2020b] that universal detectors can be found in different models trained on ImageNet . Moreover, a recent perceptual study shows that feature visualizations provide helpful information to better understand the information processed by CNN, even if looking at the maximal activation images is more helpful [Borowski et al. 2020]. Despite this, optimization-based methods can isolate the causes of some behaviors from mere correlations. A neuron may not be detecting what one initially thought it would. For instance, the DeepDream [Mordvintsev et al. 2015] algorithm can generate a visualization of dumbbells with an attached arm. Thus it can reveal a bias in the training dataset which contains numerous images of weightlifters holding dumbbells. The CNN might consider the arm as part of the dumbbell object or this element as essential to classify the image. Moreover, optimized images are more flexible, they can be used for looking at the interaction between channels [Olah et al. 2017] or for looking at some particular directions in the feature space [Carter et al. 2019]. Szabó et al. [2020] visualize the impact of the fine-tuning of a network on fine-grained, natural images datasets. They demonstrate various properties of the transfer learning process such as the speed and characteristics of adaptation, neuron reuse and characteristic spatial scale of the optimized images. In another direction, several papers have proposed methodology to determine the way the different features contribute to a classification by combined feature visualization techniques with attribution methods [Olah et al. 2018; Carter et al. 2019; Shi et al. 2020a].

Do CNNs learn meaningful features? Works such as [Erhan et al. 2009; Simonyan et al. 2014; Olah et al. 2017; Olah et al. 2020a] tend to show that a deep neural network learns meaningful and high-level features (for instance eye detector or dog head detector) whereas others show that those networks are primarily detecting textures [Geirhos et al. 2019] or even statistical regularities on the dataset [Jo et al. 2017]. Indeed, visually imperceptible modifications of the image may lead to a com-

pletely wrong prediction, as it has been shown by Szegedy et al. [2014], as mentioned before. These adversarial examples may be considered as an illustration of the lack of robustness of those models. Moreover, [Geirhos et al. 2019] show that ImageNet-trained CNNs are strongly biased towards recognizing textures rather than shapes. They illustrate their claim by observing a huge performance drop when changing the texture of the image while keeping the shape intact[19]. A related conclusion is made by Brendel et al. [2019] because they observe that an image only containing the texture information[20] can most of the time still be correctly classified by CNN. Jo et al. [2017] find that CNNs trained on an unfiltered natural image dataset exhibit a tendency to focus on the image statistics of the training set, yielding significant drop on high frequency filtered images that are still identifiable by humans. More examples of differences between what is used by CNNs for recognition and the human visual system can be found in [Geirhos et al. 2020]. In this work, the authors introduce a taxonomy of shortcut learning phenomenons for deep learning model and propose strategies to diagnose and understand them.

Nevertheless, several works tend to show that CNNs learn high levels of abstraction. Nguyen et al. [2015] present several ways to create images that completely fool deep networks (with evolutionary algorithms or gradient ascent) but that are unrecognizable to humans. It can be white noise-like images or psychedelic ones composed of simple and repetitive geometric patterns. But they were also able to generate highly abstract images that are both recognizable by networks and humans. CNNs may be able to learn a high level of abstraction but fails to deal with out-of-the-distribution samples. Most strikingly, White [2019] used a drawing system (only able to vary strokes placement, thickness and color) to generate abstract drawings with strong responses across different ImageNet classes for one given CNN. These abstract images may also be recognized by humans and by other pretrained CNN architectures indicating that these images generalize well. Thus, understanding CNN internal representations is still a much discussed and much studied topic. We only mentioned few of the attempt to this open problem.

2.2.1.2 Feature Inversion

Feature Visualization via activation maximization is different from the Mahendran et al.'s work [2015] about inverting deep representations. The *feature inversion* method reconstructs an input from its deep representation at a given layer. This algorithm achieves an approximate CNN inversion by minimizing the Frobenius distance between the feature representation of the input image and the synthesized one by gradient descent. For the input image I and the layer l the loss minimized with respect to $\tilde{I}$ is:

$$\mathcal{L}_{Geom,I,l}(\tilde{I}) = \frac{1}{2}\sum_{p=1}^{m_l} \|f^l - \tilde{f}^l\|_{\mathcal{F}}^2, \tag{2.5}$$

with f^l the feature maps of the input image I and $\tilde{f}^l$ the one of the synthesized one $\tilde{I}$. Some constraints such as the Total Variation can be added to this optimization to

[19] Thanks to neural style transfer techniques that we will describe later, in Section 2.5.2.2.

[20] They generate texturized images by applying the texture synthesis method from Gatys et al. [2015b], that we will describe later, in Section 2.5.2. This technique is related to the neural style transfer algorithm.

get better perceptual results. Mahendran et al. [2015] illustrate that the feature maps encode some geometric content of the image. The problem of inverting a network is tackled in a different way by Zeiler et al. [2014] with the use of an unsupervised deconvolutional network attached to each layer.

2.2.2 Networks Comparison

Comparing different CNN architectures trained on different datasets is not an easy task but it may help to understand them.

2.2.2.1 Parameter Similarity

This comparison can be made in the parameter space, for instance by computing the ℓ_2 distance between learned weights of two differently-initialized networks, as in [Neyshabur et al. 2020]. In [Bernstein et al. 2020], they derive a distance on neural networks based on a perturbation analysis of the neural network function and Jacobian. This distance has been used to provide a learning process which does not need a learning rate tuning, but it cannot be used for comparing what has been learned by different architectures.

2.2.2.2 Feature Similarity

The comparison can also be made in the feature space. We will now quickly describe some of the methods developed for computing a feature similarity between different layers or different models. The goal is to compare how similar are the representations learned by two layers or two networks. Li et al. [2015] apply linear correlation and mutual information analysis to study one-to-one mappings between channels[21], and found that some representations are shared by differently-initialized networks, but others are not. They applied a spectral clustering algorithm to find many-to-many mappings that permits clustering groups of neurons among two different networks. Their approach finds matching Gabor filters and also matching color blobs. The idea of neuron activation subspace matching is also developed by Wang et al. [2018a]. They conclude that differently-initialized networks are not as similar as expected in terms of subspace match.

Another solution is to use the statistical tool named Canonical Correlation Analysis (CCA). CCA seeks two linear combinations of two tensors which have maximum correlation with each other. It can be used to compute a similarity between these two tensors[22]. In the case of CNNs these two tensors are the feature maps of different models or layers.

Two adaptations of CCA have been proposed to deal with the huge computational cost due to the deep neural networks and the size of training sets:

- Singular Vector Canonical Correlation Analysis ((SV)CCA) preprocesses neuron activation vectors with a Singular Value Decomposition to remove low variance directions [Raghu et al. 2017]

[21] Named neurons in this paper.

[22] For instance, the mean squared of the p CCA correlations is one of the possible similarity index. This index is equal to the double sum of the dot product of the normalized principal components of the two considered tensors divided by p the number of features (in the smaller representation).

- Projection Weighted Canonical Correlation Analysis (PWCCA) is a projection weighted mean of the CCA correlation coefficients [Morcos et al. 2018].

Morcos et al. [2018] present a simple mathematical overview of CCA, which is completed by Kornblith et al. [2019]. In this last article, another variant introduced by Cortes et al. [2012] and named Centred Kernel Alignement (CKA) is used to compare CNNs.

CKA is a normalized version of the Hilbert-Schmidt Independence Criterion, i.e. the squared Hilbert-Schmidt norm of the cross-covariance operator to take into account nonlinear kernels on the two tensors to compare. The normalization permits to have an invariance to isotropic scaling measures. In the linear case, CKA is equal to the squared Frobenius norm of the dot products (similarity) between the feature maps X and Y. It is equivalent to the normalized version of the dot product between reshaped inter-example similarity matrices (also called Gram matrices) :

$$linear\ CKA(X,Y) = \frac{\left\|X^{\mathrm{T}}Y\right\|_{\mathrm{F}}^{2}}{\left\|X^{\mathrm{T}}X\right\|_{\mathrm{F}}\left\|Y^{\mathrm{T}}Y\right\|_{\mathrm{F}}} = \frac{\left\langle \mathrm{vec}\left(XX^{\mathrm{T}}\right), \mathrm{vec}\left(YY^{\mathrm{T}}\right)\right\rangle}{\left\|XX^{\mathrm{T}}\right\|_{\mathrm{F}}\left\|YY^{\mathrm{T}}\right\|_{\mathrm{F}}} \tag{2.6}$$

This similarity score is invariant to isotropic scaling and orthogonal transformation but not to invertible linear transformation. Indeed, scale information of the feature maps are meaningful as it has been illustrated by the texture synthesis with CNN [Gatys et al. 2015b] based on the Gram Matrix of the feature maps.

The difference between the mean squared CCA correlation and the CKA index is the dot products are weighted by the amount of variance explained by the principal components. CKA is placing greater emphasis on similarities between theses components that are responsible for more variance than in the original representation.

These similarity indexes can be used for determining the correspondence between the hidden layers of CNN trained from different random initializations and different architecture. In [Raghu et al. 2017; Morcos et al. 2018], the authors discover that lower layers in neural networks converge first, before higher layers. Better, lager or wider networks converge to more similar solutions in the feature space. Raghu et al. [2019] use (SV)CCA to analyze the hidden representation of different models trained on medical images. They show that larger models change less through training, especially at the lowest layers, even if they are randomly initialized. Moreover, Kornblith et al. [2019] show that models trained on CIFAR-10 and CIFAR-100 develop similar representations in their early layers but not in the later ones. Neyshabur et al. [2020] use CKA to compare hidden representations of the different models, fine-tuned or trained from scratch on different image datasets (clipart, X-ray). They observe that the ImageNet pretrained model and the fine-tuned one are highly similar across layers. This similarity is even high between two instances of a fine-tuned model, as they learn to classify the same dataset, from the same starting point. On the contrary, the similarity is very low between a fine-tuned model and a model trained from scratch or between two instances trained from scratch. Note that the feature similarity is stronger in the penultimate layer than the other ones, for the previously mentioned cases. Thus, two fine-tuned instances are similar, especially compared to the randomly-initialized case. The fine-tuning does not change much the hidden representations of the model. The same kind of conclusions can be drawn from the distance of the models in the parameters space, the study of the loss landscape and the kind of mistakes made.

Feature visualization techniques, metrics in both the feature and parameter spaces will be used to provide quantitative analysis of the changes introduced by the learning process on artistic dataset, in Section 4.4.

2.3 Deep Learning for Art Analysis

Applied mathematics and computer vision algorithms have been used for art analysis applications long before the rise of modern deep learning. Although, art analysis also benefits of visual recognition performances of CNNs , there is some continuity in the research works. Nonetheless, artistic images are still a challenging subject due to the high variability of artistic techniques, genre or level of abstraction. In this section, we will present the deep learning methods applied to art analysis. First, in Section 2.3.1, we will give a brief insight of algorithms for art analysis before deep learning methods. Then, we will present different ways to use CNNs for art analysis, from supervised to unsupervised setups, in Section 2.3.2. Section 2.3.3 will be devoted to the main improvement directions. Section 2.3.4 will be about off-the-shelf applications of CNNs to art analysis without any training. In the last Section 2.3.5 we will discuss about the philosophy of using deep learning model for large scale art analysis.

2.3.1 Algorithms for Art Analysis Before Modern Deep Learning

Image Processing has been applied to artistic images analysis in several ways. It has been used for estimating information about the tangible material of the artworks, for instance with crack network detection in the painting layer [Mueller et al. 1993] or removal the texture artifacts of X-ray artwork acquisition [Heitz et al. 1987], but also for more visual recognition tasks as authentication, date estimation or semantic recognition. We will focus on this late category of works. In parallel to advances in computer vision and image analysis, studies have shifted from local image features to more global image features.

2.3.1.1 Statistical Image Properties

First, a number of statistical image properties have been studied in artworks as it has been done for natural images. These properties have been investigated in images of artworks that represented different artistic styles, techniques or cultural provenances. For the local features, we can mention for instance the moments of the pixels [Graham et al. 2007] or their intensity [Graham et al. 2008]. A number of global statistical image properties have been studied in artworks, the fractal dimension [Taylor et al. 1999; Braun et al. 2013], the slope of the log-log Fourier spectral plots [Graham et al. 2007; Graham et al. 2008; Braun et al. 2013], self-similarity [Braun et al. 2013], entropy of edge orientations [Redies et al. 2017b] or all at the same time [Redies et al. 2017a]. The number of images used to compute these statistics also increase over time. These works are numerous and conclusions diverse. For instance, [Taylor et al. 1999; Taylor 2002] focused on the drip paintings of Jackson Pollock and found that he painted fractals similar to those found in nature. Graham et al. [2007] and Graham et al. [2008] detect

approximately scale-invariant statistical properties of western and eastern paintings. Graham et al. [2007] and Graham et al. [2010] show similarity between the statistical properties of artistic and natural images. Redies et al. [2017b] exhibit that Western artwork may be characterized by a lack of correlations between edge orientations. It is still unclear how universal these properties are across artistic styles and times. Most of the time, the works try to make a link between computer vision methods and their neural underpinnings in the human visual system, as in [Graham et al. 2007; Redies et al. 2017b].

2.3.1.2 Art Classification

A complete description of the features and classifiers used for style classification of paintings before 2005 can be found in the Lombardi PhD manuscript [2005]. These features are mainly related to pixels' intensity and color histograms, gradients of the image, responses to common filters as Sobel or Gabor, norms of wavelets coefficients or Fourier transforms and marginal of color histograms. The classifiers come from the whole spectrum of machine learning from MLP to naive Bayes classifier. Next, researchers expanded the range of features, classifiers or tasks. Zujovic et al. [2009] propose using wavelets decomposition and color information with SVM or MLP for painting genre classification. Johnson et al. [2008] perform a detailed analysis of brushstrokes in the work of Van Gogh using Gabor, Complex and D4 wavelets in combination with SVM. They conclude that brushstroke analysis is helpful in artist attribution but that it also depends on external factors like canvas degradation and pigment loss. [Jafarpour et al. 2009] propose to use wavelets decomposition of patches of Van Gogh's paintings with random forest or SVM for date classification. In a related paper, Polatkan et al. [2009] use similar technique for authentication purposes instead of stylistic analysis. These techniques based on wavelets decomposition are characterizing the brushstrokes scale for precise classification task with a small number of high resolution artworks (hundreds at the maximum) for training. Several of the previously mentioned features such as moments of the pixels intensity distributions or color distributions, gray-level co-occurrence matrices, energy of discrete wavelet coefficients and others have also been used for painters classification with lower-resolution images [Cetinic et al. 2013]. Several works propose to use features from the computer vision field. Bressan et al. [2008] use SIFT features and local color statistics for finding similarity between paintings and infer connections between artists with graph analysis. SIFT have also been used for artistic style classification [Saleh et al. 2014] but also for artist, period, material or type classification [Mensink et al. 2014]. Crowley et al. [2013] propose a weakly supervised pipeline to recognize images of gods and animals in decorations on classical Greek vases. Their method first combines a text mining method for finding visually consistent clusters among the noisy and image-level labels. After a cleaning of the irrelevant regions of the images, they use a MIL formulation to identify the regions of the images containing the gods or animals. Finally, a sliding windows classifier based on Deformable Part Model (DPM) is fully supervised with these pseudo-labels.

2.3.1.3 Image Retrieval and Cross-depiction

Image retrieval Another classical computer vision task also relevant for art images is content-based image retrieval. It consists in retrieving images based on similarities in their contents (textures, colors, shapes, etc.) to a user-supplied query image. Hurtut et al. [2008] propose to model the difference between the spatial organization of color of two images as an optimal transportation problem between coarsely sampled thumbnail of the images. They address the decision stage with an adaptive matching criterion based on a statistical approach, itself founded on an a contrario approach. Sun et al. [2009] propose a method to detect full or part of illegal copies of line-drawing images with HOG features, stable extremal region detection and quantization by hash table for speedup. A method for the extraction of stroke contours based on a selection of the level lines of the topographic map is proposed in [Hurtut et al. 2011]. This method can be used for image retrieval in line artworks datasets.

Cross-domain image retrieval Early works on cross-domain image comparisons were mostly concerned with sketch retrieval, see e.g. [Del et al. 1997]. Given a user-sketched template, the algorithm returns images which contain an object that matches this template (with some elastic deformation). Based on HOG features pyramid for image representation, Shrivastava et al. [2011] propose to use an *exemplar-SVM* [Malisiewicz et al. 2011] with an iterative hard-negative mining procedure to cross-domain image retrieval (finding the same building in different depictions). This exemplar-SVM is trained with one unique positive example (the query) and a lot of negative ones. One interesting application of this work is the estimation of the point of view used in a real monument depiction. In contrast to [Shrivastava et al. 2011] who work on 2D images, Aubry et al. [2014] propose to learn mid-level 3D scenes that are reliable for detecting 2D depictions of the scene despite changes in the domain. This approach allows to recover the approximate camera viewpoint of the artwork. This representation of 3D scenes for alignment to 2D depictions is based on densely sampled patches represented by HOG at multiple scales. Then an exemplar-SVM is learned as previously. To make the model more robust, the authors add an extra step to discard unstable patches across changes in viewpoint

Cross-depiction Wu et al. [2014] propose a concept of cross-depiction going one step further in the conceptualization: recognizing generic objects regardless of whether they are from photographs, paintings or comics. The shift is that we go from the image retrieval by content task (find images with the same shape or color composition) to a semantic recognition one (recognize a horse in print or watercolor). They use HOG features [Dalal et al. 2005] and model objects through graphs of labels in a similar way to [Felzenszwalb et al. 2010]. It is advanced in [Hall et al. 2015] that structured model (DPM or fully connected multi-labeled graph with HOG features) are more prone to succeed in cross-domain recognition than appearance-based models (such as Fisher vectors of SIFT or Bag-of-Words (BoW) of HOG or SIFT).

2.3.2 Use Cases of Convolutional Neural Networks for Art Analysis

Applications of computer vision tasks dealing with hand-drawn or computer generated non-photographic images benefited from the resurgence of CNNs. One point in common between almost all works in the field is the reuse of architectures that were originally designed for photographs tasks. Some works use the pre-final features of a network as the only features retained to represent an image and do not fine-tune the network for the task at hand. Other methods allow for a certain amount of fine-tuning and add a specific network after the original architecture. In this paragraph, we will mention various applications of CNN to art analysis from supervised to unsupervised computer vision tasks with a focus on image classification, object detection and instance recognition.

2.3.2.1 Art Classification

Several works have tried to transfer the tremendous classification capacity of CNNs for photographic data to perform alternatives classification tasks, possibly on different modality of images.

Off-the-shelf feature extraction In the seminal work of Crowley et al. [2014], it is shown that recycling CNNs directly for the task of recognizing objects in paintings, without fine-tuning, yields surprisingly good results compared to Fisher Vector Representation. Moreover, the authors estimate that there is a drop between training the final classifier on natural images instead of artistic images from the same domain. As this drop is not too strong, they propose an on-the-fly system for arbitrary images recognition. The user can ask for any query. The training images are crawled from Google Images. They are used to learn a classifier on the CNN features of those photographs. Then, the classifier is applied on CNN features of the art images to output images related to the user's query. The same fall in performance in the training on photo, testing on art images protocol have been exhibited for CNN and handcrafted features based methods by Hall et al. [2015]. These two works assert that CNN features are better than handcrafted ones. Similar conclusions were also made for artistic style classification in [Karayev et al. 2014], for object recognition in artistic drawings by Yin et al. [2016], in paintings by Lang et al. [2019], visual link retrieval by Seguin et al. [2016] and for different art classifications tasks[23] by Strezoski et al. [2017a]. On the contrary, off-the-shelf features from AlexNet are often beaten by other images representation for style, scene or genre classification [Saleh et al. 2016]. Regarding the used features, their best result is achieved with the feature fusion method which also included CNN-based features.

The on-the-fly recognition system mentioned before has been extended later with the use of a detection model [Crowley et al. 2016]. They propose a relatively basic methodology in which for each image the region with the highest (class agnostic) "objectness" score is used instead of the whole image. They illustrate once again that the

[23]They compare the use a CNN as an off-the-shelf feature extractor to a SIFT based method [Mensink et al. 2014]. The CNN model leads to better performance when using with a multi-tasks training setup.

deep features from CNN trained on natural images are good enough for art images classification but training the final classifier on art images provide better results than using natural ones.

Fine-Tuning However, when the training set is big enough, fine-tuning the pre-trained model in a supervised way may be the best solution. This methodology has been applied in several setups such as artistic style classification [Hentschel et al. 2016; Tan et al. 2016; Bianco et al. 2017; Huang et al. 2017a; Lecoutre et al. 2017; Mao et al. 2017; Elgammal et al. 2018], object recognition in drawings [Yin et al. 2016], iconographic characters recognition [Madhu et al. 2019], face classification [Tian et al. 2020], genre classification [Florea et al. 2017; Cetinic et al. 2018], scene classification [Florea et al. 2017], author classification [Bianco et al. 2017; van Noord et al. 2017; Sabatelli et al. 2018], material classification [Sabatelli et al. 2018], artist nationality classification [Cetinic et al. 2018] or several of those tasks at the same time [Bianco et al. 2019; Garcia et al. 2019].

Many works demonstrate that fine-tuning models pre-trained for object recognition on ImageNet perform better than a model fully trained from scratch on art images or a pretrained model used as an off-the-shelf feature extractor [Hentschel et al. 2016; Seguin et al. 2016; Tan et al. 2016; Huang et al. 2017a; Lecoutre et al. 2017; Mao et al. 2017; Elgammal et al. 2018; Garcia et al. 2019]. In [Cetinic et al. 2018], the authors evaluate the impact of domain-specific weight initialization and the kind of fine-tuning used (number of frozen layers for instance). They compare different pre-training on different natural images datasets and they highlight that the bigger (in terms of training images and number of labels), the better will be the results. They show that fine-tuning networks pre-trained for scenes and objects recognition yields better results than fine-tuning networks pre-trained for object recognition for artist, genre, style and time period classification. Tian et al. [2020] also fine-tune ImageNet pretrained CNN for gender or status classification. They reveal that newer and larger architectures often achieve better performance. Other works fine-tune CNN pretrained on specific dataset. For instance, Madhu et al. [2019] use a VGG model pretrained on a face dataset for recognizing Mary or Gabriel characters. Using models pretrained on artistic datasets may also help to get better results. Bianco et al. [2017] pretrain their CNN on the Kaggle dataset Painterbynumbers. Sabatelli et al. [2018] demonstrate that the same architecture fine-tuned on the Rijksmuseum dataset yields better results compared to the sole ImageNet one, for an artistic task (both if used as off-the-shelf feature extractors and if fine-tuned). Several works highlight that it exists an optimal number of layers to freeze depending on the target dataset and the transferred model as mentioned before, [Lecoutre et al. 2017; Cetinic et al. 2018]. Works using models trained from scratch are rare [Florea et al. 2017; Badea et al. 2018] and the one using an ad hoc CNN architecture even rarer [Strezoski et al. 2017b].

Art classification datasets On the one hand, some research teams created huge training datasets, such as BAM!, OmniArt and Art500k. In [Wilber et al. 2017], an annotated database named BAM! of 2.2M contemporary artworks from Behance (a website of portfolios from professional artists) is introduced, on which it is shown that fine-tuning improves recognition performances. The OmniArt dataset introduced in

[Strezoski et al. 2018] contains 1M historical artworks of 4 different types (from craft to paintings). They illustrate that according to the task (style and type classification against school and genre classification), training from scratch on the OmniArt dataset may lead to better performance than fine-tuning a model pretrained on ImageNet. Moreover the kernels of the first layer are similar between the two schemes. Art500k is a large scale visual arts dataset introduced by Mao et al. [2017] containing 500k artworks annotated with artists, style, medium and genre labels. These three large scale datasets are not openly accessible yet. There are no models pretrained on them shared to the community by the research teams. These datasets are important for the computer vision communities to reach comparable performance to the natural images applications. Indeed several works state that the performance of the CNN is positively correlated with the number of samples per class - more samples per class equals better classification capabilities [van Noord et al. 2015; Strezoski et al. 2018].

Besides, most of the artworks datasets contain many different art forms (paintings, prints, sculpture, crafts, etc.) [Mensink et al. 2014; Zhang et al. 2019a] and the ones that contain mostly paintings are smaller [Crowley et al. 2014; Westlake et al. 2016]. Most of the time those datasets only contain style or author metadata [Khan et al. 2014; Tan et al. 2016; Lecoutre et al. 2017] instead of depicted objects or iconographic elements description. Sometimes, some datasets are really specific to a given class of objects such as the eye in paintings [Strezoski et al. 2020], face in pre-modern Japanese Art [Tian et al. 2020], Mary and Gabriel characters [Madhu et al. 2019] or concepts specific to art history [Cetinic et al. 2019b]. In Section 4.2.1, we will introduce a new paintings dataset, on which we perform classification and weakly supervised detection experiments on iconographic classes that could not be learned on photographs, such as Jesus Child or Saint Sebastian.

2.3.2.2 Object Detection

The object detection problem (recognize and locate an object) in artworks has been less studied. Ginosar et al. [2014] show that DPM outperforms other approaches, including some CNNs, for the detection of people in cubist artworks maybe due to the level of abstraction, the fact the model used is not up to date and the relatively small size of the dataset. The YOLO network [Redmon et al. 2016] trained on natural images can also be used for people detection in cubism artworks. [Westlake et al. 2016] propose to perform people detection in a wide variety of artworks (through a newly introduced database) by fine-tuning a network in a supervised way. People can be detected with high accuracy even though the database has very large stylistic variations and includes paintings that strongly differ from photographs in the way they represent people. Wilber et al. [2017] evaluate the performance of CNNs to detect common objects in different types of artworks (such as bicycles, dog or car). They show that fine-tuning a detector on training images from the same domain as the target one leads to better performance than using images from another domain (artistic or not). Strezoski et al. [2018] also propose to fine-tune a detection network trained in a fully supervised manner on classical Pascal VOC classes. Gu et al. [2019] study the performance of common objects detection (person, horse, etc.) in Chinese paintings. They assert that adding a small object proposal network based on a low-level layer improves detection performances. According to them, detecting objects in artworks is

a more difficult task than in natural images because they obtain lower score on the first task. Thus, there is still a performance gap to fill.

2.3.2.3 Visual Link and Instance Recognition

The instance recognition task (i.e. recognize the exact same object in different artworks or modalities) also has benefited from the CNN powerful prediction performance. This task is related to the image retrieval one mentioned before.

Seguin et al. [2016] propose to tackle the visual link retrieval by fine-tuning CNN with triplets of images (two positives and one negative) and a triplet loss. One can consider two images to share a visual link if the visual correlation between them is considered too high to be solely due to randomness, if there is a visual similarity between them. In fact, this depends on the kind of positive pairs provided for the training process. Once the CNN is trained, it can be used to provide a similarity metric between images. As it has been trained with artworks using different techniques, this metric is able to deal with cross-modalities links. Next, it is even possible to provide several images containing the positive pattern wanted by the user and negative images to avoid some parasitic outputs of the model [di Lenardo et al. 2016].

Collomosse et al. [2017] propose a triplet loss based CNN for image retrieval with a sketched shape, and a set of "style" images[24]. Their model learns a feature embedding that be used for measuring style similarity independent of geometric structure.

Shen et al. [2019] propose an efficient but costly method to find near duplicate patterns in different art techniques (oil, pastel, drawing, etc.). The training alternates between two steps: a mining for hard-positive training samples thanks to CNN features extraction and spatial consistency, and one step of CNN fine-tuning with a triplet loss. Then a refinement step is needed to detect instances replica, based on geometric correspondences (Hough voting plus RANdom SAmple Consensus (RANSAC)) and deep features one.

Zero-shot learning for instance recognition using web supervision has also been considered for artwork analysis [Del Chiaro et al. 2019b]. They use webly abundant but noisy annotations to learn efficient alignment of semantic and visual information based on CNN off-the-shelf feature extractor. Once again, more data (image and metadata) provides better performance. Image retrieval from a textual description or the opposite may also be solved by cross-modal CNN models. This approach has been improved by using a context-aware embedding [Garcia et al. 2019].

2.3.2.4 Others Supervised Tasks

The same kinds of methodologies have been applied to other supervised tasks such as visual question answering [Bongini et al. 2020; Garcia et al. 2020] or jigsaw puzzles solving with cultural heritage images [Paumard et al. 2018]. Some works try to solve the cross-depiction problem mentioned before with CNN, for instance in [Li et al. 2017a], where a robust low rank parametrized CNN model is proposed to recognize common categories in an unseen domain (photo, painting, cartoon or sketch) or [Thomas et al. 2018] with data augmentation and adversarial training.

[24] It can be just a set of visual reference images.

2.3.2.5 Unsupervised Learning

Unsupervised learning aims to find the structure and the regularity of an unlabeled dataset for the purpose of clustering or for extracting useful representations. This paradigm has been broadly used with CNNs for visual tasks with natural images but also with art ones. Some works propose to do clustering of art images thanks to deep features. In a remarked work, Elgammal et al. [2018] perform a correlation analysis[25] of learned features extracted from several different style-trained CNN models in order to understand how learned representations are related to art history methodologies for identifying styles.

Wynen et al. [2018] propose to learn without supervision a sparse dictionary on top of means and variances of VGG features at different layers. This method enables approximating any image with a convex combination of the dictionary atoms (named archetypes). Each of these archetypes can be visualized by a synthesis texture to understand its characteristic stroke, style or colors. This method can also be used for style transfer. This method has been compared by Huckle et al. [2020] to the use of raw VGG features or Gram matrices of the VGG features as an unsupervised style representation for art images. Surprisingly, the raw features provide the best results to capture style in their dataset of contemporary art.

Castellano et al. [2020a] propose to obtain visual similarity score by computing the average over the nearest neighbor of the ℓ_2 distance between a reduced version of the pretrained CNN features of the artworks. On the contrary to [Seguin et al. 2016], this approach is unsupervised. Moreover they use it to estimate a graph of the "influence" between artists. Castellano et al. [2020b] propose to perform a clustering on the embedded features of a fine-tuned convolutional autoencoder to cluster artworks. This method provides better clustering than using a frozen pre-trained autoencoder. Their method is able to recover the three main artistic periods of Pablo Picasso's production.

2.3.3 Improvement on Convolutional Neural Networks for Art Analysis

The use of CNNs for art analysis by the computer vision community has yielded to several improvements in diverse directions. We will briefly present some of them now. First, we will mention multi-resolution strategies, then the use of neural style transfer and next the adaptation of CNN architectures. The last section will be dedicated to the understanding of CNN trained on artistic images.

2.3.3.1 Multi-resolution Strategies

As artworks are often composed of objects or characters at different scales, several works proposed multi-resolution strategy to consider both fine details and coarse structures within images. Bianco et al. [2017] propose to use a three branches model. Two of them are fed with random crop of size 227×227 from the 512×512 image whereas the last one is fed with a random crop from the down-sampled version of the training image

[25] With classical data analysis techniques such as Principal Component Analysis (PCA) or Independent Component Analysis.

(down to 256 × 256). This work is extended by Bianco et al. [2019]. They propose to add a multi-tasks learning and a training on Region Of Interest (ROI) proposal to replace a random crop selection. Only this last element improves the performance. van Noord et al. [2017] propose to use a multi-resolution CNN method to deal with variation in image resolutions and scales. They propose to train an ensemble of CNNs working at four different scales[26]. This ensemble of CNNs outperforms any single scale model because artist attribution may require very fine grain information. Nevertheless, those methods are not specific to art analysis.

2.3.3.2 Neural Style Transfer for Art Analysis

To face the lack of data, several teams tried to propose effective data augmentation scheme for art analysis.

Some teams [Florea et al. 2017; Badea et al. 2018; Madhu et al. 2019; Sarı et al. 2019] tried to use a sophisticated method for data augmentation or domain adaptation of artistic datasets using the neural style transfer algorithm [Gatys et al. 2015a] presented later in Section 2.5.2.2, or its variants [Johnson et al. 2016; Huang et al. 2017b]. First, Florea et al. [2017] and Badea et al. [2018] show that a domain adaptation by transferring style on natural images with the neural style transfer method from [Gatys et al. 2015a] do not bring a big performance advantage comparing to using the raw natural image and cause a huge overhead. Then, several works show that neural style transfer may be an effective solution to the domain adaptation task. Datasets with images that have been transformed by neural style transfer allow to have better performance than using the raw natural image datasets. Smirnov et al. [2019] apply this method for object recognition in paintings, Gupta et al. [2018] for identity recognition in painted portraits and Sarı et al. [2019][27] for sex recognition with an off-the-shelf CNN features extractor. Madhu et al. [2019] show that first fine-tuning the CNN on a gender classification tasks with images that have been transformed by a style transfer algorithm[28] leads to a better character recognition at the end.

Thomas et al. [2018] train a model for cross-domain objects classification on both natural and style transformed[29] images with an additional style invariant loss. This loss predicts the image modality / style. During backpropagation, the opposite of its gradient is used to enforce the lower layers to learn style-invariant features. This use of synthetic artistic images outperforms all other domain adaptation techniques on average. In [Inoue et al. 2018], an efficient pipeline is proposed to train a detector on new artistic modalities in a semi-supervised manner. This approach requires natural images with bounding box annotation of those classes and involves a relatively costly style transfer procedure[27]. In particular, this method only allows the detection of object classes that are present and have been annotated in natural images. In contrast with such methods from the domain adaptation fields, we will propose a weakly supervised approach that allows the detection of new classes, in Chapter 5.

Using neural style transfer for data augmentation may even be beneficial for not art related tasks [Perez et al. 2017; Geirhos et al. 2019; Jackson et al. 2019; Cicalese

[26] From 256 to 2048 pixels.
[27] They use the CycleGAN model [Zhu et al. 2017a].
[28] They use the Huang et al.'s method [2017b] .
[29] With [Johnson et al. 2016] or [Huang et al. 2017b] methods.

et al. 2020; Karatzoglidi et al. 2020].

Simpler solutions than neural style transfer can be considered depending on the target set. For instance, Yin et al. [2016] propose to use synthetic "drawing-like" versions of natural images by taking the difference of the grayscale images and their Gaussian blur, followed by a contrast normalization. This "drawing-like" training set is used to train the CNN and leads to better performance than using raw natural images.

2.3.3.3 Architecture Adaptation

Some works tried to propose methods that are specific to art analysis to improve the performance.

On the one hand, some works try to adapt the architecture of the CNN for artistic style classification [Chu et al. 2016b; Mao et al. 2017; Chu et al. 2018b; Chen et al. 2019]. Huang et al. [2017a] improve the accuracy by adding a gray-level co-occurrence matrices as input to the network in order to describe the brush stroke information. Most works have been inspired by the use of Gram matrices for neural style transfer [Gatys et al. 2015a] to represent the artistic style or at least the texture of images. Mao et al. [2017] propose to add to their model a branch that considers the Gram matrices of one of the network layers. Their two branches version of the VGG16 provides better performance. Chu et al. [2016b] evaluate the efficiency of different correlations of the off-the-shelf deep features for the style or artist classification tasks. They show that Gram matrices of the features are better than the raw features and that the dot product of the Gram matrices with the cosine similarity of the features is even better. In addition, the correlations across multiple layers do not provide better performances. Wei et al. [2017] made the same conclusion for Japanese Ukiyo-e instead of Western paintings. In a following work, [Chu et al. 2018b] propose to learn the correlation by replacing the SVM classifier with a CNN on top of the off-the-shelf deep features. This solution leads to higher performances. Moreover, using the Gram matrices of the features instead of the raw features provides a gain of performance. Chen et al. [2019] go a step further by using Gram matrices of several layers and by fine-tuning the whole network. They even improve the performance by learning an adaptive version of the Gram matrices by learning how to weight the Gram matrices terms.

On the other hand, some works try to add external information for improving the classification performances. Garcia et al. [2019] show that multi-tasks learning or context-aware embedding may improve art classification. The last solution consists, in addition to training the main CNN for the art task, to train a CNN encoder to output an embedding vector containing information about non-visual artistic knowledge. This knowledge is represented by a graph of information whereas the encoder input is the deep features of the image.

2.3.3.4 Understanding CNNs Trained on Art Images

With the success of CNNs, there is a corresponding need to be able to explain their decisions and understand their internal representations. Some studies looked at what CNNs learned on artistic dataset or how they had been modified to be more adapted to the new art domain.

Yin et al. [2016] show that the low-level filters discard color information during fine-tuning for adapting to an almost black and white drawings training dataset. Elgammal et al. [2018] show that AlexNet fine-tuned on a style classification task will discard oriented-edge-like filters except for a horizontal one. Usually, the first convolutional layers are easy to interpret as they contain primitive features like edges, lines and color transitions, but the higher-level filters are much harder to interpret. Two tools are used to interpret these layers: feature visualization and attribution, as mentioned in Section 2.2.

van Noord et al. [2015] use an attribution method based on occlusion [Zeiler et al. 2014] to determine artist-characteristic patches. Surprisingly, the texture of the material on which an artwork is created is used to determine Rembrandt's authorship instead of the print lines. By looking at the channel responses, Tan et al. [2016] conclude that lower layers learn simple patterns and higher ones, complex objects parts such as the circular shape for portrait class. Strezoski et al. [2017b] look at the feature visualizations and attributions of a small CNN trained on artistic dataset. Some of the characteristic patterns of the classes can be found in the visualizations (as a circle for the portrait class too). The attributions show that high-level layers learn more semantically relevant features than lower-level layers (e.g. facial features as mustache against contour). In a similar way, Wilber et al. [2017] use a Generative Adversarial Network (GAN) based visualization technique for synthesizing images maximizing a score for emotion or media classification tasks. Once again parts of objects are recognizable and can be associated with the emotion labels (for instance teeth for scary or landscape for peaceful). Sabatelli et al. [2018] also observe the attributions for two models (one pretrained on ImageNet and the other on the Rijkmuseum art dataset) to investigate which pixels of each input image contribute the most to the final classification predictions. The model fine-tuned on the Rijkmuseum dataset focuses more on small details (with the lower layers of the network) to perform artist attribution compared to the ImageNet dataset. Then, van Noord et al. [2017] observe in the attribution images a shift to finer details when moving to higher resolutions in their ensemble of CNNs fine-tuned at four scales for author classification. Finally, Offert [2018] look at the feature visualizations of the classes portrait and landscape from a fine-tuned InceptionV3. Faces and drapery seem to be the defining features of a portrait according to this CNN.

2.3.4 Off-the-shelf Applications

Recently, CNN models became so efficient and easily usable that several research teams used them off-the-shelf without any training for large scale data analysis. Brachmann et al. [2017] use different type of variant of CNN features to understand the specific visual properties of artworks in comparison to natural images, in the continuation of the Graham et al.'s work [2007]. They conclude that on their given datasets, artworks tend to be filled with structure over the entire image on the contrary to natural images. Second, art images show more variability of the features across the image than the natural patterns (plants or vegetation) but less than human-made scene (objects or urban scene). Moreover the statistics of art images seem different from the ImageNet ones. [Cetinic et al. 2019a] use CNN models trained on natural images for predicting subjective assessments of fine art images as aesthetics, memorability and sentiment scores. They also try to study how these scores are correlated to different artistic

styles and genres, as well as how they correlate with high-level visuals attributed from other CNN models. The study suggests some conclusions such as the fact that abstract styles tend to be more memorable, but have a lower aesthetic and positive sentiment score. Paintings that include motifs related to nature (landscape, sea, animals) tend to have a higher aesthetic score, etc. Jenicek et al. [2019] show that an off-the-shelf pose estimation network with a robust spatial verification[30] outperforms methods based on VGG features for content-based image retrieval for figurative western artworks. Madhu et al. [2020] use a pretrained pose estimation network with image processing techniques without any training to estimate characters poses and gazes, foreground and background separation. This allows to have a simple representation of the underlying structure and composition within an image.

2.3.5 Discussion about Deep Learning for Art Analysis

As mentioned before these CNN models have been used for analysis artworks images in relation to art history.

Now, we mention some of the attempts about art history (and more precisely paintings history) discoveries thanks to these modern data science methods. Elgammal et al. [2018] try to find the main components in the deep features of painting images. They also try to correlate these dimensions with time or with Wölfflin's visual principles [1915]. These principles are old and basic descriptions of paintings. For instance, two of these principles are "linear vs painterly" and "multiplicity vs unity". The results may indicate that the CNN learn a smooth transition between styles without explicit time information during training. By looking at the activation manifold with diverse data analysis tools, the authors claim to have discovered quantitative connections in art history, such as the Cézanne role as a bridge between Impressionism and Cubism-Abstract art, but these are relatively coarse observations. Next, Cetinic et al. [2019b] train a CNN to rate artworks according to the previously mentioned five Wölfflin's principles. Sarı et al.' pipeline [2019] recognizes the color of the clothing depending on the sex of the characters but it does not enable testing hypotheses such as the Laqueur's transition from one to two sex model. Castellano et al. [2020a] only recover the main temporal periods from art history, using an unsupervised clustering model. Huckle et al. [2020] use unsupervised style embedding to study a set of contemporary young artists. They find no connections between visual styles and the artists social proximity, gender or nationality. Other works have unclear purposes. For instance, prediction of aesthetic, sentiment and memorability scores by Cetinic et al. [2019a] is particularly questionable due to subjectivity but also to the biases of the pretrained model and the art dataset used.

For the moment, CNN applications to art analysis mainly manipulate color, shape, texture, or patterns to analysis influences or uniqueness of an artwork. This is somehow an old fashion way to make art history, a "connoisseurial art history"[31], from 19th-century art historians. The elements described within the images or the labels used such as a coarse grain artistic style [Tan et al. 2016; Lecoutre et al. 2017] or the Wölfflin's principles [Cetinic et al. 2019b] are limited. As mentioned by [Spratt 2017], even the

[30] Based on RANSAC.

[31] As mentioned by Pollock [2014] about computer vision models applied to art analysis by Saleh et al. [2014].

most advanced computer vision model is mostly operating on what Panofsky termed the "pre-iconographical" level: a factual description of the objects or scene within the image [1939]. The second step of Panofsky is "iconographical" or conventional level, i.e. an understanding of the actions or facts presented. For instance, understanding the religious scene or characters. This is why it is necessary not to learn visual categories that can not be found in photography. In this regard, we propose a weakly supervised approach to learn new categories in various domain in Chapter 5. The last level of Panofsky's system is the "iconology" or intrinsic one, which consists in explaining the historical, religious or philosophical influences that the artists condensed into his work. The difficulty of such level of interpretation for a machine is pointed out by Lang et al. [2018].

The current models are purely visually based and neglect artwork position in a broader historical and social context. Indeed, they are connecting traditional formal analysis of art with computer vision and machine learning methods, more than claiming new art history conclusions. Nevertheless, these models allow to only focus on the visual content and thus helps to build a decontextualization of images. Thus, the visual link retrieval models [di Lenardo et al. 2016; Seguin et al. 2016] are an excellent illustration of the possibility of only using images in the same spirit as the concept of pattern migration, developed by Warburg, back to the 1920s, in his *Mnemosyne Atlas* project [2012].

Moreover, they are the first and needed step to more complex models that could be able to take into account social and historical context [Garcia et al. 2019]. At the same time, criticism of the manipulated concepts, used corpora or research hypothesis is critical for art history [Bishop 2018]. It starts to be addressed by digital art historians and computer vision researchers. Finally, several of the previously mentioned research teams developed online search engines [Artrendex 2018; Crowley et al. 2018; Seguin 2018] dedicated to fine arts for different visual recognition tasks for helping art historians to browse online art databases or realizing large scale analysis, such as the distance viewing methodology [Bender 2015]. The reader may also refer to the recent survey about machine learning for cultural heritage by Fiorucci et al. [2020].

2.4 Multiple Instance Learning and Weakly Supervised Object Detection

Machine learning has achieved great success in various tasks, particularly in supervised learning tasks such as classification and regression. Most successful techniques require ground-truth labels (for instance the class to which the training example belongs) to be given for a big training dataset. However in many tasks, it can be impossible or difficult to obtain strong supervision information due to the high cost of the data-labeling process. For instance, it requires around one minute to draw a bounding box around a common object [Su et al. 2016] or 1 hour and half to segment an urban scene image [Cordts et al. 2016]. Moreover the micro-task platforms as Amazon Mechanical Turk create social exploitation and excessive precariousness [Tubaro et al. 2019]. Thus, it is desirable for machine-learning techniques to be able to work with weak supervision that meaning to be able to extract information that are not explicitly provided in the

training set. *Weakly supervised learning* is a generic term that refers a variety of studies that attempt to construct predictive models based on weak supervision. We define weakly supervision as the setup where only coarse-grained labels are provided to the algorithm or when a label is provided to a group of elements/instances. For instance, only image-level labels are provided whereas the task consists in producing object-level labels or pixel segmentation of the image. In Figure 2.6, one can see some different examples of weakly supervised tasks from computer vision field.

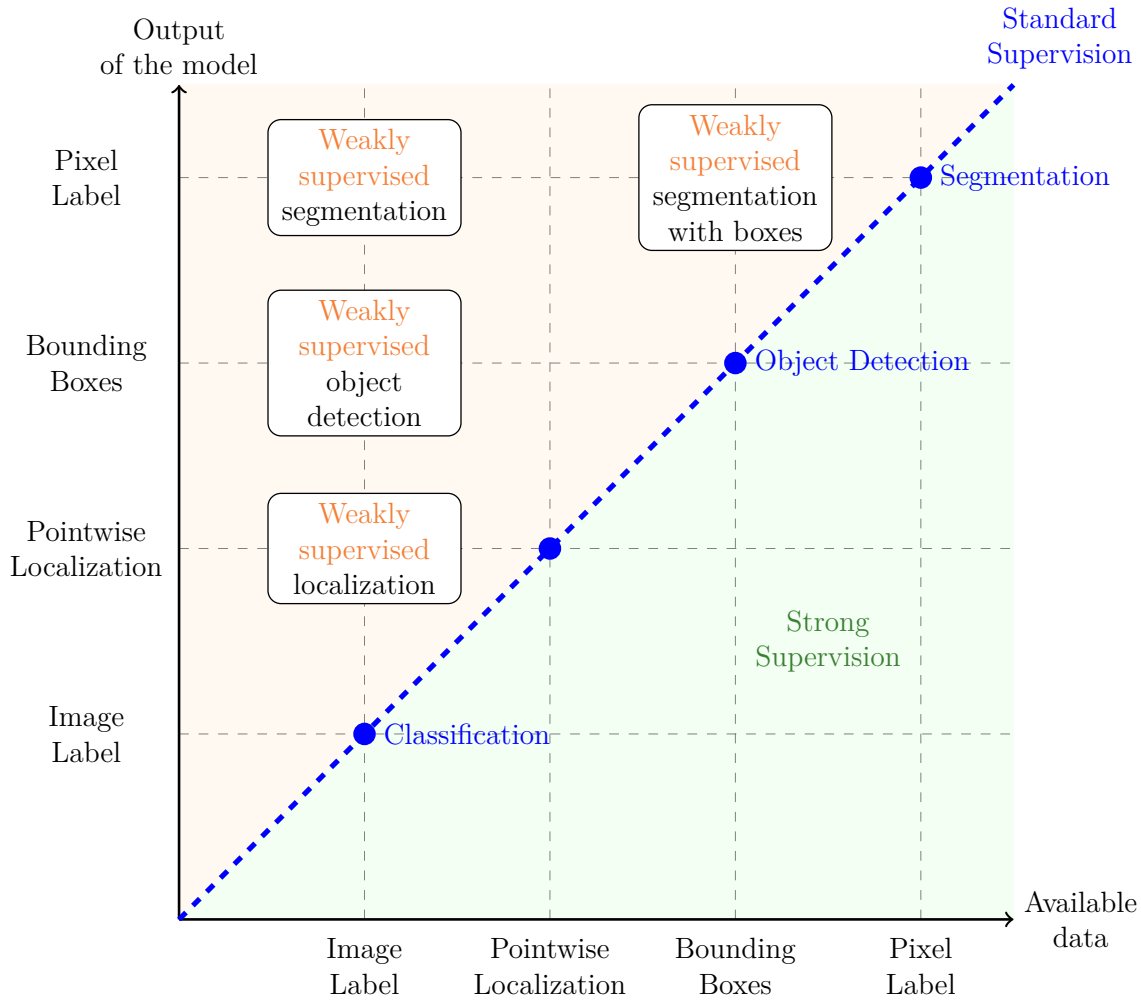


Figure 2.6: Boundaries between strong and weak supervision for some tasks in computer vision. Inspired by Bilen [2018].

We will not take into account the *semi-supervision* (or incomplete supervision) defined as the setup in which only a subset of training data is labeled while the rest remains unlabeled, as in [Zhu 2005]. Neither will we consider the inaccurate or noisy supervision, i.e. the given labels are not always exact. This last paradigm mainly used in the webly supervised setup [Chen et al. 2015; Del Chiaro et al. 2019a]. The weakly supervised task we want to solve is the *Weakly Supervised Object Detection* (WSOD) task. It refers to the task of learning an object detector using limited annotations, usually image-level ones. Often, a set of detection (e.g. bounding boxes) is considered at image level, of which we only know if it contains the category of interest. The

most popular statistical paradigm corresponding to this problem is referred to as *Multiple Instance Learning* (MIL) [Dietterich et al. 1997] that is why we will first develop this framework in Section 2.4.1 before describing state-of-the-art WSOD methods in Section 2.4.2.

2.4.1 Multiple Instance Learning Paradigm

We first give some basic notations related to MIL. Let $\mathcal{B} = \{B_1, B_2, \dots B_N\}$ a set composed of N bags. Each bag B_i corresponds to a positive or negative label $Y_i = \{-1, 1\}$ and contains K_i feature vectors: $\{X_{i,1}, X_{i,2}, \dots X_{i,K_i}\}$ where $X_{i,k} \in \mathbb{R}^d$. A set of elements is usually called a bag of instances. On the contrary to the supervised setup, we only have a label at the bag level as in Figure 2.7. Each of these feature

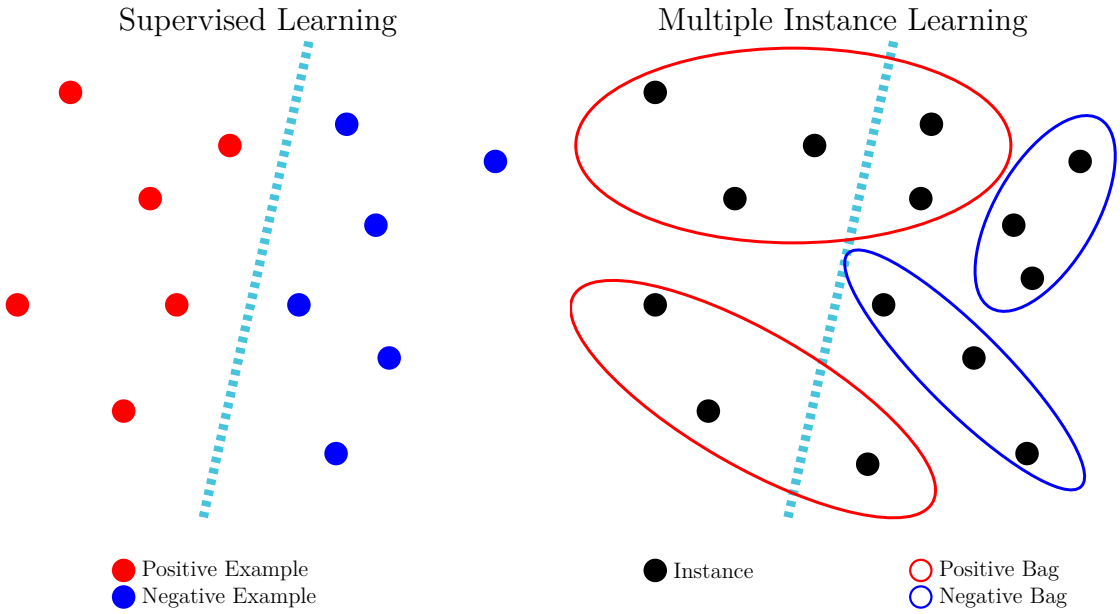


Figure 2.7: Supervised learning vs MIL : in supervised learning all the examples are labeled whereas in MIL only the bags are labeled, i.e. the instance labels are unknown. The blue dotted line shows the separator learned by the classifier.

vectors (instances) corresponds to a positive or negative label. Instance labels are unknown in positive bags, but are assumed all negative in negative bags. Indeed, a common hypothesis (called the MIL assumption) is that a bag is labeled positive if at least one instance contained in the bag is labeled positive:

$$Y_i = \begin{cases} +1 & \text{if } \exists\ k \in \{1, \dots, K_i\} : y_{i,k} = +1 \\ -1 & \text{if } \forall k \in \{1, \dots, K_i\} : y_{i,k} = -1 \end{cases}, \tag{2.7}$$

where $y_{i,k}$ denotes the instance label. Thus the problem is asymmetric between the two classes. One important information about the bags composition is the proportion of positive instances in positive bags called Witness Rate (WR). When the WR is very high, positive bags contain only a few negative instances.

As mentioned before in Section 2.1.1.1, in a typical supervised problem, the goal is to learn the best prediction function f_ω minimizing the empirical risk over the training set (equation (2.2)). However, in the MIL setup, we only have bag-level information during training, whereas we are looking for an efficient instance-level prediction. Observe that, in the case of WSOD, we are interested in the instance-level classification task (in order to be able to classify each vector and hence each region of the image) and not in the bag-level one, which is also classical in MIL applications (for drug detection for instance). As noted in [Vanwinckelen et al. 2016], the accuracy of algorithms can be very different between these two different tasks. There are two main ways to tackle the fact that we only have bag level ground truth information. First, one can aggregate all the predictions of one bag to a single prediction (at bag level) during training. Hence, we can write:

$$\hat{y}_i = g\left(\{\hat{y}_{i,k}\}_{k\in\{1\dots K_i\}}\right), \tag{2.8}$$

with g an aggregation function over the elements of a bag i. Then, the bag level loss function can be written as:

$$L(Y_i, \hat{y}_i) = l\left(Y_i, g\left(\{\hat{y}_{i,k}\}_{k\in\{1\dots K_i\}}\right)\right), \tag{2.9}$$

with l a classical binary loss function.

The best way to aggregate instance-level predictions in order to find a classifier separating each of the individual vectors $X_{i,k}$ of each bag at test time is still an open problem. Some research works use for g the max operator [Zhou et al. 2002], the average or the Log-Sum-Exponential (LSE) [Ramon et al. 2000]. Indeed, since the training is done with only bag-level information, at test time the learned classifier must be able to handle each instance almost independently from the others.

Second, one can consider each instance of a bag individually (as in the mi-SVM case, see Figure 2.8) and the loss function can be written as

$$L(Y_i, \{\hat{y}_{i,k}\}_{k\in\{1\dots K_i\}}) = g\left(l\left(h_{i,k}(y_i), \{\hat{y}_{i,k}\}_{k\in\{1\dots K_i\}}\right)\right), \tag{2.10}$$

where g is an aggregation function (usually an average), l a loss function and $h_{i,k}$ an assignation function of the label associated to the instance (i,k). $h_{i,k}$ is usually named a latent label [Felzenszwalb et al. 2010] or pseudo-label. Indeed, the true label of the instance (i,k) is unknown in positive bags. If we consider that the label of a bag is equal to the label of its instances $h_{i,k}$ is the identity, otherwise it is a function from $\{-1,1\}$ to $\{-1,1\}$ depending on the bag and the instance.

We will now present some methods proposed to deal with this generic MIL problem.

2.4.1.1 SVM based Solution

A well-known solution to this problem through a generalization of SVM [Cortes et al. 1995] has been proposed in [Andrews et al. 2003]. The idea behind the SVM-based approach is to learn a hyperplan to separate positive and negative instances and perform bag label prediction through an aggregation function.

Actually, two flavors are considered in [Andrews et al. 2003]: the Maximum Pattern Margin SVM (mi-SVM) and the Maximum Bag Margin SVM (MI-SVM). In the case of mi-SVM, each element of positive bags is assigned a label and the SVM margin is

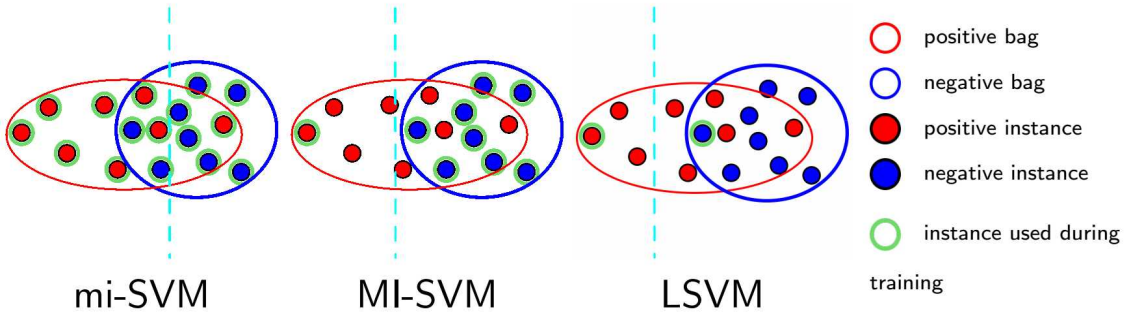


Figure 2.8: Comparison of standard SVM based MIL models. The blue dotted lines show hypothetical hyperplanes learned by the models, and the green circles show the instances used during the SVM training.

imposed at the instance level. In the case of MI-SVM, the SVM margin is imposed the most positive element of each positive bag and to the least negative element of each negative bag.

Several heuristics to solve the non-convex problem posed by the MIL have been proposed. For example, Gehler et al. [2007] introduce a new objective function that tries to estimate the quantity of positive examples in a positive bag, before using deterministic annealing to optimize it. In contrast to the MI-SVM method, the algorithm can consider several elements as positive in the positive bag. In [Joulin et al. 2012], the authors propose a convex relaxation of the softmax loss. A comprehensive review of SVM based MIL methods can be found in [Doran et al. 2014]. From this review it appears that mi-SVM and MI-SVM are still competitive on the tasks studied there. The mi-SVM algorithm is not vulnerable to multimodal distributions thanks to the use of a kernel. It also seems to be robust to low WR, thanks to of the individual instances labels, according to [Carbonneau et al. 2016b].

In [Felzenszwalb et al. 2008] a reformulation of MI-SVM is presented and called Latent SVM (LSVM). The LSVM model is defined in [Felzenszwalb et al. 2008] before realizing the relationship to MI-SVM in [Felzenszwalb et al. 2010]. In the LSVM only the instance with the maximum output latent variable is used to represent its bag, whereas all negative instances are used for MI-SVM. Figure 2.8 summarizes the instances on which the SVM margins are imposed in the most popular SVM based MIL methods. It should be noted that in these works [Felzenszwalb et al. 2008; Felzenszwalb et al. 2010], a bag of instances represents the set of parts of an object and the MIL formulation is used to train an object detector with a fully supervised training. The aggregation is made over the part of an object and can be directly compared to the ground truth label on the contrary to the WSOD method we develop later (Section 2.4.2). The LSVM method has been the subject of much improvements, for instance by tracking negative evidence [Durand et al. 2019], adding a symmetric prior to the problem [Durand et al. 2018] or adaptation for particular cases, for example for structured output [Yu et al. 2009].

2.4.1.2 Other Ways to Tackle the MIL Problem

Neural Network based Methods Another approach to the MIL problem is to use neural networks whose architecture treats each instance symmetrically, before an

explicit aggregation (max, average) is performed. From this point a classical neural network performs a classification task [Ramon et al. 2000; Zhou et al. 2002].

In [Wang et al. 2018b], a new multiple instance neural network to learn bag representations is presented. They incorporate recent improvements in deep learning (such as DropOut, ReLU, deep supervision and residual connections) to get better results. These works focus on the bag classification task but they all, by design, provide an instance classification network: it is sufficient to feed the network with a bag consisting of one item.

Work on the features space Some of the methods consist in focusing on selecting the most pertinent features rather on the optimization problem. The MIL-Boost algorithm by Viola et al. [2005] is the same as gradient boosting except the loss function is based on bag classification error. They adapt the feature selection criterion to optimize the performance of the model. This method is applied to object detection in images. Zhang et al. [2002] use the Expectation Maximization algorithm to search for the maximum of the Diverse Density (DD) measure. The DD measure is high in neighborhoods containing many instances of different positive bags and few from negative ones. The DD-based methods work under the assumption that the positive data form compact clusters in the features space. From a recent survey on MIL by Carbonneau et al. [2016a], it appears that the most efficient algorithm for an instance-level classification seems to be a clever variation of bagging and multiple classifiers to deal with multimodal distributions. This method named RSIS [Carbonneau et al. 2016b], probabilistically identifies the positive instance in positive bags using a procedure based on random subspacing (random selection of the dimension of the features and the training samples) and clustering (k-means). This score indicates the likelihood that an instance is positive. The score is high if there is a majority of instances from positive bags in the cluster and few negative instances. This score is computed over all the different random subspaces. Given those scores an instance selection probability distribution is obtained for each bag. This leads to get different training subsets and a classifier (an SVM) is trained on each of the subsets. MI-SVM and RSIS are similar in the sense they rely on the identification of the most positive instances in each bag. This strategy seems successful to some degree, but is prone to ignore more ambiguous positive instances that are dominated by others in the same bag, according to the instance-level classification benchmarks from [Carbonneau et al. 2016a].

2.4.2 Weakly Supervised Object Detection

The goal of this section is to provide a literature review of the Weakly Supervised Object Detection (WSOD) task. WSOD consists in training object detectors with only image-level label supervision [Nguyen et al. 2009]. We first present the characteristics of the WSOD problem defined as a MIL one (Section 2.4.2.1). We then introduce the main directions for improving WSOD models (Section 2.4.2.2). Next, we present two main families of approaches based on CNN for tackling this problem. The first one is based on extracting localization information directly from the deep feature maps (Section 2.4.2.3) whereas the other one is based on region proposals (Section 2.4.2.4). In Section 2.4.2.5, we give an overview of the solution of the WSOD based on transfer learning WSOD before concluding in Section 2.4.2.6.

The standard MIL pipeline for WSOD is the following one:

1 Determine a windows space: extract a certain number of regions of interest from the image
2 Feature extraction: compute a feature vector per region (handcrafted or CNN based)
3 Classification: classify each feature vector of each region with a MIL formulation of the problem
  3.1 Re-localization: updating the labels of the instances/regions in the bags
  3.2 Re-training: training the model with the updated labels

Most of the time, once the windows space is determined, the methods focus on the classification step by alternating between re-localization and re-training of the model. The recent methods tend to do these three main steps at the same time, in an end-to-end manner. For instance, some CNN-based methods regroup feature extraction and classification [Bilen et al. 2016; Kantorov et al. 2016; Diba et al. 2017; Tang et al. 2017a] whereas others regroup the three [Zhu et al. 2017b]. A lot of work has been done to propose efficient models learned with weak supervision on a task of detection but this is still an open problem. We will do a brief summary of the research in this field in the following.

2.4.2.1 Characteristics of WSOD seen as a MIL Problem

First, we note specific characteristics of the object localization or detection tasks in computer vision seen as a MIL problem. These characteristics have been defined by Carbonneau et al. [2016a] in the previously mentioned survey. One of the parameters that can most influence the performance of the MIL algorithms is the bag composition. In the WSOD problem the Witness Rate is usually low because, it is standard to extract an important number of box candidates from a given image (between 1k and 100k).

Relations between Instances Most MIL methods assume that positive and negative instances are sampled independently from a positive and a negative distribution. However, in the WSOD case, the independent and identically distributed assumption is violated because structure and correlation exist between the instances and bags. According to Carbonneau et al. [2016a], there are three types of relations: intra-bag similarities, instance co-occurrences and structure.

Intra-bag similarities The instances belonging to the same bag may share similarities that instances from other bags do not share. For example, some methods use densely extracted patches which overlap (Figure 2.9) since they share a certain number of pixels, they are likely to be correlated. Also, the background of a picture could be split in different segments which can be very similar (blue patches in Figure 2.10). To face this difficulty, Kantorov et al. proposed to take into account some context around the ROI [Kantorov et al. 2016].

Instance Co-occurrence Instances co-occur in bags when they share a semantic relation. This type of correlation happens when the subject of a picture is more likely to be seen in some environment than in another, or when some objects are often found

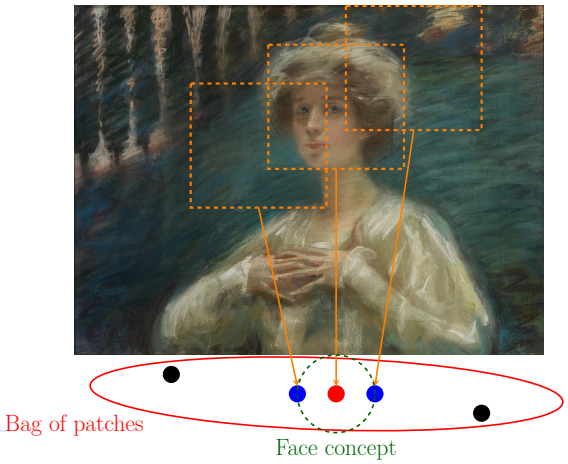


Figure 2.9: Illustration of intra-bag similarity between instances: The patches are overlapping, and thus, share similarities with each other. Figure based on Carbonneau et al. [2016a].

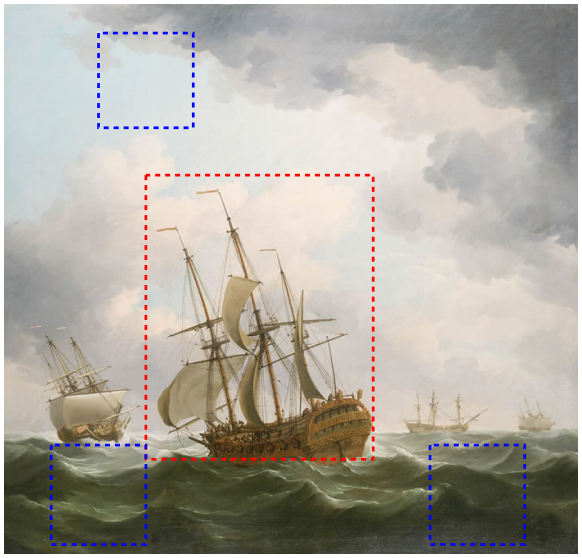


Figure 2.10: Example of co-occurrence and similarity between instances: three patches contain sea and sky and are therefore very similar. Moreover, since this is an image of boats, the background is more likely to be the ocean than a crop field. Figure based on Carbonneau et al. [2016a].

together. For example, the boats of Figure 2.10 are more likely to be found on the sea than in crop filed. Thus, the observation of water patches might help to decide if the image contains rather a boat than a farm tractor. This kind of negative evidence has been exploited by Durand et al. [2016] in their WELDON model and variants. The co-occurrence of instances can be useful for instance classification problems but they may confuse MIL models. If a given positive instance often co-occurs with a given negative instance, the algorithm may consider the negative instance as positive, which in this context would lead to a higher false positive rate.

They may also explain why sometimes, the model detects bigger objects than expected (or a part of the background as belonging to the object). This phenomenon

have been mentioned before, in Section 2.2.1.1.

Instance and Bag Structure In the case of extracted regions from an image, there is a spatial relationship between the region. For instance, the sky is always above the characters within the image. Graph models were proposed to better capture the relations between the different instances of an image [Zhu et al. 2017b].

Data Distributions Many methods make implicit assumptions on the shape of the distributions, or on how well the negative distribution is represented by the training set.

Multimodal Distributions of Positive Instances Some MIL algorithms work under the assumption that the positive instances are located in a single cluster or region of the feature space. In object detection, the target concept may correspond to many clusters. Their appearance also changes depending on the point of view for example. It is unlikely that a compact location in feature space encompasses all of these variations. Instance-level SVM-based methods like mi-SVM [Andrews et al. 2003] can deal with disjoint regions of positive instances using a kernel. In [Carbonneau et al. 2016b] instances are grouped in clusters and the composition of the clusters are analyzed to compute the probability that instances are positive. The multimodal distributions can also be dealt with the use of the k nearest neighbors for ranking instances [Siva et al. 2011], the multimaps learning [Durand et al. 2017] or the multiple clusters learning (from Proposal Cluster Learning (PCL) [Tang et al. 2018a]).

Non-Representative Negative Distribution In some applications as person detection, the training data cannot entirely represent the negative instance distribution. For instance, provided sufficient training data, it is reasonable to expect that an algorithm learns a meaningful representation that captures the visual concept of a human person. However, it is almost impossible to entirely model the negative class distribution (all that's not a human). In contrast, in some applications like cancer detection in histology, the negative (healthy) cells compose the negative class may be modeled using a finite number of samples as those tissues possess a limited appearance. Most of the methods are based on one-class SVM whereas some methods model only the positive class as in [Dietterich et al. 1997].

Label Ambiguity Label ambiguity is inherent to weak supervision. One the one hand, an object can be composed of really discriminating parts (for instance the face for a person) and this part of object can be sufficient to classify this instance as a positive one. On the other hand, instance labels might not have clear semantic. Which part of an image depicting an abandoned place is ruins? This may lead to noisy or not well-defined labels.

2.4.2.2 Main Research Directions to Improve WSOD Methods

There are several ways to improve a WSOD model by working on the different parts of the decomposition of the problem mentioned at the beginning of Section 2.4.2.

Windows space The box proposals can be naively produced by sliding windows which provide dense and numerous proposals (> 100k per image), based on different translations, scale and aspect ratio. This leads to a very low WR which leads to a higher false negative rate and limits the algorithm performance. That is why most works try to limit the number of region proposals or even select the most pertinent ones as soon as possible in the whole pipeline. The possible size of the windows can be constrained thanks to some apriori on the object size [Nguyen et al. 2009].

The regions can be proposed by an unsupervised object proposal algorithm as Selective Search [Uijlings et al. 2013] or EdgeBoxes [Zitnick et al. 2014] mainly used in WSOD. These methods provide typically 2k box proposals per image, selecting regions with strong textures or containing most of the energy of the gradient of the image. A supervised classification network can also be used as a proxy to determine which regions of the images are the most pertinent. In [Li et al. 2016], regions are collected by a mask-out classification strategy to select the best positive regions in each image with a classification proxy score and then fine-tune a detector with those propositions as "ground truth" bounding boxes.

The bounding boxes can also be extracted a posteriori after the full training of the model. They are obtained with a thresholding of the internal activation (feature maps) of the fine-tuned network in the Soft Proposal Network (SPN) model [Zhu et al. 2017b].

Finally, in our case (see Section 5.3.2.1), we use the boxes generated by the Region Proposal Network (RPN) from a pretrained Faster R-CNN [Ren et al. 2015], it is a sparse space of around 300 regions per image.

Initialization of the classification step Some works focus on the initialization of the classification phase (step 3 in the standard pipeline defined at the beginning of Section 2.4.2). This initialization is crucial due to the non-convexity of the problem. This concerns the labels associated to the instances from positive bags. The boxes labeled as positives, at the initialization, can be the whole image or the whole image minus a margin [Bilen et al. 2014] or a selection of "good" regions as in [Song et al. 2014]. In this latter work, the author construct a graph of initial positive boxes built on three criteria. The positive boxes must be relevant (occur in many positive images), discriminative (dissimilar to boxes from negative images) and complementary (captures multiple mode). Then they proposed to learn a smooth version of an SVM [Joulin et al. 2012] on the features from R-CNN [Girshick et al. 2014].

Re-localisation and Re-training Most of the time, the WSOD models use a max margin formulation (as in MI-SVM mentioned before) for the Re-localisation thus the model only keep one positive instance per positive image.

We can highlight two main differences with the supervised learning of an object detector. First, we consider only one positive instance in each positive image whereas in the supervised case we can consider several instances. Indeed, several instances of the same visual category may be present in the image. Second, the true negative instances from the positive images are completely ignored in the weakly supervised setup. In most works about fully supervised object detectors [Girshick 2015; Ren et al. 2015], a

mining of hard negative examples is used to train the detector on tricky regions of the positive images (for instance the part of the objects).

In the seminal work [Nguyen et al. 2009], they propose to alternate between optimizing the SVM objective and finding the instances within the images that maximize the SVM scores. This second step is done with a non exhaustive search thanks to a branch-and-bound algorithm to handle the very large number of regions. Other works try to prevent the model drift with the iterative re-localisation step by stopping the learning when the detector starts to drift away from the initial localization of the objects of interest [Siva et al. 2011].

Furthermore, some works propose more robust optimization by using a different aggregation function g, for instance by relaxing the max operator by a softmax [Bilen et al. 2014]. In [Jie et al. 2017] it is proposed to replace the max with a more sophisticated technique that considers spatial neighbors with a dense sub-graph discovery: the regions of the graph are considered as connected if they have an Intersection over Union (IoU) > 0.5. In [Ilse et al. 2018], they propose a two-layers neural network (or MLP) to learn the MIL pooling operator. It includes a gated attention mechanism (i.e. an element-wise multiplication) between two branches of the neural network. Yan et al. [2018] propose a dynamic pooling scheme to learn the instance-to-bag relationship based on the routing model from Capsule Network [Sabour et al. 2017]. The idea is to iteratively update the instance contributions to its bag embedding during each feed-forward step. More variants of the max operator are also mentioned in Section 2.4.2.3. On the other hand, some works try to incorporate the fact that an image can contain multiple instances of the same class as in [Ren et al. 2020].

Adding priors yields a better proposal selection Another direction is to add prior to the model in order to get better proposals selection or refinements. Simple priors can be added as a mutual exclusion between classes in the case of multiple class detection but it is usually not possible because WSOD is often a multiple labels setting (some regions may belong to several labels). Some papers are interested in quantifying how likely a window contains an object of any class by defining an objectness score for each window. It can be defined with edge straddling for instance [Zitnick et al. 2014]. In [Bilen et al. 2016], a feature map extracted from the spatial pyramid pooling is multiplied to its objectness score via a scaling layer as in equation 2.11. This objectness score can also be used in an additive way as in equation 2.12 as it is done in [Siva et al. 2011; Tang et al. 2017c]. These methods tend to push towards the whole object instead of small regions.

$$\underset{k}{\text{argmax}} \;\; f_w(X_k) * Obj(k), \tag{2.11}$$

$$\underset{k}{\text{argmax}} \;\; \lambda f_w(X_k) + (1-\lambda) Obj(k), \tag{2.12}$$

with $\lambda \in [0, 1]$.

Change the way the optimization/learning process is done In [Cinbis et al. 2014], it is proposed to train on all folds but k for the positive examples and all the negative examples to reduce the overfitting. The relocalisation are done on the left apart positive images. Some papers are inspired from curriculum learning [Bengio et

al. 2009]. In [Kumar et al. 2010], they select positive samples via an inter category competition and reduction of the size of the windows over time.

Improve the feature representation The most effective improvement in WSOD has been made by using better feature representation of the images. We can see 3 different kinds of feature representations:

1 Classical feature representation as BoW obtained by SIFT descriptors [Nguyen et al. 2009] or Fisher vectors [Cinbis et al. 2014].
2 Off-the-shelf features from pretrained CNN [Bilen et al. 2014; Song et al. 2014; Teh et al. 2016][32].
1-2 Both classical representations and CNN extracted ones [Wang et al. 2014; Cinbis et al. 2016]
3 End-to-end CNN trained [Bilen et al. 2016]

As mentioned before, fine-tuned CNNs provide state-of-the-art results in object detection [Girshick 2015] and CNNs are able to learn objects or object parts localization during a classification training [Oquab et al. 2015]. That is why, CNNs become the best solution for feature learning for WSOD.

We will now detail the CNN-based methods for WSOD in the two following Sections 2.4.2.3 and 2.4.2.4. These methods can be split into two groups: the ones using the feature maps for inferring object localization and the other ones using region proposal algorithms. The first set of methods provides good pointwise localization whereas, the second set provides the best performances for the WSOD. The superiority of the second set of methods can be due to the relatively small size of feature maps inside the CNN compared to the size of the input image.

2.4.2.3 Methods based on Deep Feature Maps

First, Simonyan et al. [2014] show that is possible to extract relatively good bounding boxes from the class Saliency maps of CNN trained on ImageNet classification task. As mentioned before in Section 2.2.1, a saliency map is obtained by computing the derivative of the class output with respect to the input image.

The seminal work of Oquab et al. [2015] pointed out that a CNN trained on a classification task is able to predict approximate localization of the object, directly from the features maps of the CNN, without explicit saliency map computing. CNNs seem to automatically focus on the most discriminative part of objects. To deal with this problem, two main research directions have been studied: improving the aggregation of instances used in the bag-level classification loss with more complex pooling function and improving the re-localization and re-training steps with cascade models.

Spatial Pooling One of the key issues is to find how to pool the spatial dimension of a deep feature map (or the regions) to obtain a score per class. This is the aggregation function mentioned before in Section 2.4.1. The most popular approach is the max

[32] In the first case, a regularize version [Joulin et al. 2012] of LSVM [Felzenszwalb et al. 2010] is used on R-CNN features whereas in the second case, it is a Structural LSVM [Yu et al. 2009] trained on DeCAF features [Donahue et al. 2014].

pooling [Oquab et al. 2015], which selects the best region to perform prediction. In the case of binary classification, this pooling is an instantiation of the MIL paradigm. A limitation of the max pooling is related to its sensitivity to noise in the region scores, because it only uses the most discriminative region. To increase robustness, a possible approach is to use several regions.

Zhou et al. [2016] use a Global Average Pooling (GAP) to pool the feature maps to a vector prediction. They map back the predicted class score to the previous convolutional layer to obtain a Class Activation Mapping (CAM) capable of highlighting the disciminative regions of a category. The models trained with max pooling tend to underestimate object sizes, while those trained with GAP overestimate them. In [Sun et al. 2016c], a LSE pooling is proposed as a trade-off between max and average pooling. In [Teh et al. 2016], an attention net is trained to compute an attention score from deep features[33] for each pre-computed box proposal.

Durand et al. propose in a string of three papers a new pooling (the models are called MANTRA, WELDON and WILDCAT [2015; 2016; 2017]). The main idea is to take into account positive and negative regions to "not focus only on discriminative part". The score is equal to the mean of the k^+ top scoring instances plus α times the mean of the k^- instances with the lowest score. This formula generalizes the different poolings mentioned previously. The next step is to generalize pooling function by replacing the max and min operators by their LSE approximation [Durand et al. 2019].

The WILDCAT model has the particularity to learn a set of M features maps per class during the training[34] before the WILDCAT pooling in order to train the whole model end-to-end through a classification loss. In [Courtiol et al. 2018], the authors proposed to add a MLP at the end of the WELDON pooling to get better classification results. None of these methods [Durand et al. 2015; Durand et al. 2016; Durand et al. 2017; Courtiol et al. 2018] are tested on a bounding boxes evaluation although these models output good weakly supervised pointwise localization of object[35]. Nevertheless, in the case of high-resolution images it can allow to efficiently localize important tiles, for instance for histology [Pirovano et al. 2020].

In [Zhu et al. 2017b], they tried to tackle the WSOD task specifically with the idea of using the feature maps to generate bounding boxes. A region proposal generator is trained using weak supervision. The feature maps are transformed into a graph then into an objectness score map. This objectness score weights the feature maps that are subsequently fed to a classification layer. The bounding boxes are obtained by thresholding the objectness score map and extracting the tightest box covering the foreground pixels. This model is named SPN.

Cascade architecture Some methods use cascade architecture to refine the predicted regions [Sun et al. 2016c; Diba et al. 2017]. The first stage (localization network) proposes a set of promising boxes that are likely to contain objects. The second stage (classification network) classifies the proposed regions. ProNet [Sun et al. 2016c]

[33] The model is used as a feature extractor.

[34] Those features maps are averaging which comes down to only having one feature map learned per class.

[35] Weakly supervised pointwise localization is a specific machine learning task less studied than WSOD but addressed by some works as [Grzeszick et al. 2018].

uses a cascade architecture to zoom into those promising boxes, as in Overfeat [Sermanet et al. 2013a], and then train new classifiers to verify them. The localization and classification networks are independent and trained iteratively. This two-stage architecture can be repeated several times to progressively zoom into objects. [Zhang et al. 2018b] propose to improve the localization proposed by CAM [Zhou et al. 2016] with a network architecture including two parallel-classifiers. The first branch localizes some discriminative object regions learned by classification and obtained with CAM. Then, the maximal value pixels of the feature maps are erased from them. These modified feature maps are used to train the second classifier. This classifier learns a new and complementary object region.

2.4.2.4 Methods based on Region Proposals

Firstly, in [Bilen et al. 2016], the Weakly Supervised Deep Detection Network (WSDDN) model is proposed? It is a two streams network: one stream for classification (which assigns each region to a class) and one for detection (which picks the most promising region in a image given a class). This second stream assumes the object is present in the region with the most salient structure. It is not a standard MIL problem, it can be seen as a mini-batch MIL [Bilen 2018] because only small batches of images are provided to the model. This model is based on unsupervised object proposals such as EdgeBoxes or Selective Search and uses a pre-trained network on ImageNet in a pure transfer learning manner. It is an adaptation of the Fast R-CNN [Girshick 2015] model to the weakly supervised setup. This seminal work has been improved by several teams. Some works focus on the loss function used to train the network whereas others propose cascaded networks to refine the proposals.

Improving the loss function Kantorov et al. [2016] propose two types of context-aware guidance models: an additive one and a contrastive one. One the one hand, the additive model uses contextual information to encourage the network to select ROI semantically compatible with the context (if the surrounding is a horse, the ROI can be a person). On the other hand, the contrastive model focuses on the absence of target-specific features (of the class of interest) in its surrounding context. It helps to separate the object and the background. The additive model consists in adding the class activation of the ROI and the one from the context (larger crops with the ROI cropped) whereas the contrastive model is based on subtraction. The additive model experimentally prevents expansion of detection beyond object boundaries whereas the contrastive model prevents contraction of detection to small object parts.

An entropy minimization is added to the two branches model in [Wan et al. 2018]. The goal is to control the variance of positive instances in order to remove some ambiguity on the detections. The proposals are spatially separated into cliques where spatial distributions and objects probabilities are jointly modeled. The first branch minimizes a global entropy over the whole image in addition to the classification loss. The second branch focuses on the object localization and minimizes the local entropy on cliques of proposals with pseudo-labels. Ge et al. [2018] propose to split the boxes proposals in m groups to get m sub-images. Then they fine-tune a CNN for multi-label classification by taking the maximum over the sub-images. Finally, this CNN is used as a feature extractor to fine-tune a final part for detection. They incorporate the

idea of Durand et al. [2015] to take positive and negative regions to not focus only on discriminative parts in the loss function.

In another direction, Zhang et al. [2018c] develop a criterion named mean Energy Accumulation Scores (mEAS) to rank localization difficulty of a positive image. This criterion is used to learn the WSDDN [Bilen et al. 2016] progressively by feeding examples with increasing difficulty. They also introduce a novel masking regularization strategy over the high-level convolutional feature maps to avoid overfitting initial samples.

Cascaded Networks To improve ProNet, [Diba et al. 2017] propose a two stages end-to-end architecture. The first stage is a CNN with a GAP as in [Zhou et al. 2016]. The class activation maps of this first stage are used to select the best candidate boxes among EdgeBoxes' box proposals. Then, they use a ROI pooling to feed a smaller net with the corresponding deep features of those boxes. This second stage is trained with a MIL loss (max over the proposals boxes and a softmax over the classes). To train end-to-end the cascaded network they use an aggregated loss function of the two networks. The authors also propose a three stages architecture where they add a segmentation network in the middle of the former model. The middle segmentation network takes as input the middle level feature maps and use the class activation map as a pseudo-ground truth to learn a binary segmentation per class. This model provides better feature maps to select the candidate bounding boxes.

A dual network is trained with a self-paced curriculum learning process for solving the WSOD in [Dong et al. 2017]. Each network not only selects more training data to update its weights but also accepts reliable training data from the other network. The model consists of the following two networks:

1 The first one named "Positive Instance Selective Network", is a ContextLocNet [Kantorov et al. 2016] based on Fast RCNN [Girshick 2015] without box regression, it will minimize the classification error given a bounding box
2 The "Region Proposal Refinement network" is based on a region fully-convolutional network [Dai et al. 2016] and maximizes the IoU between the pseudo bounding box and a pseudo-ground truth

The algorithm iteratively generates training subset with pseudo bounding boxes, and re-trains the two networks. To generate the new training pool of the images with pseudo labels, each image is labeled by the networks and the wrong prediction are pruned (image with the wrong label compared to the image-level label, image with too many bounding boxes, image with bounding boxes containing bounding box with higher confidence score).

The WSDDN [Bilen et al. 2016] have also been improved by Tang et al. [2017a] under the name of Online Instance Classifier Refinement (OICR). The idea is to use multiple streams. Each stream is supervised by the latter to create a multi-stage classifier refinement. The method is inspired by mi-SVM since they assign a pseudo-label to each of the regions, but instead of using an alternative strategy to relabel the instances and retrain the full model, they use an online refinement algorithm. Moreover, they selected instances according to the spatial relation (spatial overlapping regions) and not only the score predicted by the classifier. The top-scoring proposal is set to 1 for the positive image and then the overlapping regions get the same label. On

the contrary to the mi-SVM algorithm where the label is given by the former classifier. This allows to learn to detect larger parts of objects gradually (refinement step by step) and not only the discriminative part. By doing that they add a prior to their model to force the next classifier to provide larger boxes than the previous ones. The authors note that finding a correct initialization can be difficult and a correct weighting of the losses is necessary to avoid unstable solutions.

The same team improves their OICR model [Tang et al. 2017a] with the PCL one [Tang et al. 2018a]. Instead of choosing the proposal with the maximum scoring for the relabeling, they determine what they call "proposal cluster centers". For each image, they use k-means clustering to divide top-ranking proposals into clusters. They choose proposals within each cluster which has the highest score to select the top-ranking proposals. Then, they group the remaining proposals into different groups using the overlapping graph, where each group is associated with a cluster center or corresponds to the background. These pseudo labels are used to supervise the next network. One of the most important contributions is that each cluster is treated as a small new bag to train refined instance classifiers without giving a strong label at each instance of the bag (like a sub MIL bag classification problem). The initialization of this algorithm and the number of clusters seem to be important.

The same team also try in [Tang et al. 2018b] to train the RPN of Faster R-CNN with only image-level annotations. It is an excellent idea because the RPN seems to be the most powerful element of the network. The method is based on several steps to refine the boxes proposal. The first set of proposals consists in an exhaustive search of sliding windows on the input image. Then, they compute an edge-like response on the features maps at different layers in the network (like the low-level statistics used in EdgeBox [Zitnick et al. 2014]). This objectness scores are used to filter the set of proposals. It is refined by a WSOD model from Tang et al. [2017a] trained to classify background or objects. This network is named "Proposal Refinement" network. They alternate between training the "Proposal Refinement" network and a WSOD network that uses these proposals. This is a promising method but it doesn't work for the moment.

In [Zhang et al. 2018d], it is proposed to refine the proposal from OICR [Tang et al. 2017a]. First, they select the bounding boxes that cover most of the high score discriminative regions returned by OICR. Then, they fine-tune a Faster R-CNN on those region proposals (considered as pseudo-ground truth). Finally, they use the RPN predictions to refine again the pseudo-ground truth proposals. In [Wan et al. 2018], the model can also be improved by training multiple detectors stacked one after another in a "Accumulated recurrent learning" manner to have a more robust pipeline. Finally, the authors of [Ren et al. 2020] also add some strong priors about the WSOD task to improve WSDDN. Indeed, they add a "Multiple instance self training" block to the model. It is designed to pick a set of high-scoring non-overlapping regions[36] as pseudo-labels. These pseudo-labels are used to train a cascade of bounding box regressors. Additionally the model is trained in an adversarial manner. An auxiliary model is trained to sparsely dropout some pixels of the deep feature maps. It drops out the discriminative parts and helps the training. They also propose a sequential batch forward and backward computation to handle one of the major bottlenecks in

[36] In some sense, it is a prior about the WR of the problem.

WSOD the memory limitations due to the big number of boxes. It is one of the most promising works for the WSOD task.

Other works not based on WSDDN In [Li et al. 2016], a two steps strategy is proposed, first collecting good regions by a mask out classification, then selecting the best positive region in each image by a MIL formulation and then fully fine-tune a detector with those propositions as ground truth. This work is built on [Song et al. 2014].

In [Arun et al. 2019], the authors propose to train two collaborative networks one of them being a Conditional Network with noisy extra-channel. The goal is to jointly minimize the dissimilarity between the prediction distribution and the conditional distribution.

2.4.2.5 WSOD with Knowledge Transfer

Another way to improve the accuracy of a machine learning model is to utilize a source dataset and transfer knowledge to the target domain through semi-supervision, homogeneous or heterogeneous transfer learning. Some research articles talk of the WSOD-with-transfer task.

On the one hand, some works transform classifier to an object detector. Hoffman et al. [2014] transform a classifier to a detector by adding to the classifier weights the mean of the weights of the nearest neighbor source target classes (in terms of distance between deep features). Tang et al. [2017b] incorporate external knowledge about object similarities from visual and semantic domains with a word embedding model. Then, they use this knowledge to adapt an image classifier to an object detector for a weakly labeled category.

On the other hand, other works try to obtain a general object detector from the source dataset. This knowledge can be the objectness predictor [Deselaers et al. 2012], the object proposals [Uijlings et al. 2018; Zhong et al. 2020] or a universal bounding box regressor [Lee et al. 2019].

In [Zhong et al. 2020], they use an iterative transfer process. They alternate between instancing the pseudo labels to the box proposals on the target and source sets and training of their deep model (an hybrid the WSDDN and the Faster R-CNN models). The class-generic objetness score from the deep detector, trained on the source domain, is used in several ways to help the learning process. It is also used during training, by adding a MSE loss between the objectness score of each box and the maximum of the classification score over classes. They pick the most confident predictions for pseudo-labeling with a score obtained as a linear interpolation between the objectness score and the classification score.

In [Uijlings et al. 2018] they train a universal detector[37] on the source dataset and uses the detection results from this detector as proposals for the MIL problem on the target dataset. They compare the impact of different objectness score used during the re-localization step. The class-generic objectness score is trained on the set

[37] Based on SSD [Liu et al. 2016b].

union over the 63k training samples of all 100 source classes in the dataset they use (a subset of ILSVRC 2013 validation set). They also study the use of the closest semantic source class or the closest ancestors in the ImageNet hierarchy information during the knowledge transfer. The class-generic objectness provides the best performance.

In Chapter 5, we propose a method for learning an object detector with weak supervision on non-photographic datasets, possibly including new classes. Our approach belongs to the WSOD-with-transfer paradigm as we transfer a CNN that have been trained on a natural image dataset with bounding box information.

2.4.2.6 Conclusion on WSOD

The domain is very active and the state-of-the-art changes regularly. The frontiers of this field are fuzzy and changing but the main goal is still to learn models able to infer object localization without such information during the training time. In brief, weakly supervised object detection is challenging due to:

- Ambiguity with parts and context
- Ambiguity with discriminative part and object of interest
- Sensitive to initialization
- Prone to overfitting

Some of the solutions are:

- Improving the aggregation function
- Robust re-localisation and re-training method
- Incorporating prior knowledge or knowledge from a different modality

2.5 Texture Synthesis With Convolutional Neural Networks

In this section, we will present the main principles and milestones about texture synthesis by example (Section 2.5.1). Then in Section 2.5.2, we will focus on the method based on CNNs , the most related to the contributions of this thesis.

2.5.1 Texture Synthesis by Example

2.5.1.1 Texture Definition

There is no formal and universal mathematical definition of texture in the field of image processing, due to the diversity of cases covered by this term. At a given scale, a texture has the same aspect whatever the observed area. It is a visually coherent and "homogeneous" area, possessing a kind of spatial repetition of the same pattern in different directions in space. Thus, a photograph of a material (bricks, marbles ...), skins, textiles, turbulence, sponges, furs, pebbles or turbulence constitutes an image of texture as can be seen in Figure 2.11. Of course, this is a non-exhaustive list of examples.

Two types of textures can be coarsely distinguished:

Figure 2.11: Examples of some texture images (from https://www.textures.com/).

- the regular textures whose periodicity of the pattern is obvious: walls, fabrics, grids, etc.
- the stochastic textures whose pixel values seem randomly scattered over the image: noise, sand, grass, crowds, etc.

The first family of textures can be described by frequency approaches or regular tiling, while the second is modeled by random process. Each type of texture has its own visual characteristics. It is therefore difficult to describe them all with a unique model. Obviously, the majority of textures belong to the continuum between regular and stochastic textures.

2.5.1.2 Principle of Texture Synthesis by Example

Exemplar-based texture synthesis aims to generate new but perceptually similar images from a given exemplar. These samples are required to be visually faithful to the example and as diverse as possible.

An example of a possible reference image accompanied by images that one would ideally like to synthesize can be seen in Figure 2.12. There are several possible solutions to the texture synthesis problem which makes it inherently ill-posedness. Nevertheless, this problem has been a fruitful way to test visually the validity of various mathematical models for more than forty years, ranging from time series [McCormick et al. 1974], Markov Random Fields [Cross et al. 1983] to wavelet decomposition [Heeger et al. 1995; Portilla et al. 2000] or non-parametric Markovian modeling [Efros et al. 1999]. More recently, CNNs have permitted impressive progress in the field, initiated by the work of [Gatys et al. 2015b], itself followed by numerous contributions, e.g. [Liu et al. 2016a; Ulyanov et al. 2016; Snelgrove 2017; De Bortoli et al. 2019; Heitz et al. 2020]. However, the best evaluation of the results is still a perceptual evaluation based on some implicit quality metric such as the absence of copies of large parts of the reference image or the pleasantness of the synthesis.

Most texture synthesis methods can be classified into two categories, *patch-based* methods and *statistics-based* methods. The patch-based methods usually generate

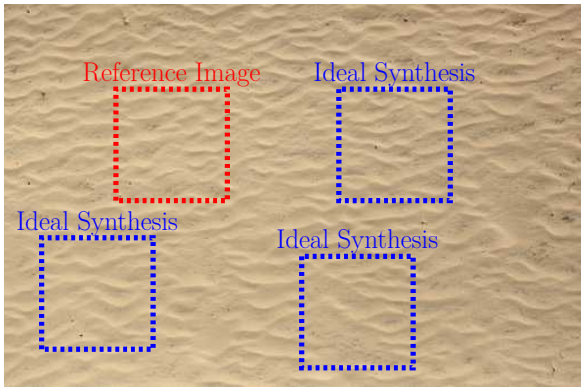


Figure 2.12: Example of a reference texture accompanied by ideal synthesized ones.

new textures by iteratively sampling patches from a given texture sample [Efros et al. 1999; Wei et al. 2000; Lefebvre et al. 2005]. They are able to generate very realistic texture examples with a high speed, even on highly structured textures. However, the generated texture examples have limited varieties and capacities of innovation. On the contrary, the statistics-based methods impose a set of statistics computed on the image or a transformed version of it, such as wavelets [Heeger et al. 1995; Portilla et al. 2000]. The statistics-based methods can generate perceptually good samples but are highly dependent on the chosen set of statistics. The statistics-based methods reach their limits on highly structured textures for which the method based on deep learning statistics are proposed in [Gatys et al. 2015b]. This approach uses the Gram matrices of feature maps obtained with a pre-trained VGG19 [Simonyan et al. 2015].

2.5.1.3 Main Milestones

We will now present some of the main works on texture synthesis.

Texture description The statistical description of textures was introduced by Béla Julesz for the analysis of the visual perception of textures. He conjectured that it was possible to discriminate textures according to their N-th order statistics so that textures within a group would be indistinguishable by human perception [Julesz 1962]. He proposed to represent textures by a Markov process. By statistics of order N, he means statistics related to the values of N pixels of the image[38]. In [Julesz et al. 1973], it is illustrated that some pairs of grayscale textures sharing the same second-order statistics are indistinguishable. However, there are textures with identical second (and even third) order statistics which are visually distinct [Julesz et al. 1978]. Numerous works have sought to determine the minimum N that allows classifying textures. Then, the considered statistics are no longer being computed only on the image but also on transformed versions of it (filtering, etc.). The underlying idea is to determine the minimum set of statistics that allow describing textures.

First models McCormick et al. [1974] propose an autoregression model for texture synthesis. This model exploits the linear dependence between pixels. Given a random

[38] In the mentioned paper, the considered statistics are estimators of the marginal distributions of N pixels.

initialization the next pixel is synthesized by taking a linear combination of the previously synthesized pixels plus a linear combination of the previous random values. The coefficients of these linear combinations are the parameters of the model. They are estimated on the reference image thanks to the time series seasonal analysis method.

For a long time, the methods considered as the state-of-the-art were methods based on Markov Random Fields [Cross et al. 1983]. The texture image is considered as a two-dimensional random Markov field. The core of these methods is to estimate the probability law of a pixel intensity knowing the texture to which it belongs. These methods consider that this law only depends on the values of a small number of the pixel's neighbors. Thus, the law followed by a pixel knowing its neighbors is chosen *a priori* and its parameters are determined from a sampling of the reference image using e.g. the maximum likelihood estimator. Having a formal model allows a theoretical analysis of the method's convergence. This method allows describing and synthesizing near-stochastic textures but fails to generate regular ones.

Filters Random Fields and Maximum Entropy (FRAME) is a Markov Random Field model constructed from the empirical marginal distributions of filter responses based on the maximum entropy principle [Zhu et al. 1998]. To sample a new texture synthesis from this probability distribution the Gibbs sampler is used. Zhu et al. [2000] develop the Julesz ensembles texture model based on a common set of statistics. They prove that this model is equivalent to FRAME in the limit of an infinite image [Wu et al. 2000].

Statistics-based methods Between 1995 and 2000, the Markov Random Field methods are supplanted by the multi-scale wavelet ones that extend the Julesz approach based on multi-scale statistics. Heeger et al. [1995] propose to characterize a texture by the first-order statistics of its colors and by its responses to different multi-scale and multi-orientation filters realized using a steerable pyramid wavelet decomposition. In a nutshell, this approach consists in computing a set of statistics from the reference image I and then performing an optimization starting from a noise image[39] by forcing the statistics of the synthesis to be almost equal to the reference image statistics. One alternate between computing the statistics on the synthesis $\tilde{I}$ and enforcing the reference statistics to it (by histogram matching in this work). There are no theoretical results of convergence. This method can be extended to the color image case by using decorrelated decompositions, for instance. This method works well in the case of quite stochastic textures. However, it fails to synthesize quasi-periodic textures, mosaics or images with several main directions. This is maybe due to the only use of the marginal responses to the filters and not the correlations between pixels, scales or orientations. For example, contours are not reproduced because they correspond to inter-scale relationships.

Portilla et al. [2000] greatly improve the previous method by enforcing higher order statistics of the filter responses from the wavelet decomposition. In the same way as before, the algorithm performs a wavelet decomposition of the reference image and the synthesized image.Then, statistical constraints are iteratively imposed on these

[39]This construction can be made from a white noise but it is not an obligation, it is only an implementation tip to maximize image entropy (without warranty).

wavelet projections. The statistics used are the first 4 moments[40], the autocorrelation of the projections and the inter-correlation between projections of the same scale and of different scales[41]. The synthesis is performed using a gradient descent, there is still no evidence of convergence. The use of these statistics allows the synthesis of much more complex textures.

Patch-based methods Then, a new type of methods which produce satisfactory visual results, is appearing on the scene but these methods based on patches have some drawbacks, we will describe later. Efros et al. [1999] propose to synthesize a new texture image from a reference one by sequentially adding pixels to the synthesis. For each new pixel, the algorithm picks up the pixel in the reference image whose neighborhood is close enough to it. To do this, the authors use a Gaussian-weighted Euclidean distance weighted by the distance between the pixels. A tolerance on the allowed proximity and the number of neighbors considered are the main parameters of the algorithm, and can have a strong influence on the output result. To compensate for the slowness of this algorithm, a faster version has been proposed by Efros et al. [2001]. This method works patch by patch instead of pixel by pixel, allowing a certain overlap between the synthesis and the new patch added to it .

The patch-based methods are numerous. For instance, Wei et al. [2000] propose to use a coarse-to-fine generation process, where the coarser level of the multiresolution pyramid, which has been already synthesized, is considered to guide the nearest neighbor finding. To accelerate it even more, they also propose a tree-structured vector quantization. To improve the quality of the synthesis, Kwatra et al. [2005] propose another kind of multiresolution strategy. At a given scale, the algorithm uses as initialization, an up-sampled version of the synthesis obtained at a lower scale, with the exception of the coarse scale. This method is combined with using multiple neighborhood sizes to capture patterns of different sizes. In the same spirit, Lefebvre et al.'s method [2005] is based on a multiple resolution strategy, but it manipulates pixel coordinates rather than colors. At each scale, it introduces randomness by jittering the coordinates and then correct the synthesis with a few iterations of neighborhood matching.

All these methods can be considered as non-parametric Markovian models and provide a procedure for introducing randomness into a reference texture without changing its perceptual properties. They make it possible to synthesize textures of excellent quality even in the case of a periodic image. Nevertheless, two majors drawbacks of these methods must be pointed out. First of all, the synthesized texture can include whole areas of copy-paste from the reference image as illustrated in [Aguerrebere et al. 2013; Raad et al. 2017a]. Second, the algorithm can reproduce many times a singular pattern of the reference image and neglecting the rest of it.

Micro-texture model An interesting method based on Julesz et al.'s idea to completely impose second-order statistics of a texture [1973] has been developed by Galerne et al. [2011]. The method consists in synthesizing new textures by enforcing a random phase on the Fourier transform of the reference image. To respect the colors, the same

[40] Mean, variance, skewness and kurtosis.

[41] In the latter case, an oversampling is carried out in order to calculate the inter-correlation.

random phase must be imposed to each channel of a given pixel. This phase randomization algorithm allows to properly synthesize stochastic textures, textures with tiny structures. However, it does not correctly synthesize structured elements and geometrical textures with large period patterns.

Alternatives to wavelets Following the milestone works relying on statistical constraints based on wavelet decompositions [Heeger et al. 1995; Portilla et al. 2000], three alternatives to wavelets were proposed to impose statistics to the synthesis:

- Sparse dictionary
- Convolutional Neural Network
- Scattering transforms

First, Tartavel et al. [2015] propose to add some constraints to the use of an adaptive sparse dictionary for texture synthesis [Peyré 2009]. The first constraint is related to the color histogram of the image, the second one to the spectrum of the image and the last one to the distribution of the dictionary's atoms use. The term on the spectrum allows to control the global structure of the synthesis whereas the constraint on the dictionary decomposition allows keep important geometrical elements of the reference image such as corners. The optimization is made in an alternate descent. This method avoids the copy-paste phenomenon that can be encountered with patch methods. This method proposes a multi-resolution strategy too. The synthesis at a lower scale is used as initialization for the next one. We point that the use of an adaptive dictionary allows to only impose statistics of order 1.

On the other hand, Gatys et al. [2015b] propose to do texture synthesis with a CNN trained for image classification task. This approach consists in minimizing the distance between the covariance matrices of the channel activation between the reference image and the synthesis (as detailed in Section 2.5.2.1). This distance can be computed for several layers in a similar way to the multiple scale of a wavelet decomposition. The optimization is made for all the layers at the same time.

Finally, Mallat [2016] propose to use second order coefficients of a scattering transform of the reference image as statistics. The scattering transforms of an image are the modules of the cascading wavelet transforms of the image. It is a model halfway between wavelet transforms (used by Portilla et al. [2000]) and CNNs. Contrary to the use of CNNs, there is no recombination of the different responses of the previous layer to obtain the next layer.

2.5.2 Texture Synthesis With Convolutional Neural Networks

The synthesis method based on CNN representation of the images [Gatys et al. 2015b] became since 2015, the state-of-the-art for texture synthesis. We will present in this section, this method and some of the improvements made to it.

2.5.2.1 Gatys et al.'s Method [2015b]

Directly inspired from the seminal work of Portilla et al. [2000], Gatys et al. [2015b] propose to constrain the statistics of a pretrained CNN feature maps. The statistics are provided by correlations between the feature maps corresponding to different filters.

The synthesis is done by minimizing the Frobenius norm between the Gram matrices of the feature maps[42] of the reference images and the synthesis at different layers. This minimization is done by gradient descent via backpropagation in the network, in a way similar to the feature inversion [Mahendran et al. 2015] mentioned in Section 2.2.1.2. One can observe in Figure 2.13, the different steps of the synthesis of an image during the gradient descent. The starting point is a very low amplitude white Gaussian noise (other kind of noise can be used). The first iterations are the most important as they structure the image. Thousand of iterations are needed to reach visual convergence but there is no guarantee of true convergence. The value of the cost function (equation (3.1)) is specified below each image. This algorithm leads to a local minimum which depends on the initialization point. More details of this method will be provided in Section 3.2.1.

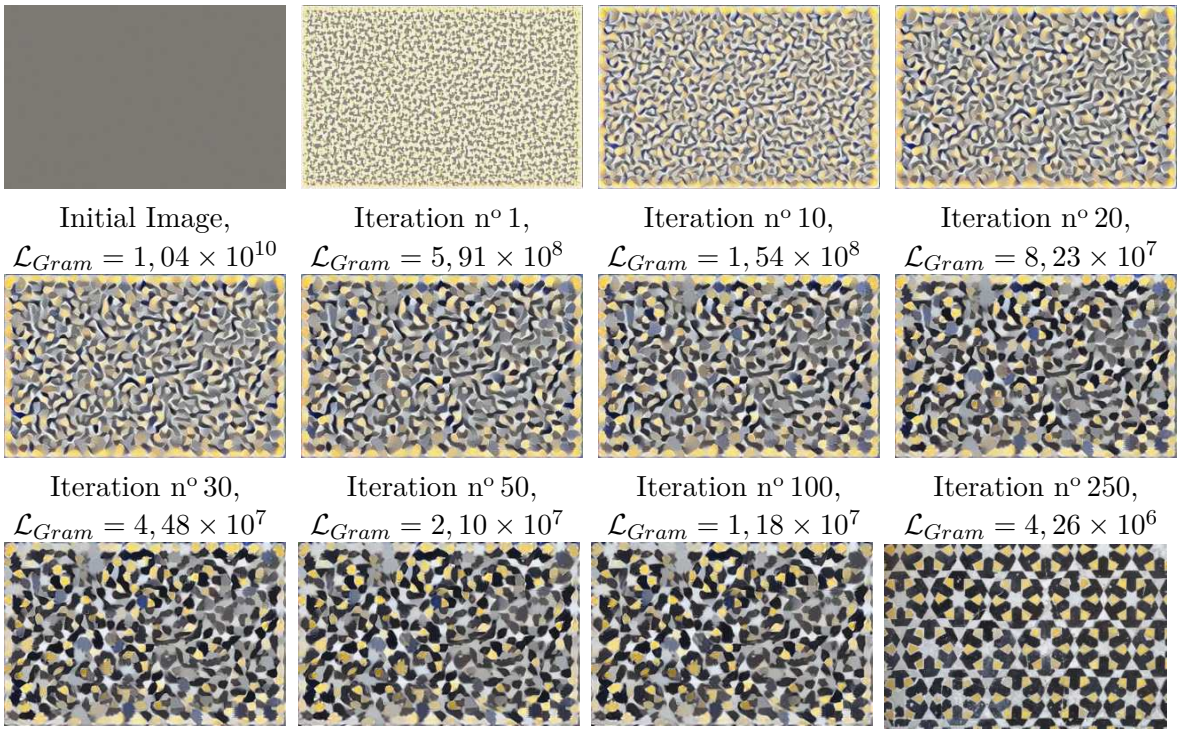


Figure 2.13: Evolution of the synthesis during the optimization. Layers used: {conv1_1,pool1,pool2,pool3,pool4}.

Increasing the numbers of layers of the CNN on which the texture representation are matched increase the complexity of the synthesis. By considering only the first layer of the network (second line on the Figure 2.14) we obtain a texture of very low granularity, with very local structures. By considering deeper layers of the network, we can see that structures of larger scales appear. The set of layers: {conv1_1,pool1,pool2,pool3,pool4} is the default case in the following work (see Figure 2.4b or Table D.3 in the Annex for the names of the layers).

[42] Their covariances.

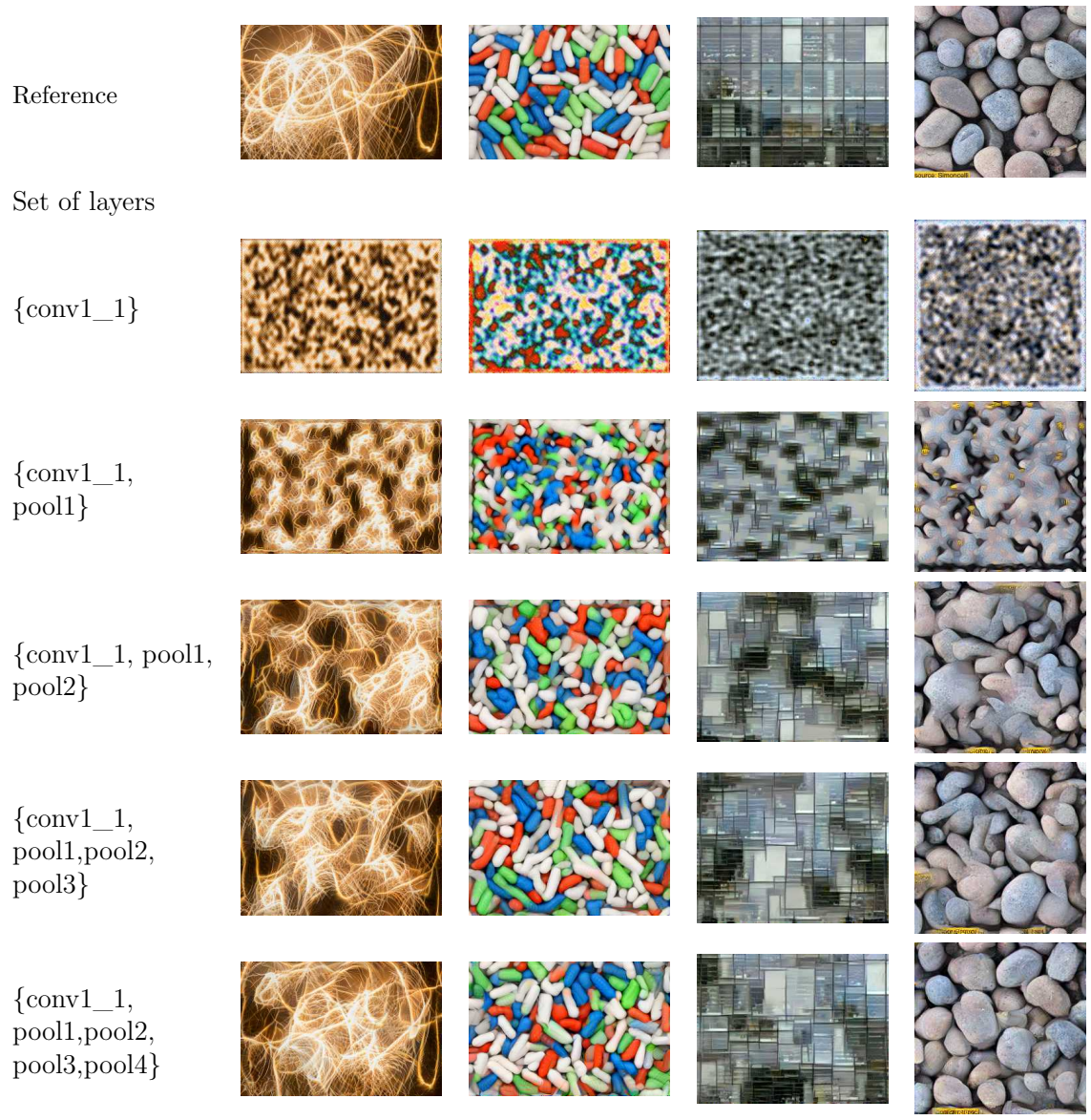


Figure 2.14: Texture Synthesis obtained after 2000 iterations with [Gatys et al. 2015b] method for different sets of layers. See the Figure 2.4b for the names of the layers.

The main difference with the Portilla et al.'s method [2000] is the use of CNN responses instead of wavelets ones, the use of only one kind of statistics (the Gram matrices), the absence of explicit inter-layers[43] statistics and the fact that the optimization is not alternated.

Nevertheless, the most important contribution is the transfer of a VGG19[44] pre-trained on the ImageNet object classification task [Simonyan et al. 2014] to compute high level image representations. More precisely, as specified in [Gatys et al. 2015b], the weights of this network are normalized on a certain set of images to be able to make the responses to the filters comparable, perhaps to be able to aggregate the terms of the loss function coming from different layers in a more consistent way. However, even

[43] They can be considered as scales.

[44] Gatys et al. [2015b] replace the max pooling layers by average pooling ones [2015b].

using the original VGG19 trained weights [Simonyan et al. 2014], one can obtain equivalent results as shown in Figure 2.15. Using the same architecture with random weights leads to experimentally poor results, as it can be seen in the last column of Figure 2.15. The trained weights are able to capture high-level visual structure as mentioned in Section 2.2. Ustyuzhaninov et al. [2016] propose to synthesize textures by imposing the Gram matrices of the responses of the image to random filters of different size[45]. This can be seen as a single layer CNN with a random initialization. The synthesis is clearly below these obtained with a pretrained VGG. This is certainly due to the absence of hierarchical representation of the image. He et al. [2016] propose a midway solution by selecting layer by layer, random weights that provides a good reconstruction of the input image. The results are perceptually degraded with this pseudo-random network compared to the trained one.

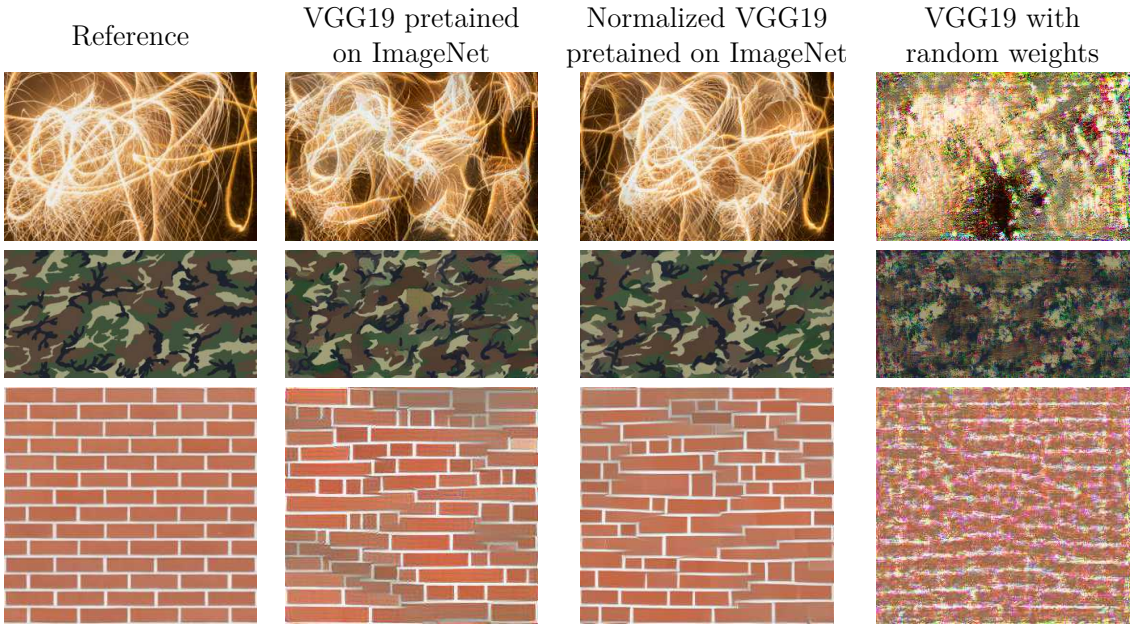
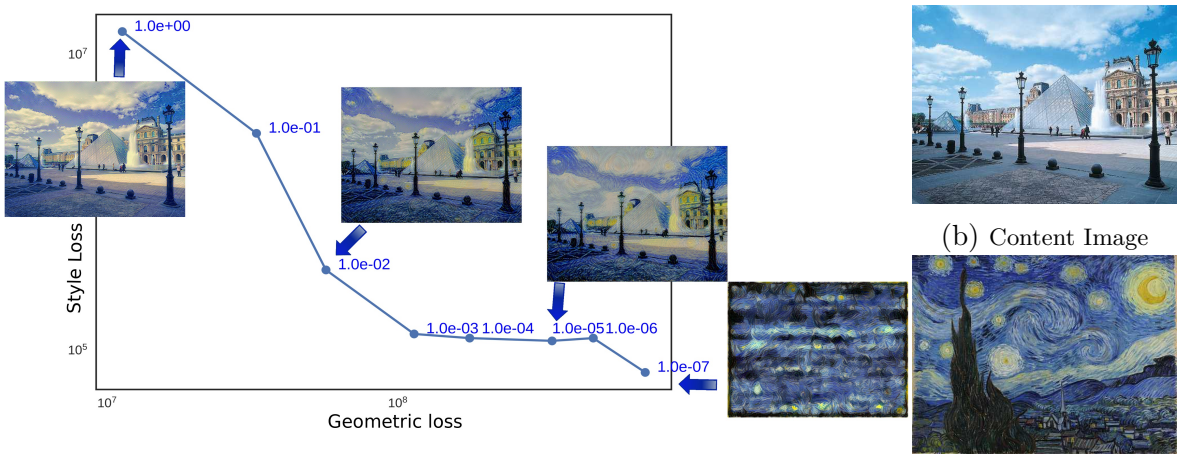


Figure 2.15: Texture synthesis [Gatys et al. 2015b] using the VGG19 architecture with weights trained on ImageNet (second column), with normalized trained weights (third column) or with random weights (last column).

Note that other pre-trained networks with different architectures can be used for texture synthesis but the VGG19 network seems to be the most efficient one, maybe thanks to its small convolution kernels [gwern 2018]. Nakano [2019] show that an adversarially robust ResNet may provide better results than non-robust one for the neural style transfer task, closely related to texture synthesis, as we will see in the next section. This also seems to be true for the other image synthesis optimization processes mentioned before: the feature visualization via activation maximization, in Section 2.2.1.1.

[45]The kernel sizes go from 3×3 to 128×128.



(a) Neural Style Transfer

(b) Content Image

(c) Style Image

Figure 2.16: Trade-off between the geometric and Gram terms of the Neural Style Transfer [Gatys et al. 2015a] cost function for several values of the ratio between the two terms. In the top right the content image: Photograph of the Louvre Museum in Paris, (source Google Images), in the bottom right *The Starry Night* by Vincent Van Gogh dating from 1889 used as style image.

2.5.2.2 Neural Style Transfer

The most impressive application of the [Gatys et al. 2015a] algorithm was one of its variants called neural style transfer. It consists in applying the style of one image to the geometric content of another as one can see in the Figure 2.16. This idea of synthesizing a texture (or stylistic image) to match a content image was formerly named "texture transfer" and previously tackled with non-parametric techniques [Eilhauer et al. 2000; Efros et al. 2001; Hertzmann et al. 2001]. Nevertheless, the CNN based image representation permits to capture bigger and more complex patterns.

The stylized image is created by minimizing the weighted sum of two loss functions thanks to gradient descent. The first term is obtained with the texture model previously described. It is used to compute a loss term between the style image and the synthesis. It is applied to a certain set of low level layers. The second term is the distance between the raw feature maps of the content image and the synthesized image, at a higher level layer. This term corresponds to the loss from the feature inversion algorithm [Mahendran et al. 2015] mentioned in Section 2.2.1.2.

The total cost function to be optimized is an additively agglomerated cost function once again. The two terms of the cost function do not evolve in the same direction so that it is sometimes necessary to degrade the geometric attachment to change the style. There is a Pareto front of the bi-objective function as we can see in the Figure 2.16a. One can see in this figure the style term as a function of the content term for several values of the weighting between the two terms. The synthesis are obtained with 2000 iterations.

Many works followed the seminal contribution of Gatys et al. [2015a], creating a certain emulation around this algorithm. The curious reader may refer to a recent review of the improvements made to this algorithm by Jing et al. [2019]. Let us note,

however, that texture synthesis and style transfer, both by CNN, are very close and that many paper studies deal with both subjects at the same time. Indeed, some issues are common to both subjects such as the ability to produce high resolution images, artifact reduction, algorithm acceleration, etc. Nevertheless, other issues are specific to the neural style transfer such as taking into account the semantic of the images [Park et al. 2019] or having a controllable synthesis [Gatys et al. 2016; Risser et al. 2017].

Neural style transfer style has several limitations. First, results illustrate that neural style transfer works better with heavy visual styles (as impressionism images or crop of highly textured images[46] for style reference) compared to realistic styles. Second, when the naturalistic images display structure on planar regions the network produces more artifacts. Lang et al. [2018] highlighted that the style in most of the approach mainly refers to color, shape, local pattern or brush stroke rather than composition, modeling of figures or choice of the topic. The style, one refers in neural style transfer is a sort of "formal" quality of artworks. Moreover, an artistic style can be defined by more than one image [Sanakoyeu et al. 2018; Wynen et al. 2018].

Nevertheless, neural style transfer and texture synthesis with CNN are an important research direction to better understand the CNN based recognition system we are using but also to better understand the relation between artistic style and semantic content in artworks. On one hand, images generated by neural style transfer or texture synthesis methods permit to show the limit of the CNN model [Brendel et al. 2019; Geirhos et al. 2019] (see in Section 2.2.1.1). Neural style transfer has been used for data augmentation or domain adaptation in numerous cases such as histology [Cicalese et al. 2020] or art recognition (as mentioned before in Section 2.3.3.2). On the other hand, several research teams try to disentangle artistic style and content, such as [Kotovenko et al. 2019] for style transfer or [Collomosse et al. 2017] for sketched based retrieval. This task is more complicated as it requires to separate style information from semantic one while they are mixed together in artworks. Thus, a better understanding of this "stylistic" part of the image though the study of the texture loss term may helps for art analysis.

2.5.2.3 Improvements on Texture Synthesis Methods with CNNs

In this section, we present an overview of the improvements proposed for the texture synthesis with CNN. In view of the method that we propose in this work, we mainly focus in this section on the works involving CNNs that have followed the seminal contribution of Gatys et al. [2015b] and particularly on works proposing new statistical constraints and focusing on long-range structure, two points that will be considered in the contributions of this manuscript (Chapter 3).

Accelerations and alternative sampling strategy In a first direction, several works have proposed ways to speed-up the synthesis process, notably through feed forward networks [Johnson et al. 2016; Ulyanov et al. 2016; Ulyanov et al. 2017; Shi et al. 2020b]. [Ulyanov et al. 2016] propose to learn a feed-forward CNN that synthesis an image from a set of multiple scales noise inputs. This generator network is trained thanks to a loss function based on the Gram matrices of the pretrained

[46] As the wave of "The Great Wave off Kanagawa" by Hokusai in 1831.

VGG19. One generator network per reference image must be learned. Johnson et al. [2016] propose a similar method but with a fully-convolutional autoencoder instead of a multi-scale architecture based on upsampling and channel-wise concatenation for scale fusions. A related work by Ulyanov et al. [2017] improve the perceptual results of Ulyanov et al. [2016] by using Instance Normalization layers but principally by learning a generator uniformly sampling the Juslez ensemble [Zhu et al. 2000]. This is done by minimizing the Kullback-Leibler divergence between the generated distribution and a quasi-uniform distribution on the Julesz ensemble. Shi et al. [2020b] aim to learn the texture synthesis optimization process from Gatys et al. [2015b] using a feed-forward network. This network takes the per-layer gradients calculated from the VGG19 as input and predicts the change to refine the input image. Such methods have enabled fast synthesis once the network has been trained for specific textures, but the quality of results is still inferior to the original approach [Gatys et al. 2015b], especially for structured textures.

Lu et al. [2015] propose an evolution of the FRAME model [Zhu et al. 1998] in the context of CNNs under the name DeepFrame. Textures are synthesized from an exponential model using features from VGG. In [De Bortoli et al. 2019], this macrocanonical approach is pushed further and fully analyzed theoretically.

Statistical constraints and losses In a different direction, a large body of works has been dedicated to add additional constraints to the synthesis, often relying on new or modified loss functions. In [Gatys et al. 2016], the color of the synthesis is constrained to specified values by color histogram matching. In [Johnson et al. 2016], a total variation term is added in the loss function for perceptual reasons. Risser et al. [2017] and Heitz et al. [2020] propose to constrain the histograms of some feature maps, in order to reduce halo artifacts. Risser et al. [2017] consider the Frobenius norm of the difference between the synthesis features map and an histogram adjusted version to match the histogram of the reference image. This target features map changes at each iteration to guide the evolution of the image being synthesized. If one considers only the term on the histograms, the results are worse than using both histogram and Gram constraints. This may be due to the iterative approximation of the histogram constraint. Heitz et al. [2020] also propose to impose the histograms of the feature maps. They consider the multi-dimensional histogram instead of the set of one dimension histograms. Their method consists in projecting the feature maps onto random directions, sorting the 1D projections and then computing the L_2 difference between the sorted vectors. This permits to capture more information from the feature maps and provides competitive synthesis without explicit constraints on the Gram matrices.

Other works such as [Liu et al. 2016a; Berger et al. 2017; Sendik et al. 2017] also propose alternative losses to add further statistical constraints. Since they explicitly deal with long -range dependency and structure, they will be reviewed in the next paragraph. It should be noted that these approaches propose to combine several statistical constraints by adding them to get the Gram matrices loss function whereas other works as [Heitz et al. 2020] have studied the use of different kinds of statistics on the feature maps. For instance, it is possible to synthesize texture by using the mean of the feature maps instead of their Gram matrices as shown in [Gatys et al. 2015b] but the result

provides clearly worse results. It is worth noting that both approaches [Lu et al. 2015; De Bortoli et al. 2019] also rely on first order constraints on features and therefore drop the use of the Gram matrices. Alternative constraints have also been investigated for the closely related task of style transfer. In [Li et al. 2017b], it is shown that matching Gram matrices reduces to kernel-based comparison of features, and various kernels are investigated in this setting. Other works investigate alternatives to the original Gram matrices, such as cross-layers (rather than within-layers) Gram matrices as in [Yeh et al. 2018] or [Novak et al. 2016], both inspired by Portilla et al. [2000]. Nevertheless, the Gram Matrices loss function is still the workhorse of the synthesis by CNN.

Incorporating long-distance dependency First, Berger et al. [2017] propose to incorporate long-distance pattern constraints by adding a cross-correlation term to the loss function. This new term is the correlation between the features maps and a shifted version of it. Different values of the shifts are used from 2 to 64 pixels for a 384 $\times$ 384 pixels images. The set of shift that it is used depends on the layers considered. In this case, the authors still use the Gram Matrices loss function. One can notice that the deeper the layer is, the smaller the shift used. This constraint seems to improve regular texture synthesis. Novak et al. [2016] propose to shift the feature maps by $\pm$ one pixel in both directions and then to compute the 9 possible Gram matrices of the feature maps. This approach is a particular case of the work mentioned above. This modification has a significant impact on the style transfer result, but in a negative way more often than not.

Then, Sendik et al. [2017] propose to add several terms (a smoothness one, a diversity one, a correlation one) to the loss function to improve the synthesis. The most interesting one is the "Deep Correlation" term. It is an complete autocorrelation[47] of the feature maps for some specific layers. This term permits imposing long-range structure in regular textures, nevertheless at the price of relatively strong artefacts. Let us mention that this approach do not consider cross correlations between different feature maps.

Apart from these works dealing with cross-correlation of features, Liu et al. [2016a] propose to incorporate the power spectrum of images in the loss function, thereby enabling the respect of highly structured textures. This work will be developed later in Section 3.2.3. In a related work, Schreiber et al. [2016] propose to impose the spectrum constraint by using a windowed Fourier Transform instead of a Fourier Transform. This enables non-stationary behaviors to be accounted for, at the cost of the inherent stationary nature of textures. Moreover, these authors propose to combine their method with the cross-correlation term from Berger et al. [2017] for a better result.

Multi-scale / multi-resolution neural synthesis Although CNNs contain in some sense a multi-scale representation of the input image, due to their hierarchical characteristic, such models are not powerful enough to deal with high resolution images. First, the approach from Ulyanov et al. [2016] and Ulyanov et al. [2017] previously mentioned can deal with high resolution images by design, at least regarding the feed-forward networks and its gradient fit on the GPU. Shi et al. [2020b] propose a multistage refinement method, by taking progressively into account the higher layers

[47] Scalar product of all the possible shifts in a feature map.

of VGG as features description. According to the authors, this improves the stability of the learning process but does not permit to create higher resolution images. An effective solution is the Snelgrove [2017]'s method [2017] which consists in creating a Gaussian pyramid of the input image before extracting Gram matrices thanks to a VGG19 network at each scale and summing up losses at each scale. The optimization is done on all scales at the same time.

On the other side, Liu [2017] and Risser et al. [2017] proposed the same year, an iterative strategy based on the idea of using the output of the synthesis process at a lower scale to initialize the next scale synthesis. This approach will be detailed in Section 3.2.2.

2.5.2.4 Other Texture Synthesis Methods

Finally, we will quickly mention some of the other texture synthesis methods that have been developed this last five years in parallel to the Gatys et al.'s one [2015b].

Generative Adversarial Network Generative Adversarial Networks (GANs) Goodfellow et al. [2014] are first used to synthesize textures by [Jetchev et al. 2016]. The training is based on the adversarial model: a discriminator tries to distinguish the fake generated samples from true ones, while a generator creates new samples. Spatial invariance assumptions are encoded in the networks as they are fully-convolutionals. An extension is introduced by Bergmann et al. [2017] to improve the textures with periodic patterns. This is made by enforcing a part of the input noise used for synthesis to be spatially periodic. Darzi et al. [2020] propose to add co-occurrences information to this method to capture local texture patterns. This is done in two ways. First, a term based on collections of co-occurrences matrices is added to the loss function and a collection of co-occurrences matrices from the reference is added to the input of both the generator and discriminator networks. Shaham et al. [2019] propose to use a pyramid of GANs. At each scale, the generator is fed with a white noise vectors and learns to output the residual image between the upsampled version of the previous scale synthesis and the current scale one. The discriminators are trained to discriminate patches and not the whole image to respect fine-grain details. Zhou et al. [2018] propose to train a fully convolutional generator designed to extend a texture of size $k \times k$ to a synthesis of size $2k \times 2k$. Their GAN model is trained with a L_1 loss and Gram matrices-based perceptual losses in addition to the adversarial loss. These three different losses are needed to produce perceptually acceptable synthesis. With these methods, once the generator has been learned, it is possible to generate multiple synthesis at high speed and low memory footprint. Moreover, this method scales easily to high resolution outputs.

Frühstück et al. [2019] have an alternative use of the GAN model. They propose to use a variant of the Markov Random Field model applied on the latent fields of the generator from a pretrained GAN. Nevertheless, this implies to sample a large set of textures to cover the latent space.

Recurrent Neural Network Theis et al. [2015] use a Recurrent Neural Network (RNN) to learn the pixel probabilities and statistical dependencies of natural images. Their model is a spatial Long Short-Term Memory (LSTM) network working on a

two-dimensional grid. It permits to generate images by taking into account neighbors values. Cai et al. [2019] propose an ensemble of three LSTMs for texture synthesis. The first one is trained to synthesize the first row of the image, a second to synthesize the first column and the last one the rest of the image. This last network output is conditioned by the upper triangular ensemble of pixels.

Multi-scale patch based methods Recently, new patch-based methods have re-emerged. Galerne et al. [2018] apply semi-discrete optimal transport in a multi-scale[48] patch space. A multi-scale strategy is added to synthesize high resolution structured textures. Houdard et al. [2020] also use optimal transport and a multi-scale strategy. They explicitly optimize the discrete formulation of the Wasserstein distance between patch distributions, at different resolutions. In another direction, Houdard et al. [2020] propose to train the Ulyanov et al.'s model [2016] with a semi-discrete Wasserstein formulation of the patches distribution instead of using a distance on the Gram matrices of the VGG19.

2.5.2.5 Evaluation of Texture Synthesis Methods

The evaluation of texture synthesis results is a challenging task. The visual quality is strongly subjective and there is not universal perceptual metric. On the one hand, some metrics have been proposed to evaluate generated images such as the "Single Image Fréchet Inception Distance" proposed by the previously mentioned work of Shaham et al. [2019] about GAN. Other works propose to used metric given by the Gram matrices of VGG features as used in [Gatys et al. 2015b] or optimal transport distance between patches distribution [Houdard et al. 2020]. Nevertheless, most of the time, the existing metrics are already used for the optimization process and are not capable of capturing the complexity of texture images. On the other hand, a perceptual study can help to distinguish two methods or to compare the synthesis to the real images. Indeed, it is shown in [Clarke et al. 2011; Dong et al. 2013; Dong et al. 2020] through extensive experiments that feature-based evaluations do not approach well human-based visual evaluation of texture similarity, especially in the case of long-range correlations. To address this need, Shaham et al. [2019] run an Amazon Mechanical Turk "Real/Fake" user study to evaluate their SINGAN model.

In Chapter 3, we propose two metrics for comparing texture synthesis methods, the first one is based on displacement map and the second one on wavelet decompositions. Moreover, we run a user study to rank five competing methods including two proposed by us.

[48] This is a multi-resolution strategy according to the definition, we will use later in Section 3.2.2.

3

Texture Synthesis With Convolutional Neural Networks

Contents

Abstract

In this chapter, we present several neural synthesis methods that significantly improve the ability to preserve the large scale organization of textures. We first propose a simple multi-resolution framework that accounts for large scale structures and allows the synthesis of high-resolution images. We then show that, in this multi-resolution framework, additional constraints are useful in the case of regular textures. A first approach combines the classical statistical constraints of neural approaches [Gatys et al. 2015b] (Gram matrices) with Fourier frequency constraints. Alternatively, the multi-resolution framework can be combined with a statistical constraint relying on the full autocorrelation of the features of the network. In an experimental part, the proposed methods are then extensively tested and compared to alternative approaches, both in an unsupervised way and through a user study. Experiments show the interest of the multi-resolution scheme for high resolution textures and the interest of combining it with additional constraints for regular textures. The last section is dedicated to unsuccessful attempts of using different statistics than Gram matrices of the feature maps.

Some of the work in this chapter is under review :

- Gonthier N., Gousseau Y., Ladjal S. *High resolution neural texture synthesis with long range constraints*; submitted at Journal of Mathematical Imaging and Vision in 2020.

3.1 Introduction

One challenge that has been faced by all methods since the early days of texture synthesis is the multi-scale nature of texture samples, implying that models should be able to reproduce both small and large scales, possibly over several orders of magnitude. For instance, parametric models for Markov Random Fields are known to be intrinsically badly suited to a multi-scale modeling. Zooming such a model by a given factor implies extremely heavy computations to derive the corresponding parameters [Gidas 1989], impairing the design of multi-scale such models. Wavelet models are more adapted by nature to multi-scale modeling, but the faithful reproduction of structured textures requires complex interactions between scales to be accounted for. The best such modeling up to date is the second order statistical model proposed in [Portilla et al. 2000], but highly structured textures still represent a challenge to such approaches. Nonparametric Markov modeling methods such as those presented in [Efros et al. 1999; Efros et al. 2001] indeed have the ability to deal simultaneously with several scales albeit at a high computational cost. However, they are also well known to produce textures with very little variety, often producing verbatim copies, see [Aguerrebere et al. 2013] and the experiments in the present chapter. The methods relying on CNNs, following the seminal work by Gatys et al. [2015b], are currently the most efficient to capture multi-scale structures. Nevertheless, they still lack efficiency when large scale regularity is needed.

In this chapter, we present several neural synthesis methods that significantly improve the ability to preserve the large scale organization of textures. We first recall the classical approach from Gatys et al. [2015b] in Section 3.2.1. Then, we propose a simple multi-resolution framework that account for large scale structures and allows the synthesis of high-resolution images (Section 3.2.2). We show that, in this multi-resolution framework, additional constraints are useful in the case of regular textures. A first approach combines the classical statistical constraints of neural approaches [Gatys et al. 2015b] (Gram matrices) with Fourier frequency constraints (Section 3.2.3), similar to those introduced by Galerne et al. [2011]. Alternatively, the multi-resolution framework can be combined with a statistical constraint relying on the full autocorrelation of the features of the network (Section 3.2.4). This approach is closely related to the one introduced in [Sendik et al. 2017], which combines correlations with Gram matrices and various additional constraints. We show that correlation terms alone yield excellent results and therefore that Gram matrices are not necessary in this case.

We then evaluate the proposed methods in an extensive experimental in Section 3.3. The evaluation of texture synthesis results is a challenging task. Some approaches draw on well-chosen statistics to estimate the quality of the results (the closest to the exemplar, the better), as for instance discussed in [Clarke et al. 2011]. In this paper, we first evaluate results in this manner, relying on Kullback-Leibler divergence between wavelet marginals, following the texture indexing scheme from Do et al. [2002]. We therefore rely on a user study to compare the framework we propose to both the original method from Gatys et al. [2015a] and some of its improvement that focus on the respect of large scale structures.

The last Section 3.4 is dedicated to unsuccessful attempts of using different statistics (such as moments, norms or spectrum) than Gram matrices of the feature maps.

3.2 Multi-resolution Strategy with Spectral or Autocorrelation-based Control

3.2.1 Reminder on the Work from Gatys et al. [2015b]

The Gatys et al.'s method [2015b] based on CNN feature maps as texture representation works as follows. Given a texture exemplar $I \in \mathbb{R}^{h \times w \times 3}$, where $n = h \times w$ is the number of pixels in the image, we first input I to the CNN to calculate the feature maps at each layer. Following the notations introduced in Section 2.1.1.1, $f^l \in \mathbb{R}^{h_l \times w_l \times m_l}$ is the extracted feature maps of the layer l with $n_l = h_l * w_l$ the number of "pixels" of this layer. To synthesize a new texture, we choose a subset[1] $\mathcal{S}$ of layers of the CNN on which we impose some statistics. We denote Φ_l the statistics computed on the feature maps of the layer l. Then we obtain a set of statistics $\{\Phi_1(f^1), \Phi_2(f^2), \cdots, \Phi_L(f^L)\}$ which is used for the definition of the loss function. This loss function is defined as the sum of the Frobenius norm between the reference image statistics and the new image ones for all the layers considered. It is defined as:

$$\mathcal{L} = \sum_{l=1}^{L} \alpha_l \|\Phi_l(f^l) - \Phi_l(\tilde{f}^l)\|_{\mathcal{F}}^2, \tag{3.1}$$

with $\alpha_l \in \mathbb{R}$ the weight of layer l. We impose constraints on each layer independently with this loss function and no inter-layer statistics. To generate a new texture image $\tilde{I}$ on the basis of a reference one I, we use a gradient descent from a white noise image to find an image that matches the reference statistics. Usually the L-BFGS-B [Zhu et al. 1997] second-order optimization method provides the best results.

Depending on the choice of the $\Phi_l(\bullet)$ function applied to the features maps of the layer l, we have different texture models. If $\Phi_l(\bullet)$ is the identity, we are in the case of the feature inversion method [Mahendran et al. 2015] mentioned in Section 2.2.1.2. The Gatys et al.'s original method [2015b] used the covariances (also called *Gram Matrix*) of the feature maps as statistics $\Phi_l(f^l) = G^l \in \mathbb{R}^{m_l \times m_l}$. The Gram matrix $G^l \in \mathbb{R}^{m_l \times m_l}$ is given by the scalar product between these different channel responses:

$$G^l_{p,q} = \frac{1}{n_l^2} \sum_{i=1}^{n_l} \vec{f}^l_p(i) \cdot \vec{f}^l_q(i) = \frac{1}{n_l^2} \langle \vec{f}^l_q, \vec{f}^l_p \rangle, \tag{3.2}$$

with p, q the index corresponding to the channels $p, q \in \{1, \cdots, m_l\}$, $\vec{f}^l_p \in \mathbb{R}^{n_l}$ the pointwise vector version of f^l_p. We will denote the version of the loss function from equation (3.1) with the Gram matrices $\mathcal{L}_{Gram,I,\mathcal{S}}$.

If $L > 1$, the loss function equation (3.1) can be seen as one of a multi-objective cost function agglomerated into a single-objective cost function. We may wonder if the different objectives $\|\Phi_l(f^l) - \Phi_l(\tilde{f}^l)\|_{\mathcal{F}}^2$ have comparable ranges. However, by choosing identical weights, i.e. $\alpha_l = 1 \; \forall \; l \in [1, L]$, we obtain perceptually acceptable results.

[1] For simplicity's sake, we will note the selected layers from 1 to L in the rest of this document but the user can choose non-consecutive layers in the network.

Synthesis examples and the influence of some parameters of this method can be seen in Section 2.5.2.1.

3.2.2 Multi-resolution Synthesis

The first modification we introduce to the method from Gatys et al. [2015b] is a straightforward multi-resolution framework that will help preserve the large scale organization of images. This strategy is relatively classical for texture synthesis methods and has been used in the past in different settings [Kwatra et al. 2005; Tartavel et al. 2015; Liu 2017; Risser et al. 2017]. This approach is much simpler than the related method introduced in [Snelgrove 2017] and, as we will see in the experimental section, yields better results.

The idea is simply to first synthesize a coarse resolution image, which is then upsampled and given as initialization for a synthesis at the next scale. This process is repeated K times until the desired resolution is reached. As illustrated in Figure 3.1, we first build an image pyramid from the exemplar image I, iteratively down-sampling it by factors $2^1, 2^2, \cdots, 2^K$, resulting in images $I^{(1)}, I^{(2)}, \cdots, I^{(K)}$. A first synthesis result is obtained by using the smallest image as the exemplar and white noise as initialization. Then, for step $k \in K, K-1, \cdots, 1$, we generate a new result using $I^{(k)}$ as the exemplar and the obtained synthesis result $\tilde{I}^{(k-1)}$ as the initialization instead of white noise. The upsampling of $\tilde{I}^{(k-1)}$ is performed using bilinear interpolation. The only parameter of this generic multi-resolution framework is the number of scales K.

This kind of multi-resolution strategy was first explored for texture synthesis with CNNs by Liu [2017], for style transfer with CNNs by [Gatys et al. 2016]. Both tasks were also studied by Risser et al. [2017]. In this last paper, it is mentioned that the multi-resolution strategy improves the quality of the synthesis but there is no full review of the advantage of this pipeline.

As can be seen in Figure 3.15, this strategy can yield strong improvements in some cases but is not enough to allow the reproduction of highly structured textures. In the following, we investigate in details the ability of our approach to synthesize high resolution textures. We show how the result can be improved by adding a careful control of the Fourier spectrum or by using the autocorrelation instead of the Gram matrices into the multi-resolution scheme. Nevertheless, even for low resolution images, the results using only the multi-resolution strategy have a better ability to preserve large structures than those without it.

This method can be considered to be analogous to the earlier work by Portilla et al. [2000]. Indeed, in this work, the authors use a coarse-to-fine procedure through the steerable pyramid decomposition of the image.

3.2.3 Spectrum Constraint

We propose to include in the synthesis a new constraint based on the Fourier spectrum of the image. It is known that such a constraint alone is an efficient way to reproduce the so-called *micro-textures* [Galerne et al. 2011] made of uniformly distributed small details. This constraint has also been used in combination with more structured synthesis methods as in [Tartavel et al. 2015].

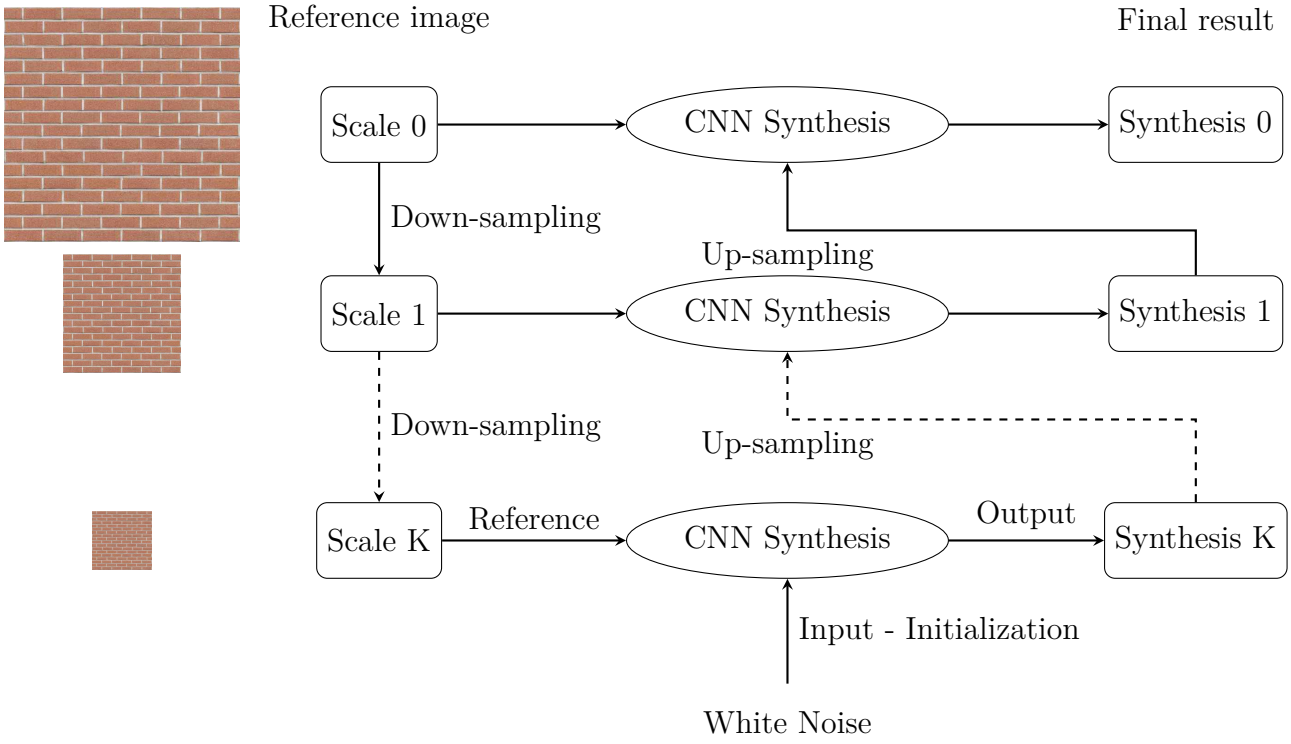


Figure 3.1: Multi-resolution strategy. The exemplar is down-sampled by factors $2^{-1}, 2^{-2}, \cdots, 2^{-k}$ to build a pyramid $I^{(1)}, I^{(2)}, \cdots, I^{(k)}$. At resolution K, a new texture is synthesized by using $I^{(k)}$ as the exemplar and the upsampled result of the synthesis at scale $K-1$ as initialization (instead of white noise). We repeat this step until we reach the top of the image pyramid. Figure based on Liu [2017].

Let us write $\mathcal{F}(I)$ for the Discrete Fourier Transform (DFT) of an image I, $\mathcal{F}^{-1}$ for the inverse DFT and $|\bullet|$ for the complex modulus. The idea is to constrain the synthesized image $\tilde{I}$ to have a Fourier spectrum $|\mathcal{F}(\tilde{I})|$ as similar as possible to $|\mathcal{F}(I)|$, the spectrum of I. A simple way to do this is to first perform the multi-resolution neural synthesis described above, and then to replace the phases of the Fourier transform of the synthesized image with random phases, before applying the inverse Fourier transform to the result [Galerne et al. 2011]. Now, this sequential strategy is not satisfactory, since the randomization of phases would destroy the effect of both the statistical constraints on the VGG features and the effect of the multi-resolution strategy. Therefore, we propose to incorporate the Fourier constraint into the multi-resolution synthesis process. A preliminary, mono-scale version of this idea was presented in [Liu et al. 2016a].

In order to include the Fourier constraint into the loss function used for synthesizing images, we first introduce $\mathcal{E}_I$, the set of images having the same spectrum as I the exemplar image. In the case of color images, this is defined as

$$\mathcal{E}_I = \left\{ J \in \mathbb{R}^{h \times w \times 3} \mid \exists\, \phi \in \mathbb{R}^{h \times w} : \mathcal{F}(J) = e^{i\phi} \mathcal{F}(I) \right\}.$$

Next, we define the Fourier loss associated with the image $\tilde{I}$ as the normalized Euclidean

distance between $\tilde{I}$ and $\mathcal{E}_I$,

$$\mathcal{L}_{spe} = \frac{1}{2n} d(\tilde{I}, \mathcal{E}_I)^2 = \frac{1}{2N} \|\tilde{I} - \mathcal{P}_I(\tilde{I})\|^2 \tag{3.3}$$

and the total loss as:

$$\mathcal{L} = \mathcal{L}_{Gram} + \beta \mathcal{L}_{spe}, \tag{3.4}$$

where β is a weighting parameter and $\mathcal{P}_I$ is the projection operator on $\mathcal{E}_I$. This projection is given by (see [Tartavel et al. 2015, Appendix A]):

$$\mathcal{P}_I(\tilde{I}_c) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\tilde{I}) \cdot \mathcal{F}(I)}{|\mathcal{F}(\tilde{I}) \cdot \mathcal{F}(I)|} \cdot \mathcal{F}(I_c)\right), \quad c \in \{r, g, b\}, \tag{3.5}$$

where $\cdot$ is the scalar product in $\mathbb{C}^3$, that is

$$\mathcal{F}(\tilde{I}) \cdot \mathcal{F}(I) = \sum_{c=r,g,b} \mathcal{F}(\tilde{I}_c)\mathcal{F}(I_c)^*,$$

with I_c, for $c = r, g, b$, being the color channels of I and a^* the conjugate of complex number a. This constraint can be seen as a specific statistic Φ_0 on the layer "0" of the network, i.e. its input[2], with $\Phi_0(X) = X - \mathcal{P}_I(X)$ and a weighting term $\alpha_0 = \frac{\beta}{2n}$. This spectrum constraint can be seen as a regularization to the Ill-posed example based synthesis problem. We can see the β term as a parameter to synthesis texture to create image in the spectrum of texture from random one to almost periodic ones.

3.2.4 Autocorrelation of the Feature Maps

In this section, we consider an alternative way to impose long-range consistency, based on the autocorrelation of the features maps. This is motivated by the fact that the autocorrelation is a proxy of repeating patterns, such as the presence of periodic elements in the signal. We can track the used of autocorrelation back to the 50's. Kaizer [1955] showed that autocorrelation can be used for characterizing textural features of aerial photographs of Arctic regions[3]. As explained in the third paragraph of Section 2.5.2.3, this idea has been explored for texture synthesis with CNNs with different modalities in [Novak et al. 2016; Berger et al. 2017; Sendik et al. 2017].

The autocorrelation function of an image is defined as the convolution of the image with itself. Let $I \in \mathbb{R}^{h \times w}$, the autocorrelation $C(I) =\in \mathbb{R}^{h \times w}$ is defined, for $\forall k \in \{1, \cdots, h\}$ and $\forall l \in \{1, \cdots, w\}$, as

$$C(I)(k, l) = \frac{1}{n^2} \sum_{i=1}^{h} \sum_{j=1}^{w} I(i, j) I(|\ i + k\ |_h, |\ j + l\ |_w) \tag{3.6}$$

$$= \frac{1}{n^2} I * I, \tag{3.7}$$

[2]The same formulation can be used for the TV term of Johnson et al. [2016] for instance.
[3]According to Haralick [1979].

with $|\bullet|_h$ being the modulo operation with divisor h. An efficient way to compute the autocorrelation is to use the DFT. According to the Wiener-Khintchin theorem, we have:

$$C(I) = \mathcal{F}^{-1}(|\mathcal{F}(I)|^2). \tag{3.8}$$

Then, we define the autocorrelation constraint at the layer l as

$$\Phi_l(f^l) = A^l, \tag{3.9}$$

with $A^l \in \mathbb{R}^{h_l \times w_l \times m_l}$ the tensor of the squared modulus of the Fourier transform of the features maps, i.e.:

$$A^l_p = \frac{1}{n_l^2} |\mathcal{F}(f^l_p)|^2, \tag{3.10}$$

with $p \in \{1, \cdots, m_l\}$ the corresponding indexes of the feature map p. Using this toric representation allows one to consider all possible shifts between pixels.

This constraint is similar to the one in [Sendik et al. 2017], except that, in our case, the autocorrelation is computed in the Fourier domain and there is no weighting of the elements of the autocorrelation matrix by the inverse of the total amount of overlapping regions. Moreover, at a fixed layer, we consider each feature maps independently whereas in [Berger et al. 2017] they compute correlation between the different features maps of a layer. Even if we don't explicitly impose the correlations between channels at a given layer, they are implicitly controlled at the previous level. Indeed, the current layer recombines all the channels of the previous one, on the contrary to wavelet or scattering transforms.

3.3 High Resolution Experiments

In this section, we perform experiments to illustrate both the multi-resolution framework and the additional constraints we propose for neural texture synthesis. After briefly introducing the methods we compare ourselves to, we first show some visual results. Then, we propose a method to evaluate the innovation capacity of algorithms, and more precisely their tendency to produce verbatim copy of the input. Further, we evaluate the methods quantitatively using the Kullback-Leibler divergence between wavelet statistics. Despite the interest of such quantitative evaluations, it is known that they have severe limitations, in particular to evaluate results at large scales [Dong et al. 2020]. Therefore, we have also conducted a medium scale perceptual evaluation from human observers, the results of which we analyze in Section 3.3.3.3. These different evaluations have been conducted on the 20 texture images visible in Figure 3.7. These high resolution (1024×1024) textures have been chosen to include both regular and irregular textures. Some of them display strong long-range dependency. Eventually, we study the effects of various parameters and briefly illustrate the ability of our method to produce higher resolution textures.

3.3.1 Architecture and Parameters

We use a VGG-19 network pre-trained on ImageNet with rescaled weights[4] as in [Gatys et al. 2015b] and we also use the same layers[5] i.e.: 'conv1_1', 'pool1', 'pool2', 'pool3', 'pool4'. The corresponding weights[6] are set to be $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = 10^9$. When the spectrum constraint is considered, we use a weighting parameter $\beta = 10^5$ unless otherwise specified. Synthesis are performed using 2000 iterations. We use Tensorflow as a deep learning framework and Scipy as an optimization package. We do not perform a histogram matching of the output image, on the contrary to several research works as [Gatys et al. 2016]. Synthesizing one texture of size 1024×1024 takes 85 minutes with a GeForce 1080 Ti for the method multi-resolution "Gram + MRInit". The overhead compared to Gatys et al. [2015b] at the same scale is limited to 25 minutes because the synthesis at lower resolutions are faster.

3.3.2 Other Texture Synthesis Methods

The first method we compare ourselves to is the original synthesis method from Gatys et al. [2015b], that from now we refer to as "Gatys". We also consider the method "Deep Corr", introduced in [Sendik et al. 2017], using the code from the authors[7], with a maximum of 2000 iterations. We also consider the multi-scale[8] texture algorithm from Snelgrove [2017], using the code from the author[9], with layers 3 and 8 and 5 octaves in the Gaussian pyramid as in the original paper. We use a maximum of 2000 iterations. From now on, we refer to this method as "Snelgrove". These last two methods have been chosen because they explicitly address the problem of reproducing large scale structures. We also consider the Feed Forward approach proposed in [Ulyanov et al. 2016] using a PyTorch implementation by Jorge Gutierrez[10]. We refer to this method as "Ulyanov". Finally, we consider two patch-based methods, from the works [Efros et al. 1999] and [Efros et al. 2001], using implementations from the online journal IPOL [Aguerrebere et al. 2013; Raad et al. 2017b], with default parameters settings. We refer to these two methods respectively as "Efros Leung" and "Efros Freeman".

3.3.3 Visual Comparisons

In Figures 3.2 to 3.5 we display synthesis results using our methods and those presented in the previous paragraph. For space reason, we only consider 4 textures, all exhibiting

[4] The rescaled VGG-19 network can be found at `http://github.com/leongatys/DeepTextures`

[5] See Figure 2.4b or Table D.3 in the Annex for the names of the layers

[6] Due to the numerical sensitivity of the LBFGS-B optimization algorithm.

[7] The code of Sendik et al. [2017] can be found on Github: `https://github.com/omrysendik/DCor`

[8] Although Snelgrove [2017] uses the term "multi-scale" of his method, we could maybe point that it is more a "multi-resolution" approach. Indeed, Kwatra et al. [2005] use the term "multi-resolution" about the use of several image resolutions and the term "multi-scale" about the use of different patch size. The "multi-scale" term refers in a similar way to the cascade of filters used to compute image representations as in [Heeger et al. 1995]. Nevertheless, these two terms appear very close to each other.

[9] The code of Snelgrove [2017] can be found on Github: `https://github.com/wxs/subjective-functions`

[10] `https://github.com/JorgeGtz/TextureNets_implementation`

some kind of long-range dependency. Their resolution is 1024×1024. Some details can be seen in Figure 3.6. All results can be seen in Annex (Section A.1).

We first notice that patch-based methods are very faithful to the reference image. However, they have the tendency to produce regions that are exact copy of the input, a verbatim effect already noticed in [Aguerrebere et al. 2013] and investigated in the next section. They also at times yield images with constant or repetitive patterns.

Among neural methods, the original "Gatys" method is still competitive, but struggles to reproduce large scales on these high resolution textures. This is due to the size of the receptive fields, which is clearly not sufficient in this case. The method from "Ulyanov" is worse in this respect. The method "Deep Corr" improves the preservation of large scale structures, but results are not satisfying, some structures are lacking and artifacts are visible. The default weighting of the four terms of the loss function used by Sendik et al. [2017] is maybe not suitable for high resolution synthesis. In contrast, the plain use of the autocorrelation term as an additional constraint, as we propose in "Autocorr", yields better results, even though no use of the Gram matrices is made. The regularization and innovation terms present in the method from Sendik et al. [2017] may also be harmful in these cases. Next, we observe that adding the Fourier spectrum constraint alone (at a single scale) yields interesting results, but is not enough to get fully satisfying results.

The multi-resolution methods, be it "Snelgrove" or the one we propose, "Gram+MRInit", "Gram+Spectrum+MRInit", "Autocorr+MRInit", all improve the original synthesis method "Gatys". In the case of very regular textures, as in Figures 3.3 to 3.5, our multi-resolution methods "XXX+MRInit" yields better results, as will be confirmed by the user study in Section 3.3.3.3. The method "Autocorr+MRInit" sometimes yields results that are clearly better than others, especially for very structured textures, as can be seen in Figure 3.4 or on the last line of Figure 3.6. Nevertheless, it also sometimes fails as in Figure 3.2 and may produce artifacts on some examples. For this reason and for human resource constraints, we choose, among our methods, to only include "Gram+MRInit" and "Gram+Spectrum+MRInit" in the user study presented in Section 3.3.3.3.

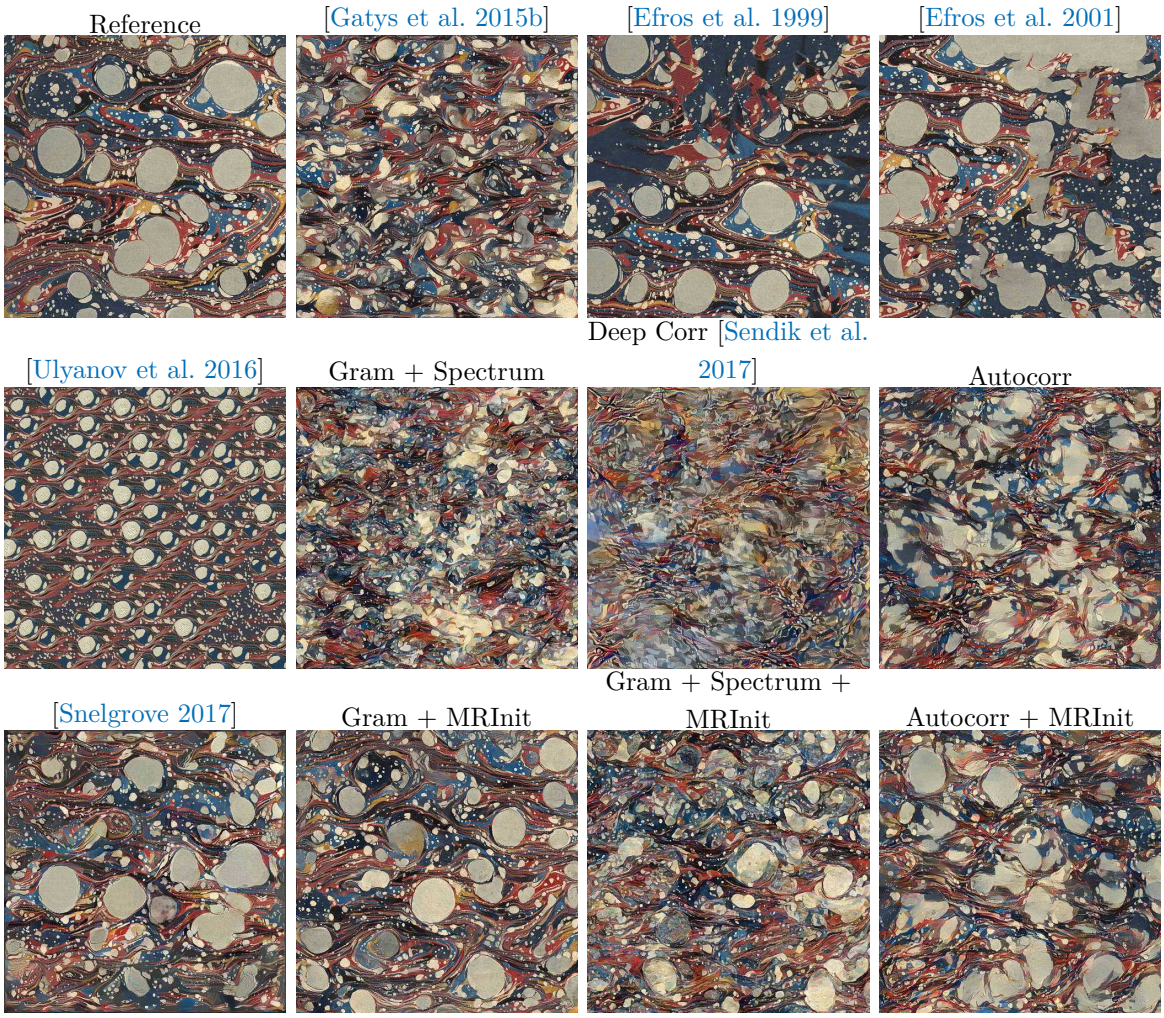


Figure 3.2: Synthesis results using different methods for a given reference of size 1048 × 1048.

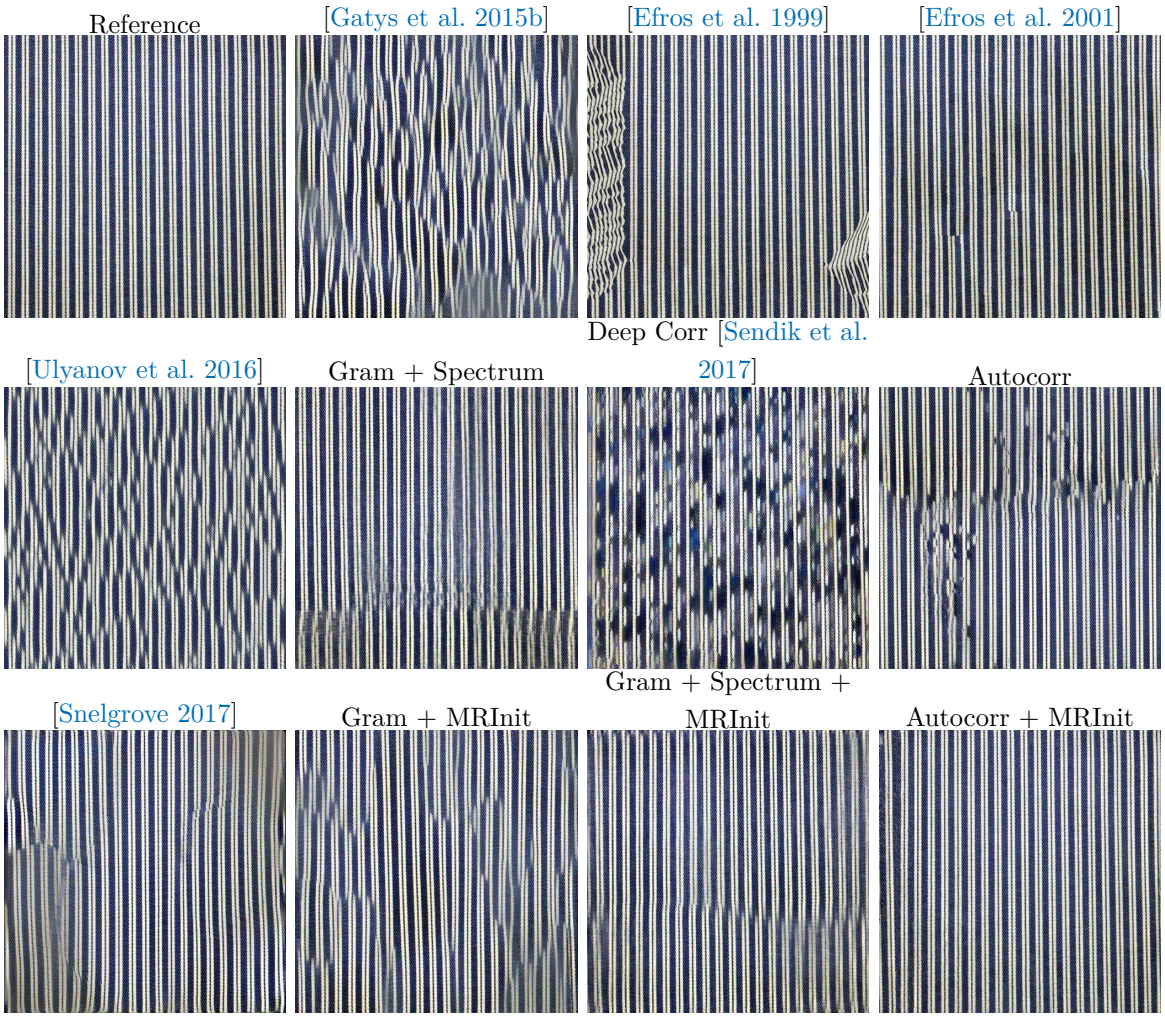


Figure 3.3: Synthesis results using different methods for a given reference of size 1048 × 1048.

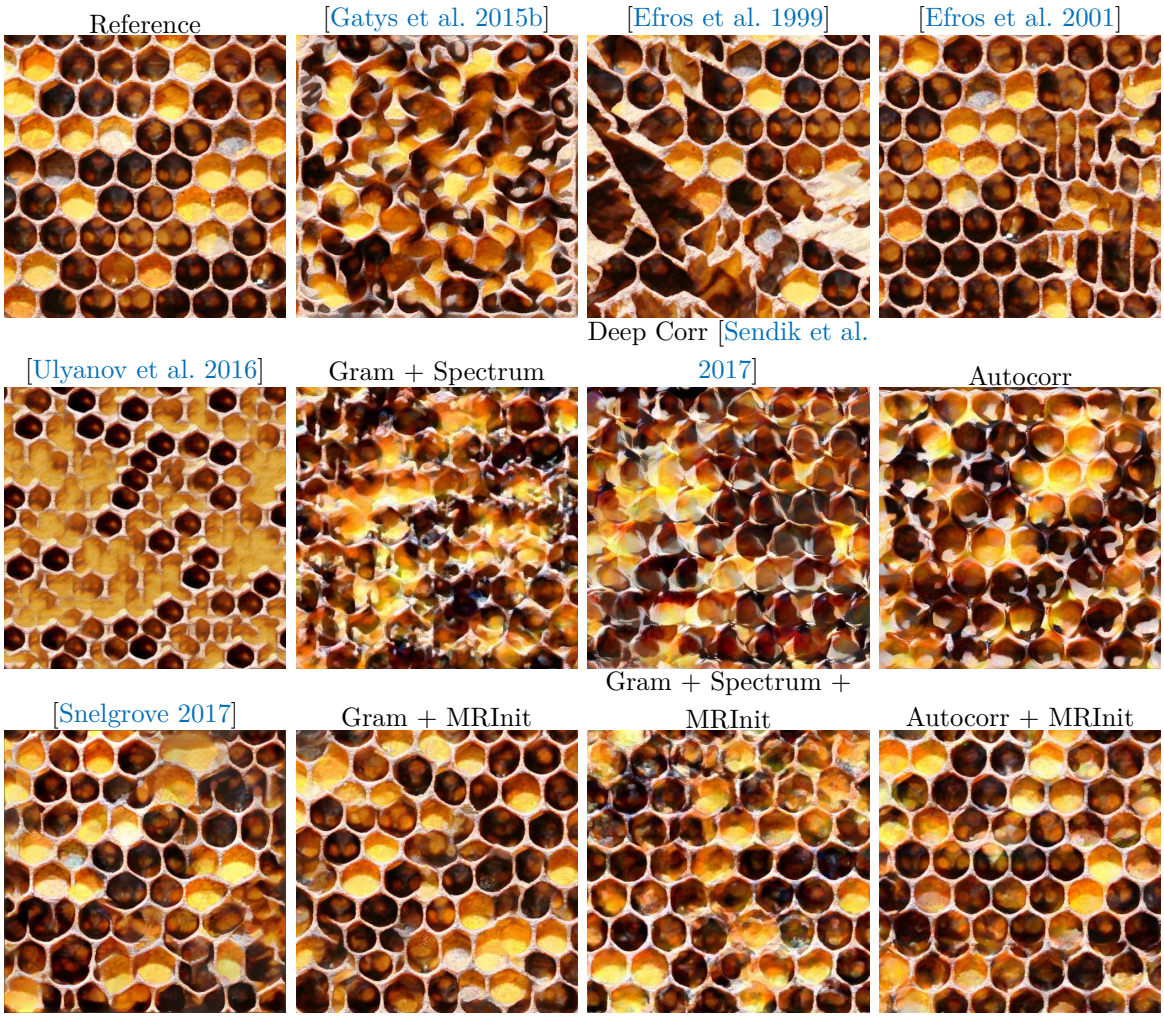


Figure 3.4: Synthesis results using different methods for a given reference of size 1048 × 1048.

Figure 3.5: Synthesis results using different methods for a given reference of size 1048 × 1048.

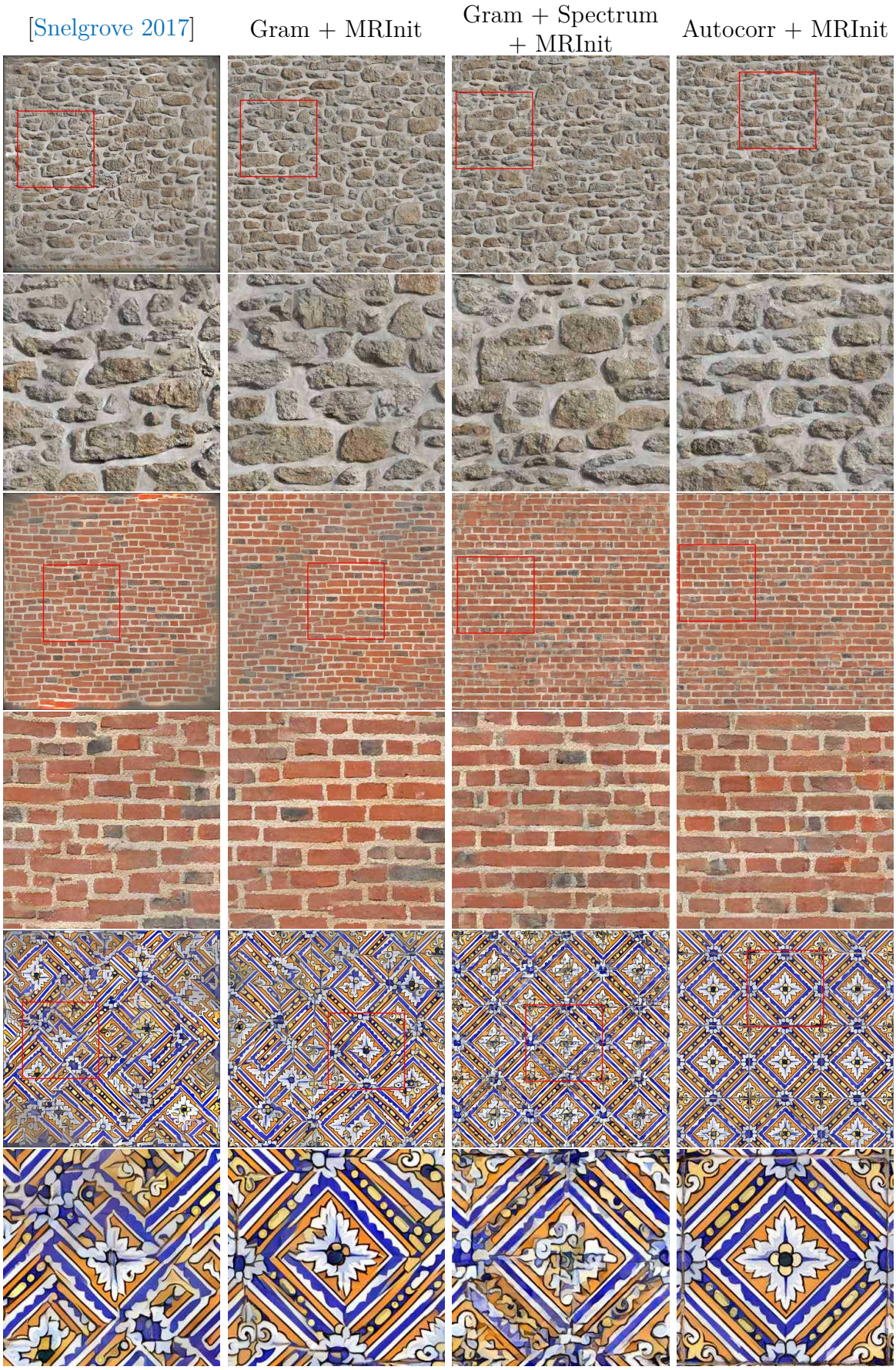


Figure 3.6: Zoom in some of the texture synthesis results. For the MSInit cases, we use $K = 2$. The region of each image framed by a red square is shown the row below.

Figure 3.7: Reference images used in the different evaluation methods.

3.3.3.1 Verbatim Copy

Texture synthesis methods should have the capacity to produce new images that are as diverse as possible. In the pioneering work FRAME [Zhu et al. 1997], this is achieved by maximizing the entropy. Similar ideas have recently been explored in [Lu et al. 2015; De Bortoli et al. 2019]. Following these ideas, texture synthesis methods could be evaluated based on their capacity to maximize the entropy under some given constraints. Such a quantitative evaluation, however, is far from being trivial and probably not tractable.

In this section, we take a pragmatic and much more modest way. We propose a simple way to evaluate the tendency of methods to locally produce verbatim copy of the input. This is a known default of patch-based methods, see e.g. [Aguerrebere et al. 2013; Raad et al. 2017b]. For each pixel of a given synthesis result, we look for its nearest neighbor in the input image. The notion of proximity is defined by comparing small square neighborhoods (patches) around each pixel. In Figure 3.8, we display the corresponding displacement map. The used color scale is obtained by assigning the x coordinate of the displacement map to red, and the y coordinate to blue. Verbatim copies of the input appear as constant regions in these displacement maps. As expected, the only two methods displaying large such regions are patch-based methods. All others seem to produce a reasonable amount of innovation, even though the multi-resolution method from Snelgrove [2017] can very occasionally produce small verbatim copies, probably due to the strong constraints it puts on the Gaussian pyramid.

In order to quantify the visual effect of the displacement maps, we propose to measure the flat regions corresponding to locally constant displacements. For each pixel of the displacement map, we count how many of its neighbors (in 4-connexity) share its color value. Denoting n_s this number, a score is defined as $DS = 1 - n_s/N_n$, where N_n is the total number of investigated neighbors. The more verbatim copy there are in the synthesis, the closest the score is to 0. The boxplots of this score for the different methods and the twenty reference images can be seen in Figure 3.9. They confirm the impression given by the displacement map that the patch-based methods yield significantly more verbatim copy than neural methods. It should be noted, however, that the proposed methodology is relatively rough and does not account either for small perturbations on the pixel positions nor for noisy pixel values.

3.3.3.2 Feature-based Evaluation

Feature-based evaluation of textures is not straightforward, because no existing feature is considered as the reference one. Moreover, such evaluations are inherently biased. In the most extreme case, one could even try to optimize the chosen features to synthesize new textures. In this work, we choose to rely on wavelet filters that both are classical texture features and are not used in any of the considered methods. More precisely, we rely on the texture features proposed in [Do et al. 2002]. In this paper, two textures are compared by computing the Kullback–Leibler (KL) divergence between parametric estimation (using generalized Gaussians) of the marginal distributions of wavelet coefficients. In order to quantify the proximity of a synthesized texture to the reference image, we propose to:

1. Compute the wavelets coefficients of the reference image and the synthesized one (in our case we choose a Daubechies 4 wavelets as in [Do et al. 2002] with 8 scales

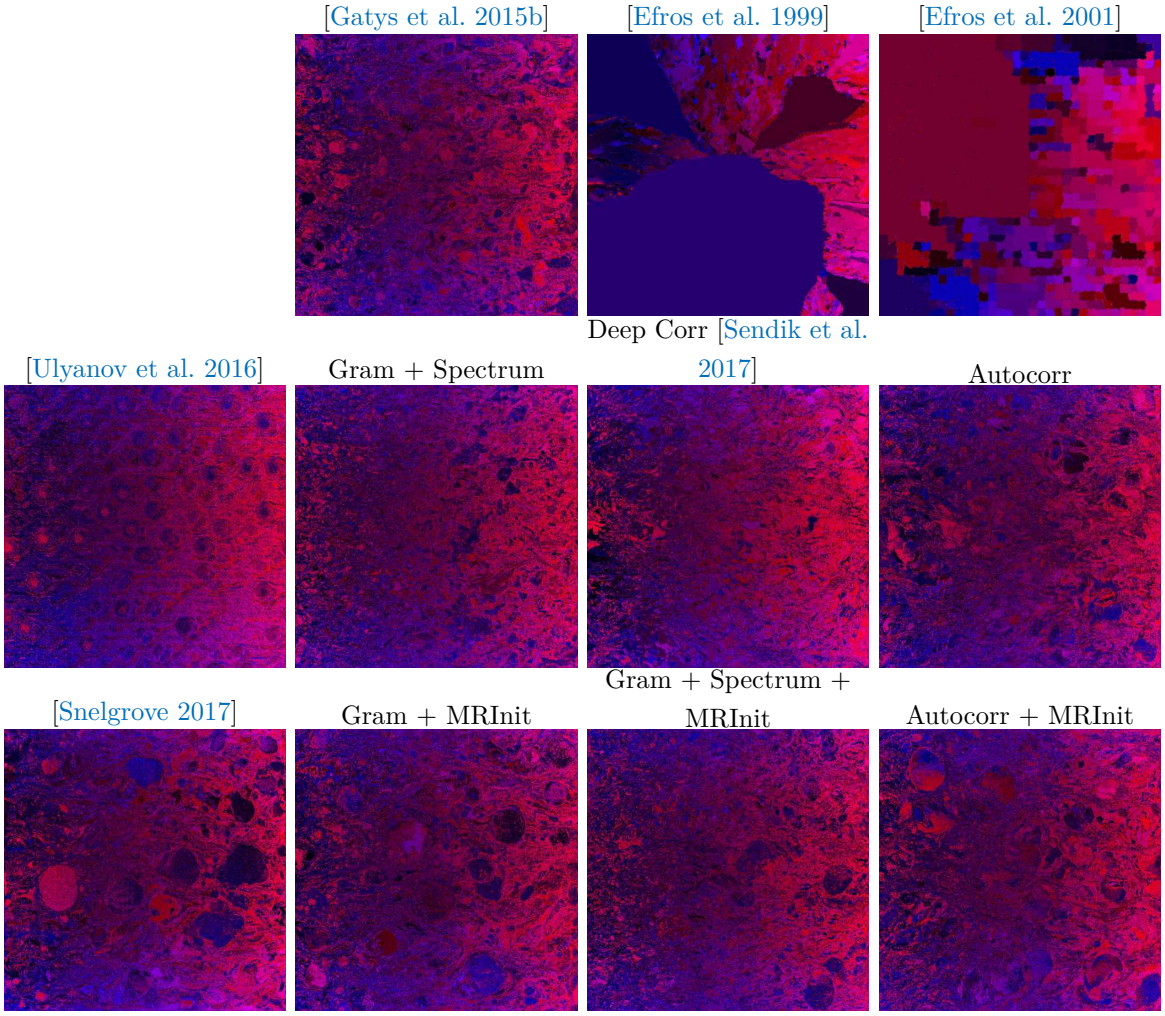


Figure 3.8: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure 3.2.

instead of 3, in order to account for large scale structures).

2. For each scale and orientations estimate the parameters of a generalized Gaussian from the empirical distribution of wavelets coefficients
3. For each scale and orientations compute the KL divergence between the estimated generalized Gaussians (using a closed-form formula)

We display in Figure 3.10 the boxplots of the log KL scores over the 20 considered images, for the different methods. For each box, the horizontal orange line corresponds to the average result and the star to the median. On the average, the best method for this evaluation scheme appears to be "Gram+MRInit". Then follow the two patch-based methods. This is in agreement with results from the previous paragraph, since indeed a verbatim copy will have a perfect score. The next method is "Gram+ Spectrum+MRInit", followed by "Snelgrove" and "Autocorr+MRInit". This evaluation confirms the good quality of results produced by the proposed "XXX+MRInit" methods, as well as "Snelgrove", at least on this image dataset containing a relatively high proportion of structured textures.

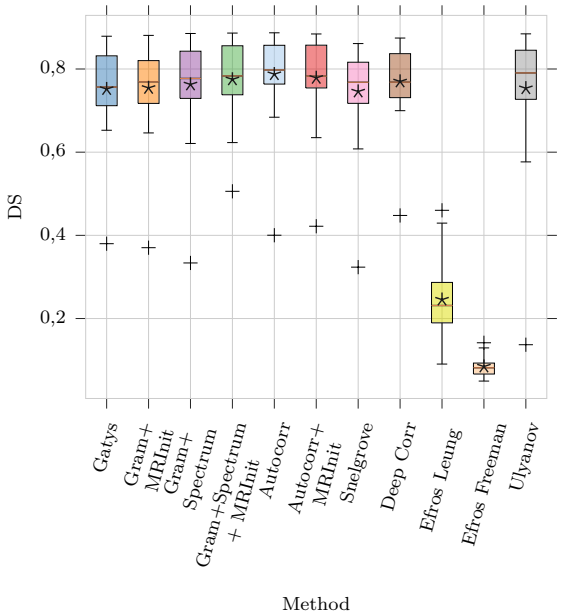


Figure 3.9: Boxplots of the displacement score for the different methods on the twenty reference images of size 1024×1024. For each box, the horizontal orange line corresponds to the average result and the star to the median. The crosses are outliers, i.e. points outside 1.5 times the interquartile range.

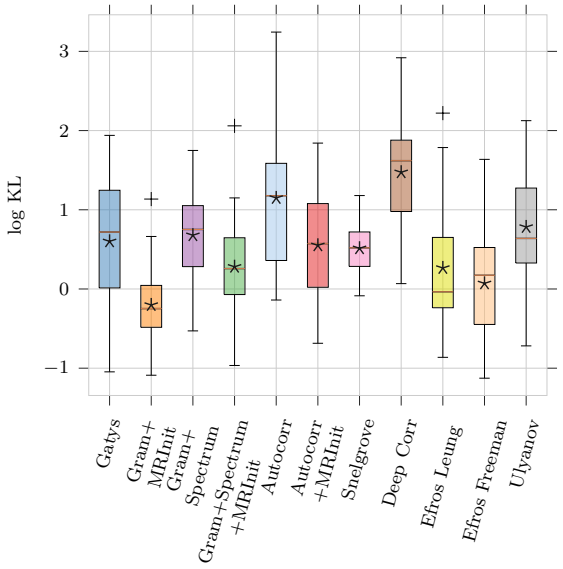


Figure 3.10: Boxplots of the log KL scores for the different methods on the twenty reference images of size 1024×1024. For each box, the horizontal orange line corresponds to the average result and the star to the median. The crosses are outliers (i.e. points outside 1.5 times the interquartile range).

3.3.3.3 Perceptual Evaluation of Texture Synthesis Methods

Then, we also evaluate the proposed methodology through a perceptual user study. Indeed, it is shown in [Clarke et al. 2011; Dong et al. 2013; Dong et al. 2020] through extensive experiments that feature-based evaluations such as the one of the previous section do not approach well human-based visual evaluation of texture similarity, especially in the case of long-range correlations, which is precisely one of the cases tackled in this paper. We therefore rely on a user study to compare the framework we propose to both the original method from Gatys et al. [2015a] and some of its improvement that focus on the respect of large scale structures [Sendik et al. 2017; Snelgrove 2017]. For ethical reasons, we decided not to rely on micro-work platforms. Most users involved are volunteer PhD students or researchers, which certainly induces some bias. The total number of persons involved was 93, each having the possibility to answer up to 40 questions.

Methodology Each question aims at comparing two methods on a given texture. In order to evaluate results at different scales, both the complete synthesis and a detail are presented to the user, see Figure 3.11. The evaluation is performed on the twenty 1024×1024 images considered in this paper. In order to get further insight on the methods, we have split the textures in two groups: regular and irregular, see Figure 3.7. Following the results of the previous sections, we chose to include in the study the five following methods: "Gatys" [Gatys et al. 2015b], "Gram+MRInit", "Gram+Spectrum+MRInit", "Snelgrove" [Snelgrove 2017] and "Deep Coor" [Sendik et al. 2017]. The first four correspond to the best feature-based score and visual impression. The last one appears to us as the most directly related to the present work in the literature, since it explicitly aims at preserving large scale structures through additional statistical constraints. For each couple of methods (out of five) and each image, we build up two setups corresponding to the two possible respective positions of methods (right and left) to avoid a possible lateral bias. This results in 400 different questions, for which we got 3170 answers. For each question, four images are presented corresponding to the two methods at two different scales (global and local). There are 4 possible answers (method 1 is the best for the global and the local scale, method 1 is the best for the local scale and method 2 the best for the global scale, etc.). Even though this is presented as a single question to the user, we treat its answer as two answers, one for the local scale and one for the global scale. This survey has been made with PsyToolkit servers [Stoet 2010; Stoet 2017]. Some screenshots of the online website used for the perceptual survey can be found in Annex, Section A.3. It should be noted that asking a question such as "which result is most similar to the reference" is not trivial. Users were indicated that by "the most similar", it should be understood "which gives the most similar visual impression". Images are not expected to correspond pixel by pixel. Ideally, a synthesized image should give the impression to correspond to a different region of the same material as the reference.

Bradley-Terry model In order to quantify the results of this study, we rely on the Bradley-Terry model [Bradley et al. 1952], as used in other perceptual study, see [Um et al. 2019]. Let $\beta_i \in \mathbb{R}$ represent the strength of method i (also called performance score), and let the outcome of a duel between methods i and j be determined by $\beta_i - \beta_j$. The

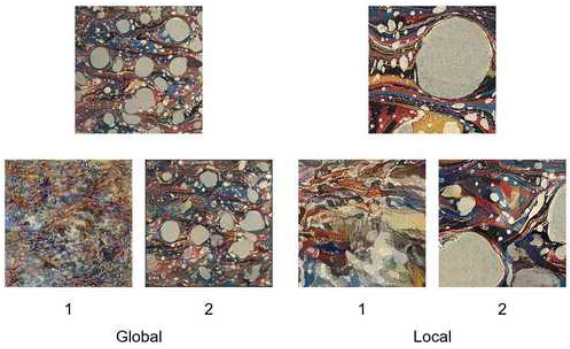


Figure 3.11: Example of the layout for one question.

Bradley-Terry model treats these outcomes as independent Bernoulli random variables with parameter p_{ij}, where the log-odds corresponding to the probability p_{ij} that method i beats method j is modeled as:

$$\log \frac{p_{ij}}{1 - p_{ij}} = \beta_i - \beta_j. \tag{3.11}$$

Equivalently, solving for p_{ij} yields

$$p_{ij} = \frac{e^{\beta_i - \beta_j}}{1 + e^{\beta_i - \beta_j}} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}}. \tag{3.12}$$

This model is over-parameterized in the sense that it is exactly the same if we add a fixed constant to all values. The Bradley-Terry model assigns scores to a fixed set of items based on pairwise comparisons of these items, where the log-odds of item "beating" item is given by the difference of their scores. The strength is estimated by second order optimization of the maximum likelihood and the standard deviation of the difference is approximated with the Hessian of this likelihood (more details about it can be found in Annex, Section A.4.1).

Duel results First, we can consider all the duels between all pairs of methods and all reference images, either from the complete set (20 images) or from the subsets of regular and irregular images separately. Results can be averaged for the global and local scale or treated separately. The results can be found in Tables 3.1 to 3.3. Overall, the two best methods for this evaluation appear to be "Gram+MRInit" and "Gram+Spectrum+MRInit". For the global scale, there is a draw for the complete dataset and for the irregular images, while "Gram+Spectrum+MRInit" wins on the regular images. For the local scale, "Gram+MInit" always win. From this, we may deduce that the spectrum constraint may be useful for preserving large scale structure of regular textures, possibly at the price of a slight degradation at a more local scale. For more irregular textures, method "Gram+MRInit" should be preferred. When we consider all images and both scales (Table 3.1) we can extract a full ranking: "Gram+MRInit" > "Gram+Spectrum+MRInit" > Snelgrov > Gatys > Deep Cor.

Winning probability An alternative evaluation consists in calculating the probability that a method i is chosen among all candidates. This "winning probability" is given

Global case

All images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-2.78e+00 (2.24e-01)	-2.68e+00 (2.21e-01)	-2.17e+00 (2.11e-01)	4.00e-02 (1.86e-01)
Gram + MRInit	2.78e+00 (2.24e-01)		1.07e-01 (1.66e-01)	6.12e-01 (1.70e-01)	2.82e+00 (2.26e-01)
Gram + Spectrum + MRInit	2.68e+00 (2.21e-01)	-1.07e-01 (1.66e-01)		5.05e-01 (1.67e-01)	2.72e+00 (2.23e-01)
[Snelgrove 2017]	2.17e+00 (2.11e-01)	-6.12e-01 (1.70e-01)	-5.05e-01 (1.67e-01)		2.21e+00 (2.14e-01)
Deep Corr [Sendik et al. 2017]	-4.00e-02 (1.86e-01)	-2.82e+00 (2.26e-01)	-2.72e+00 (2.23e-01)	-2.21e+00 (2.14e-01)	

Regular images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-2.67e+00 (3.19e-01)	-3.54e+00 (3.50e-01)	-1.93e+00 (2.95e-01)	-2.17e-01 (2.62e-01)
Gram + MRInit	2.67e+00 (3.19e-01)		-8.73e-01 (2.61e-01)	7.35e-01 (2.49e-01)	2.45e+00 (3.09e-01)
Gram + Spectrum + MRInit	3.54e+00 (3.50e-01)	8.73e-01 (2.61e-01)		1.61e+00 (2.81e-01)	3.33e+00 (3.40e-01)
[Snelgrove 2017]	1.93e+00 (2.95e-01)	-7.35e-01 (2.49e-01)	-1.61e+00 (2.81e-01)		1.72e+00 (2.85e-01)
Deep Corr [Sendik et al. 2017]	2.17e-01 (2.62e-01)	-2.45e+00 (3.09e-01)	-3.33e+00 (3.40e-01)	-1.72e+00 (2.85e-01)	

Irregular images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-2.78e+00 (2.24e-01)	-2.68e+00 (2.21e-01)	-2.17e+00 (2.11e-01)	4.00e-02 (1.86e-01)
Gram + MRInit	2.78e+00 (2.24e-01)		1.07e-01 (1.66e-01)	6.12e-01 (1.70e-01)	2.82e+00 (2.26e-01)
Gram + Spectrum + MRInit	2.68e+00 (2.21e-01)	-1.07e-01 (1.66e-01)		5.05e-01 (1.67e-01)	2.72e+00 (2.23e-01)
[Snelgrove 2017]	2.17e+00 (2.11e-01)	-6.12e-01 (1.70e-01)	-5.05e-01 (1.67e-01)		2.21e+00 (2.14e-01)
Deep Corr [Sendik et al. 2017]	-4.00e-02 (1.86e-01)	-2.82e+00 (2.26e-01)	-2.72e+00 (2.23e-01)	-2.21e+00 (2.14e-01)	

Table 3.1: Difference between the methods strengths $(\beta_i - \beta_j)$ (equation (3.11)) Index i corresponds to rows and index j to columns. When $|\beta_i - \beta_j| > 1{,}96\hat{\text{se}}_{ij}$ the method i is considered as beating the method j and the text is displayed in green. In the opposite case, the text is red. When the text is black, the difference is not significant.

Local case

All images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-1.62e+00 (1.77e-01)	-9.65e-01 (1.65e-01)	-1.02e+00 (1.64e-01)	1.45e+00 (2.04e-01)
Gram + MRInit	1.62e+00 (1.77e-01)		6.59e-01 (1.61e-01)	6.04e-01 (1.63e-01)	3.07e+00 (2.27e-01)
Gram + Spectrum + MRInit	9.65e-01 (1.65e-01)	-6.59e-01 (1.61e-01)		-5.59e-02 (1.54e-01)	2.41e+00 (2.15e-01)
[Snelgrove 2017]	1.02e+00 (1.64e-01)	-6.04e-01 (1.63e-01)	5.59e-02 (1.54e-01)		2.47e+00 (2.16e-01)
Deep Corr [Sendik et al. 2017]	-1.45e+00 (2.04e-01)	-3.07e+00 (2.27e-01)	-2.41e+00 (2.15e-01)	-2.47e+00 (2.16e-01)	

Regular images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-2.31e+00 (2.89e-01)	-1.74e+00 (2.74e-01)	-1.80e+00 (2.74e-01)	8.95e-01 (2.78e-01)
Gram + MRInit	2.31e+00 (2.89e-01)		5.74e-01 (2.33e-01)	5.15e-01 (2.38e-01)	3.21e+00 (3.32e-01)
Gram + Spectrum + MRInit	1.74e+00 (2.74e-01)	-5.74e-01 (2.33e-01)		-5.90e-02 (2.29e-01)	2.64e+00 (3.16e-01)
[Snelgrove 2017]	1.80e+00 (2.74e-01)	-5.15e-01 (2.38e-01)	5.90e-02 (2.29e-01)		2.69e+00 (3.18e-01)
Deep Corr [Sendik et al. 2017]	-8.95e-01 (2.78e-01)	-3.21e+00 (3.32e-01)	-2.64e+00 (3.16e-01)	-2.69e+00 (3.18e-01)	

Irregular images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-1.62e+00 (1.77e-01)	-9.65e-01 (1.65e-01)	-1.02e+00 (1.64e-01)	1.45e+00 (2.04e-01)
Gram + MRInit	1.62e+00 (1.77e-01)		6.59e-01 (1.61e-01)	6.04e-01 (1.63e-01)	3.07e+00 (2.27e-01)
Gram + Spectrum + MRInit	9.65e-01 (1.65e-01)	-6.59e-01 (1.61e-01)		-5.59e-02 (1.54e-01)	2.41e+00 (2.15e-01)
[Snelgrove 2017]	1.02e+00 (1.64e-01)	-6.04e-01 (1.63e-01)	5.59e-02 (1.54e-01)		2.47e+00 (2.16e-01)
Deep Corr [Sendik et al. 2017]	-1.45e+00 (2.04e-01)	-3.07e+00 (2.27e-01)	-2.41e+00 (2.15e-01)	-2.47e+00 (2.16e-01)	

Table 3.2: Difference between the methods strengths $(\beta_i - \beta_j)$ (equation (3.11)) Index i corresponds to rows and index j to columns. When $|\beta_i - \beta_j| > 1{,}96\hat{\text{se}}_{ij}$ the method i is considered as beating the method j and the text is displayed in green. In the opposite case, the text is red. When the text is black, the difference is not significant.

Global and local case

All images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-2.10e+00 (1.36e-01)	-1.70e+00 (1.30e-01)	-1.50e+00 (1.26e-01)	7.48e-01 (1.31e-01)
Gram + MRInit	2.10e+00 (1.36e-01)		3.93e-01 (1.14e-01)	5.98e-01 (1.17e-01)	2.85e+00 (1.53e-01)
Gram + Spectrum + MRInit	1.70e+00 (1.30e-01)	-3.93e-01 (1.14e-01)		2.05e-01 (1.12e-01)	2.45e+00 (1.47e-01)
[Snelgrove 2017]	1.50e+00 (1.26e-01)	-5.98e-01 (1.17e-01)	-2.05e-01 (1.12e-01)		2.25e+00 (1.45e-01)
Deep Corr [Sendik et al. 2017]	-7.48e-01 (1.31e-01)	-2.85e+00 (1.53e-01)	-2.45e+00 (1.47e-01)	-2.25e+00 (1.45e-01)	

Regular images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-2.42e+00 (2.10e-01)	-2.50e+00 (2.12e-01)	-1.82e+00 (1.98e-01)	3.27e-01 (1.84e-01)
Gram + MRInit	2.42e+00 (2.10e-01)		-8.77e-02 (1.64e-01)	5.94e-01 (1.68e-01)	2.74e+00 (2.19e-01)
Gram + Spectrum + MRInit	2.50e+00 (2.12e-01)	8.77e-02 (1.64e-01)		6.82e-01 (1.68e-01)	2.83e+00 (2.20e-01)
[Snelgrove 2017]	1.82e+00 (1.98e-01)	-5.94e-01 (1.68e-01)	-6.82e-01 (1.68e-01)		2.15e+00 (2.08e-01)
Deep Corr [Sendik et al. 2017]	-3.27e-01 (1.84e-01)	-2.74e+00 (2.19e-01)	-2.83e+00 (2.20e-01)	-2.15e+00 (2.08e-01)	

Irregular images

	[Gatys et al. 2015b]	Gram + MRInit	Gram + Spectrum + MRInit	[Snelgrove 2017]	Deep Corr [Sendik et al. 2017]
[Gatys et al. 2015b]		-2.10e+00 (1.36e-01)	-1.70e+00 (1.30e-01)	-1.50e+00 (1.26e-01)	7.48e-01 (1.31e-01)
Gram + MRInit	2.10e+00 (1.36e-01)		3.93e-01 (1.14e-01)	5.98e-01 (1.17e-01)	2.85e+00 (1.53e-01)
Gram + Spectrum + MRInit	1.70e+00 (1.30e-01)	-3.93e-01 (1.14e-01)		2.05e-01 (1.12e-01)	2.45e+00 (1.47e-01)
[Snelgrove 2017]	1.50e+00 (1.26e-01)	-5.98e-01 (1.17e-01)	-2.05e-01 (1.12e-01)		2.25e+00 (1.45e-01)
Deep Corr [Sendik et al. 2017]	-7.48e-01 (1.31e-01)	-2.85e+00 (1.53e-01)	-2.45e+00 (1.47e-01)	-2.25e+00 (1.45e-01)	

Table 3.3: Difference between the methods strengths $(\beta_i - \beta_j)$ (equation (3.11)) Index i corresponds to rows and index j to columns. When $|\beta_i - \beta_j| > 1{,}96\hat{\text{se}}_{ij}$ the method i is considered as beating the method j and the text is displayed in green. In the opposite case, the text is red. When the text is black, the difference is not significant.

by the average over j of the probability p_{ij} that a participant chooses the candidate i over j:

$$W_i = \frac{1}{N-1}\sum_{j\neq i}^{N} p_{ij} = \frac{1}{N-1}\sum_{j\neq i}^{N} \frac{e^{\beta_i-\beta_j}}{1+e^{\beta_i-\beta_j}} \tag{3.13}$$

In contrast to the pairwise probability p_{ij}, W_i represents the probability that a candidate i was preferred over all other candidates. We can estimate the standard error of W_i as:

$$\Sigma_i = \frac{1}{N-1}\sqrt{\sum_{j\neq i}^{N} \hat{\sigma}_{ij}^2}. \tag{3.14}$$

under the hypothesis that the p_{ij} are independent. These winning probabilities are displayed in Figures 3.12 to 3.14 and confirm the duel results.

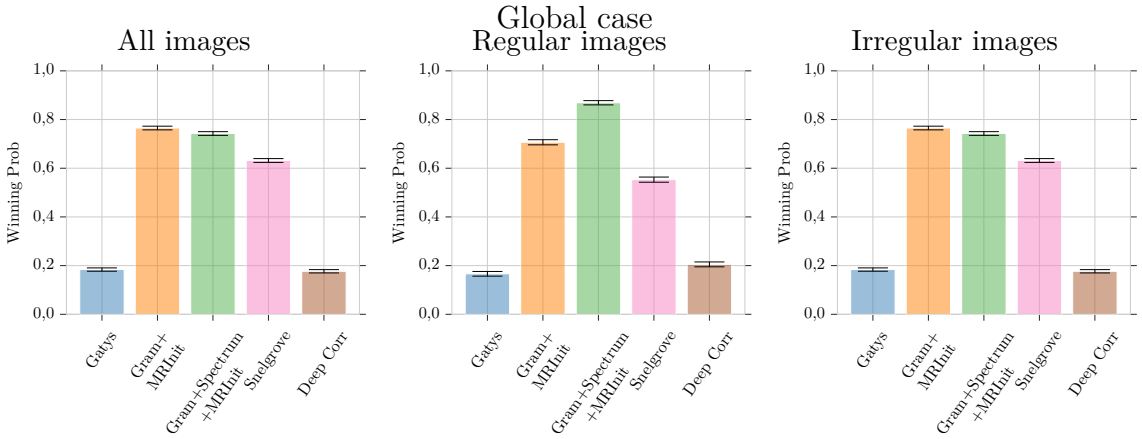


Figure 3.12: Winning probabilities W_i with standard error Σ_i for the different methods for the global case.

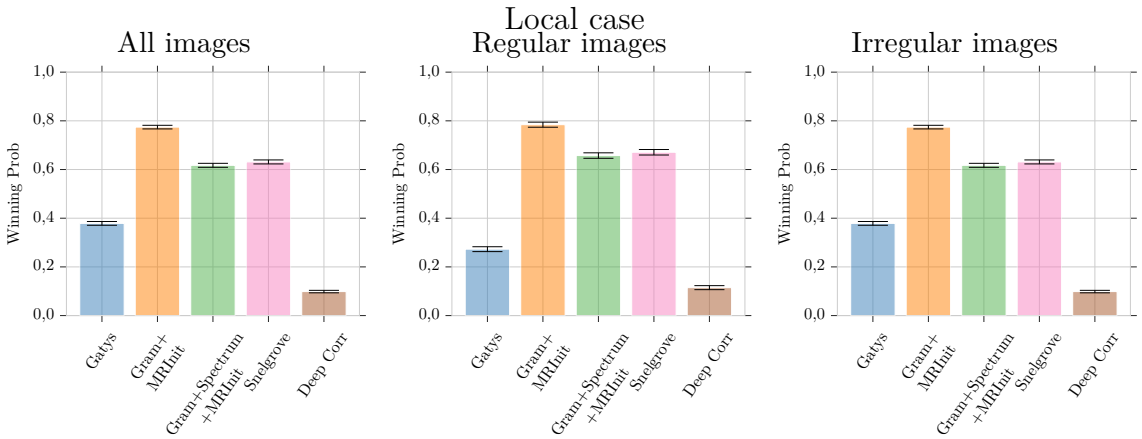


Figure 3.13: Winning probabilities W_i with standard error Σ_i for the different methods for the local case.

Another possibility is to consider each reference image as an individual study and to compute a winning probability over the whole set of reference images along with a near-convergence consistency metric as in [Um et al. 2019]. This exercise is available in Annex, Section A.4.2.

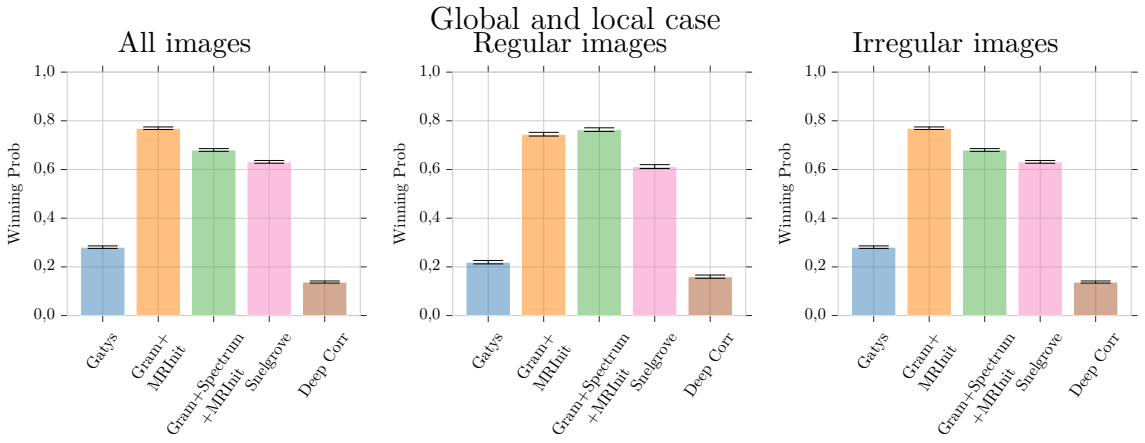


Figure 3.14: Winning probabilities W_i with standard error Σ_i for the different methods for both global and local cases.

3.3.4 Influence of Parameters

In this section, we display experiments illustrating the effects of two parameters of the proposed method: K, the number of considered scales, and β, the weighting of the spectrum term when using the method "Gram+Spectrum+MRInit".

3.3.4.1 Multi-resolution Strategy

In Figure 3.15, we display synthesis results with K ranging from 0 (original method from Gatys et al. [2015b]) to 4. The quality of results increases up to $K = 2$. This confirms the fact that the size of the filters in the VGG19 network is too small to describe large scales. Its also illustrates the fact than the VGG filters are versatile and provide good features at different scales, since the network has been trained on 224×224 input images. From $K = 3$, the method starts to produce results that are very similar to the reference, the case $K = 4$ being almost a copy of the reference. This may be due to the fact that in these cases, the number of parameters of the synthesis model is up to two orders of magnitude larger than the number of pixels of the coarse image. In other words, the multi-resolution strategy reduces too much the solution space for this optimization problem. In practice, $K = 2$ appears a good choice for synthesizing 1024×1024 images.

3.3.4.2 Weighting of the Spectrum Constraint

In Figure 3.16, we display the result of the synthesis for different values of β, the parameter weighting the spectrum constraint, using the method "Gram+Spectrum+ MRInit". For the structured textures for which the spectrum term is useful, the best results are obtained for a relatively large β, of the order of 10^5 for the brick image (second column). For more irregular textures, such high values may deteriorate results. This is in agreement with the results from the previous evaluations, where a value $\beta = 10^5$ was used. The problem of automatically setting this parameter is open. For instance, in [Risser et al. 2017], it is proposed to clip the gradient of a given term of the loss if the amplitude of this gradient is bigger than a threshold. Nevertheless, the optimization would be even better if one can dynamically tune the weighting of the

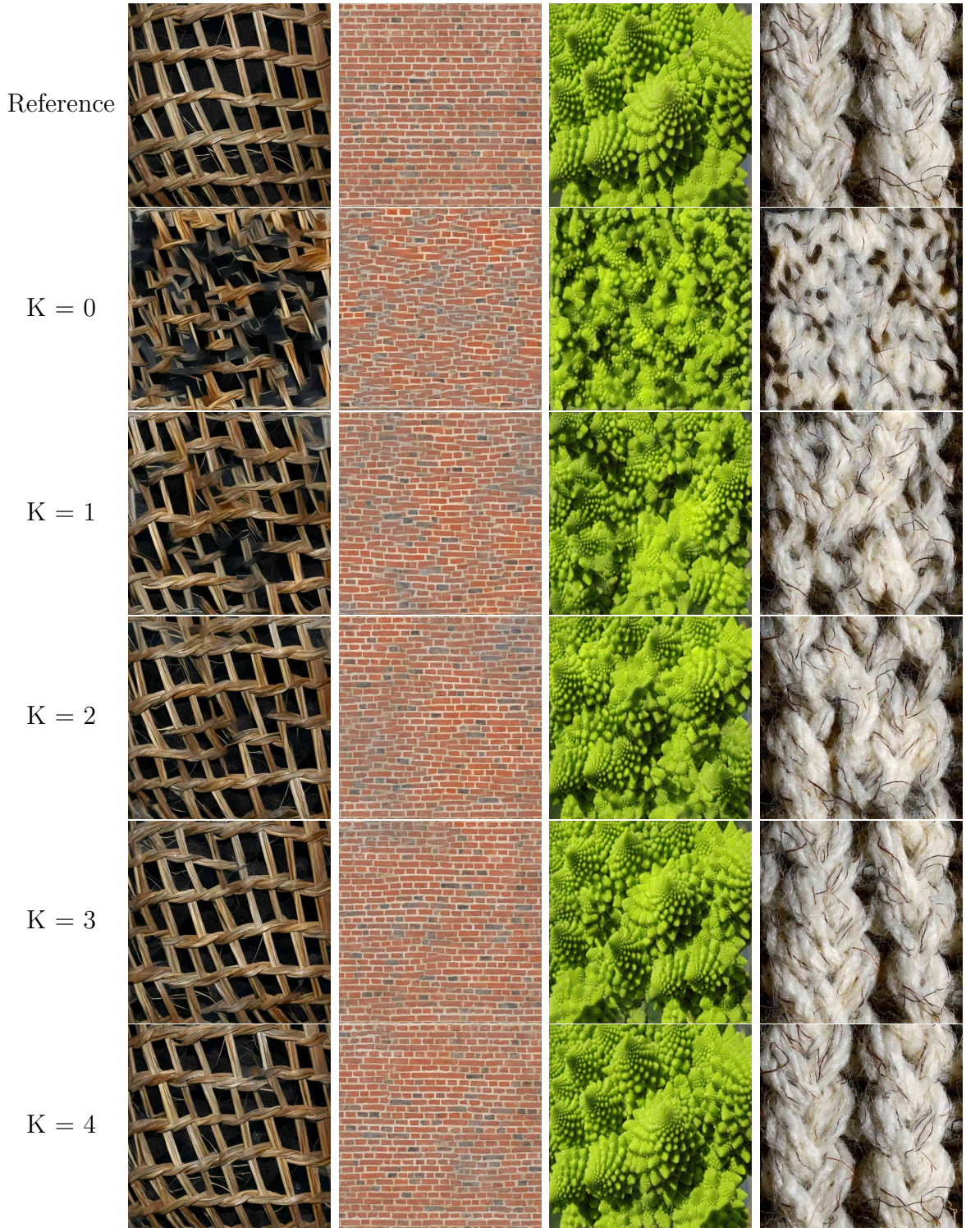


Figure 3.15: Synthesis results using a different number of scale K in the multi-resolution strategy. $K = 0$ correspond to the original method from Gatys et al. [2015b].

different loss's terms based on non-gradient information such as the magnitude of the losses or statistics from the reference image.

In Section A.5 of the annex, synthesis with different values for β without the multi-resolution strategies can be found. The weighting value still influences a lot the result.

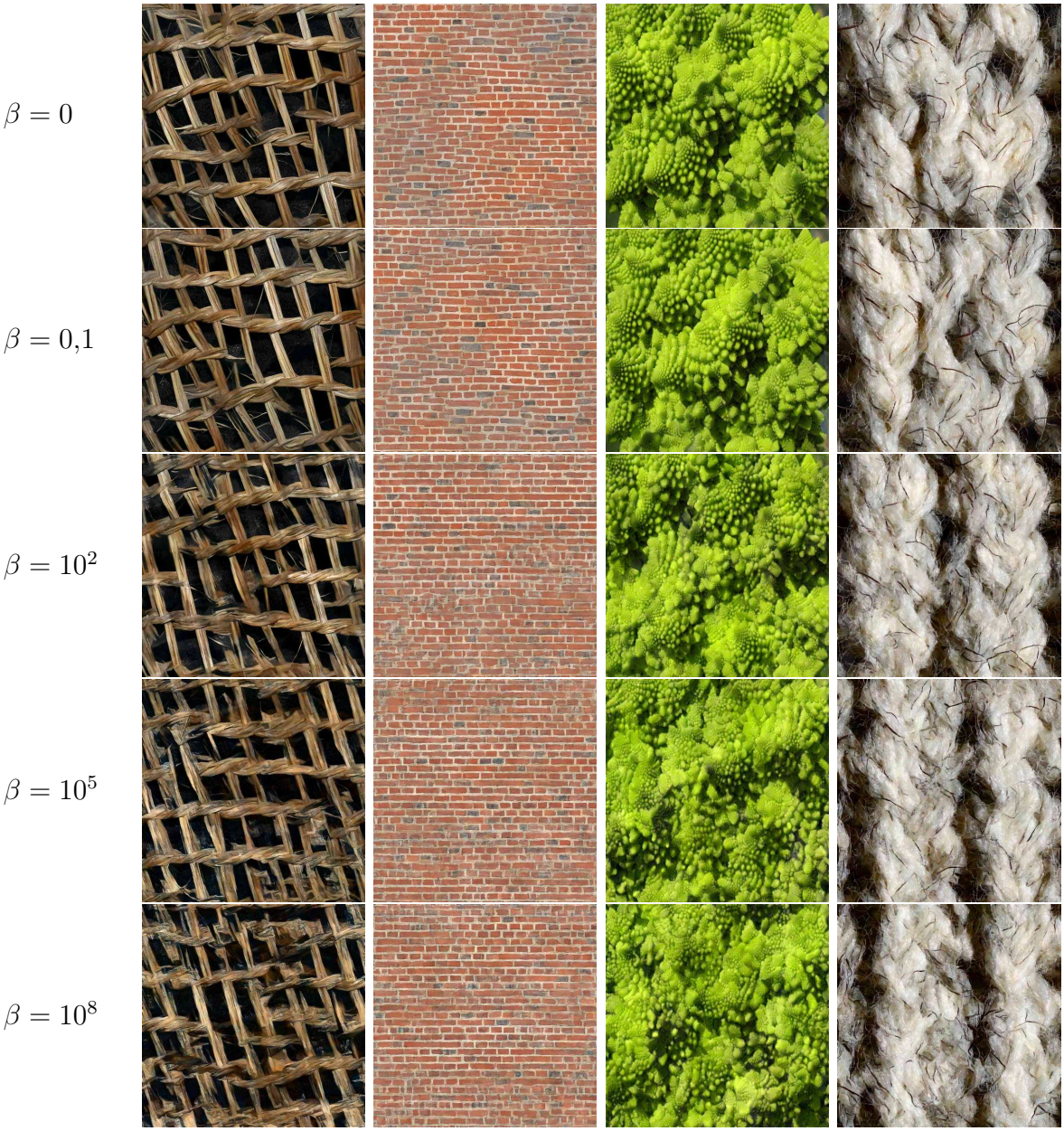


Figure 3.16: Synthesis results using different β in equation (3.4) (original can be seen in Figure 3.15), $\beta = 0, 10^{-1}, 10^2, 10^5, 10^8$ with the multi-resolution strategy and $K = 2$.

3.3.5 Higher Resolution Synthesis

We conclude this experimental section by showing synthesis results of higher resolution (2048×2048). We consider methods "Gatys", "Gram+MRInit", "Gram+Spectrum+MRInit" (both using $K = 3$) and "Snelgrove" [Snelgrove 2017] (with one more octave than previously). The results can be seen in Figure 3.17. More results can be seen in Annex (Section A.6). Unsurprisingly the interest of the multi-resolution schemes is even stronger in this case and the mono-scale method fails. Figure 3.17b shows the ability of the spectrum constraint to enforce large scale regularity at this resolution.

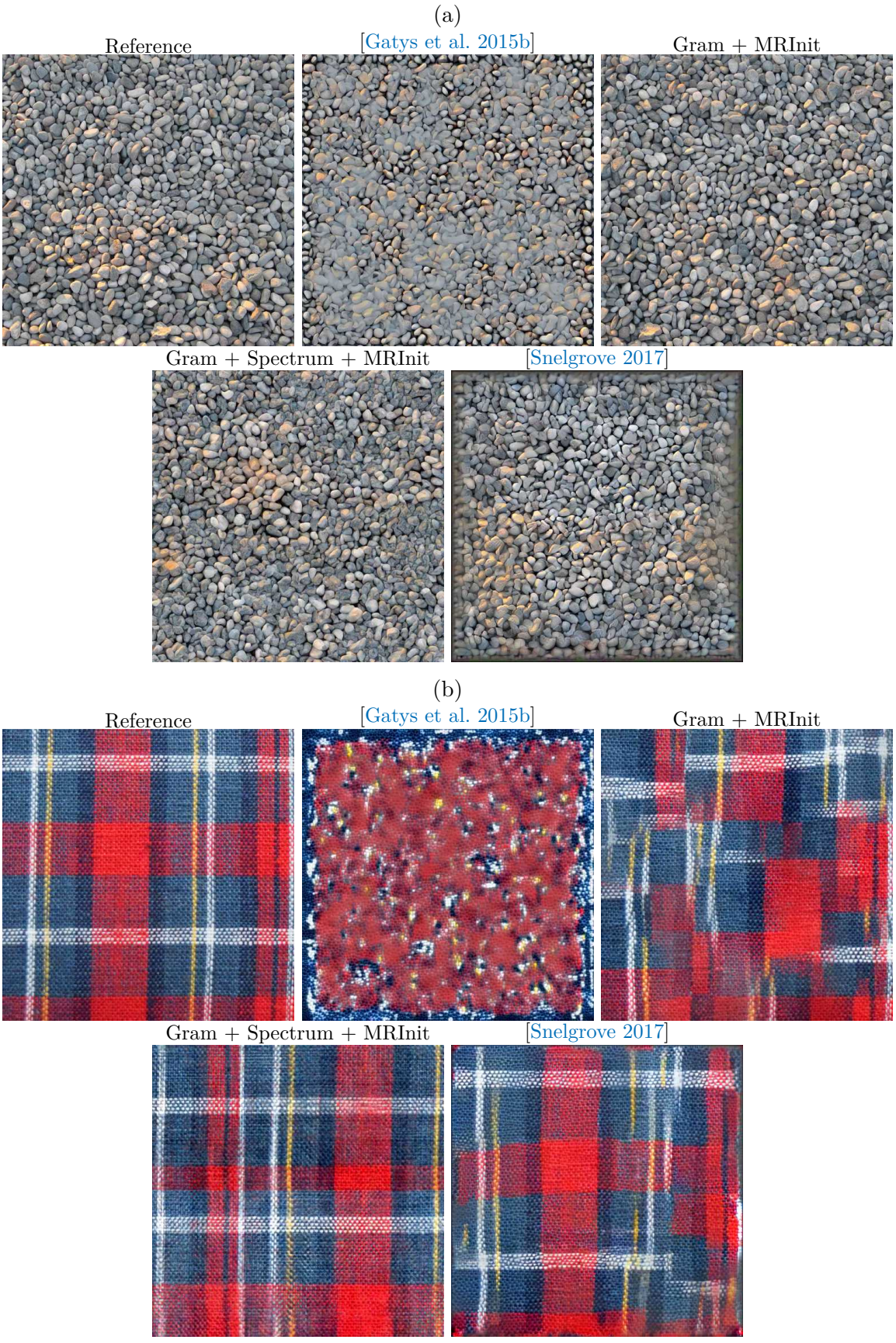


Figure 3.17: Synthesis results using different methods for two given references of size 2048×2048.

3.4 Unsuccessful Attempts

This section brings together unsuccessful experiences about using other kinds of statistics for texture synthesis with CNNs and therefore may serve as a basis for future research. We tried to impose simpler statistics than the Gram matrices such as moments or norms but also more complex ones such as the spectrum of the whole feature maps.

3.4.1 Using Other Statistics for Texture Synthesis

3.4.1.1 Moments and Norms of the Feature Maps

As mentioned previously, numerous works tend to find the minimal set of statistics needed to correctly describe texture images. Some works have shown that second-order statistics between features are not necessary to get satisfying results. De Bortoli et al. [2019] successfully use the mean of the feature maps. Observe that the same statistics do not provide good results for the original method [Gatys et al. 2015b, Figure 3]. This, combined with the highly redundant nature of networks such as VGG19, trained for recognition, suggests that much room is available to reduce the number of parameters in these models. One solution is to consider the J first standardized moments of the feature maps. Thus for the feature map p of the layer l we can define the first J standardized moments in the following way:

$$\mu_j(f_p^l) = \begin{cases} \frac{1}{n_l}\sum_{i=1}^{n_l} f_p^l(i) & \text{if } j=1 \\ \frac{1}{n_l}\sum_{i=1}^{n_l}\left(f_p^l(i)-\mu_1(f_p^l)\right)^2 & \text{if } j=2 \\ \frac{1}{n_l}\sum_{i=1}^{n_l}\left(\frac{f_p^l(i)-\mu_1(f_p^l)}{\sqrt{\mu_2(f_p^l)}}\right)^k & \text{otherwise} \end{cases}. \tag{3.15}$$

The considered statistics is then:

$$\Phi_l(f^l) = \begin{pmatrix} \mu_1(f_1^l) & \cdots & \mu_1(f_{m_l}^l) \\ \vdots & \ddots & \vdots \\ \mu_J(f_1^l) & \cdots & \mu_J(f_{m_l}^l) \end{pmatrix}. \tag{3.16}$$

Nevertheless, some channels may be completely equal to zero after ReLU activation. This leads to a variance equals to zero and then the calculus of the standardized moment of order $J > 2$ leads to a division by zero. To face this problem, we have also considered another type of first-order statistics of the feature map: the first Q p-norms. This is defined as follows:

$$\Phi_l(f^l) = \begin{pmatrix} \|f_1^l\|_1 & \cdots & \|f_{m_l}^l\|_1 \\ \vdots & \ddots & \vdots \\ \|f_1^l\|_Q & \cdots & \|f_{m_l}^l\|_Q \end{pmatrix}, \tag{3.17}$$

with $\|f_q^l\|_p = \left(\frac{1}{n_l}\sum_{i=1}^{n_l} \mid f_q^l \mid^p\right)^{1/p}$, the classical p-norm. In both cases, we consider that all the terms of the Φ_l operator have the same dynamic even if it not the case.

3.4.1.2 Random Phase Feature Maps

As the use of the autocorrelation do not impose the covariances between feature maps, we also propose a more constrained statistic. We take inspiration from the Random Phase Noise algorithm [Galerne et al. 2011], where a new texture image is synthesized by adding a random phase θ to the Fourier phase of the image. In our case, we considered the feature map as a multi-channel image and we add the same random phase to the Fourier transform of each channel of features map f^l. This is done only for the last layer L of the CNN considered for the synthesis because otherwise it is too constrained. The random phase version of f^L is equal to:

$$\hat{f}_p^L = \mathcal{F}^{-1}(\mathcal{F}(f_p^L)\Theta_L), \tag{3.18}$$

for each channel p. It is a term product. With $\Theta_L(x,y) = e^{i\theta(x,y)}$ for $x \in 1, \cdots, h_L$ and $y \in 1, \cdots, w_L$ with $\theta(x,y) \sim \mathcal{U}]-\pi, \pi]$ such that Θ_L has an hermitian symmetry[11]. The corresponding loss function is:

$$\mathcal{L}_{Random\ \ Phase,L} = \frac{1}{2}\|\hat{f}^L - \tilde{f}^L\|_{\mathcal{F}}^2. \tag{3.19}$$

In the experimental part, we propose to use this operator for the deepest layer and the autocorrelation for the other ones to also respect low-level patterns of the reference image. Then the complete loss function becomes:

$$\mathcal{L}_{Random\ \ Phase+Autocorr,\mathcal{S}} = \sum_{l=1}^{L-1} \alpha_l \|\Phi_l(f^l) - \Phi_l(\tilde{f}^l)\|_{\mathcal{F}}^2 + \alpha_L \mathcal{L}_{Random\ \ Phase,I,L}(\tilde{I}), \tag{3.20}$$

with the autocorrelation Φ_l defined in equation (3.9).

3.4.2 Preliminary Experimental Results

In this section, we perform experiments to illustrate the different statistics that can be used instead of Gram matrices for texture synthesis with CNN with low resolution images (around 256×256). We can see the results for three different reference images in Figures 3.18 to 3.20 and for more reference images in the Annex Section A.7.

First, we can confirm that using only the mean of the feature maps with the original Gatys et al.'s optimization process [2015b] yields to worst results than using the Gram matrices, as it has already been shown. Including the variance (case "Moments J=2") provides better results whereas using the variance alone leads to halo artifacts (Figure 3.19). This halo effect can also be found in the case of the Gram matrices as mentioned by Risser et al. [2017]. When we consider the first 4 moments (J=4), the algorithm completely fails. It may be due to a different in the range of the skewness or kurtosis compared to the mean or variance. On the other hand, the method based on the first 4 p-norms enables to have better synthesis, especially for the stochastic case, in Figure 3.18. Then, the random phase feature maps method fails to provide adequate synthesis. Indeed, this method only considers the last layer without any control on small patches. Combining random phase feature maps on the last layer with

[11] Experimentally, Θ_L is obtained by taking the phase of a Gaussian noise image.

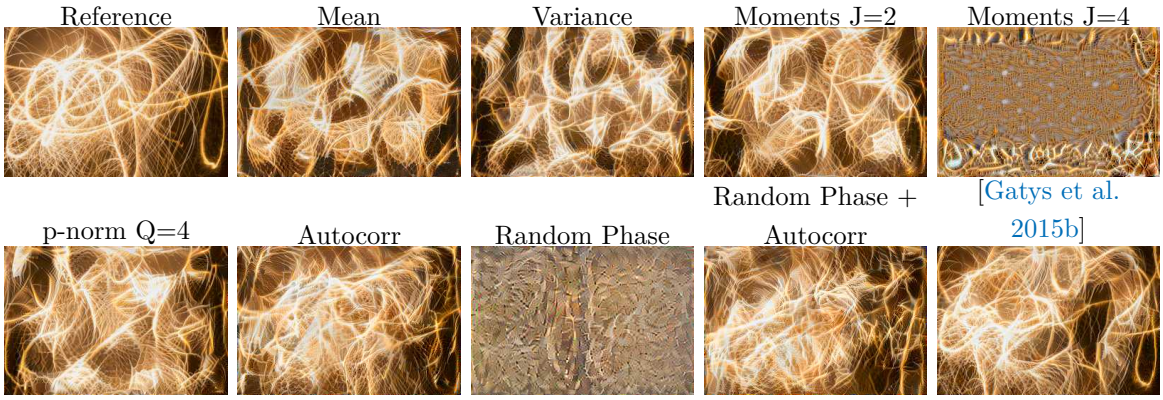


Figure 3.18: Synthesis results using different methods for a given reference of size 209×314.

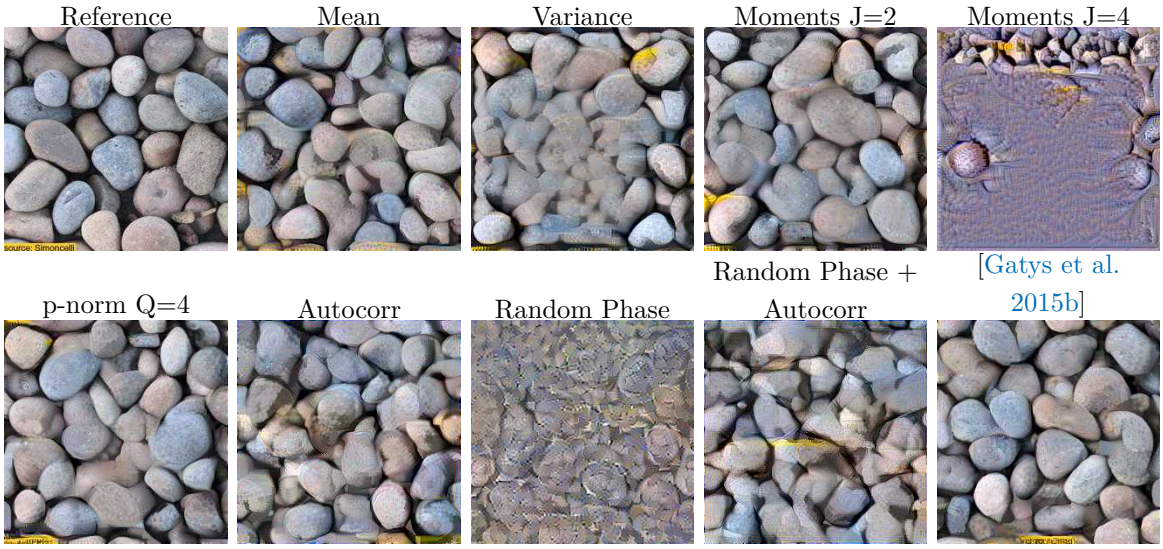


Figure 3.19: Synthesis results using different methods for a given reference of size 256×256.

autocorrelation on the others yields to excellent regular synthesis (see Figure 3.20). Nevertheless, this method is not consistent. It may provide excellent output as bad ones. It may exist an optimal value for α_L (equation (3.20)) that enables to have a more reliable method.

The last thing we can point out is that the autocorrelation method synthesizes images with more artifacts than the Gram based one. These kind of artifacts are frequent in image synthesis with CNNs. There is no convincing indication of their origin although a potential explanation is given by Odena et al. [2016]. Total variation [Johnson et al. 2016] or spectrum constraints [Liu et al. 2016a] may be a solution to control them. Indeed, the spectrum analysis is an efficient way to find them [Zhang et al. 2019b]. This may be due to the fact that CNNs are focusing on high frequency during training as mention before in the last paragraph of Section 2.2.1.1 about the Jo et al.'s work [2017].

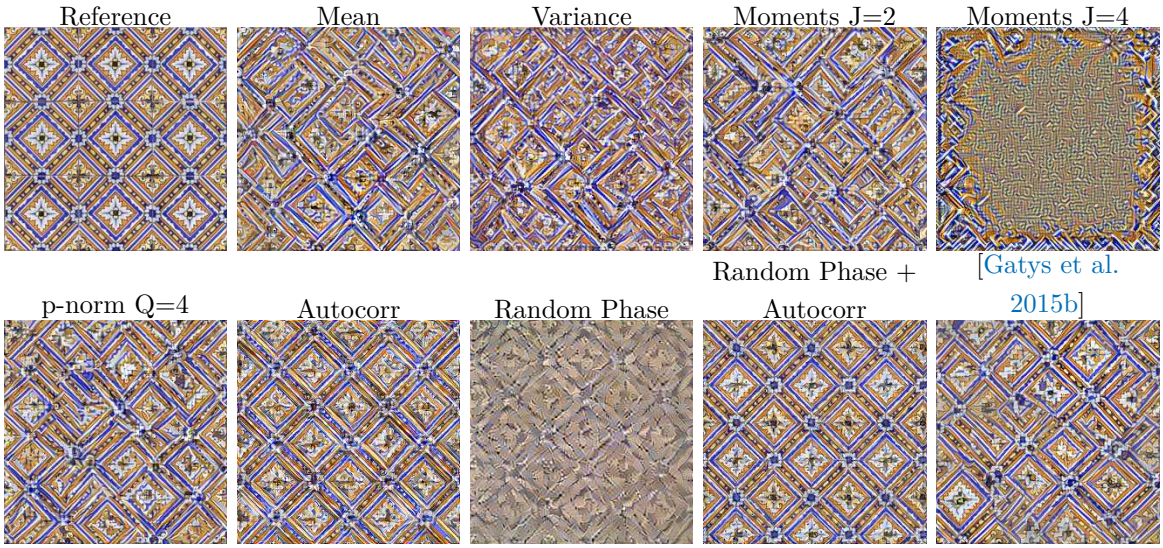


Figure 3.20: Synthesis results using different methods for a given reference of size 256×256.

High resolution synthesis can be found in Annex (Section A.8) for the promising "Random Phase + Autocorr" method, with or without the multi-resolution strategy.

3.4.3 Number of parameters of the different texture synthesis methods

One important metric in texture synthesis is the number of parameters used to represent the image. It provides an idea of the balance between fidelity and the amount of information needed. The approximate formula[12] of the number of parameters (and a numerical evaluation of it) for the different methods proposed in this manuscript are grouped into Table 3.4. The number of parameters for the approaches based on Gram matrices, moments or norms of the feature maps only depend on the set of layers considered whereas the methods based on the spectrum and/or the autocorrelation also depend on the size of the image. The number of parameters increases linearly with the number of scales for our multi-resolution strategy but quadratically with the size of the image with the use of a spectral or an autocorrelation constraint. The use of moments or norms instead of the Gram matrices reduces 100 times the number of parameters whereas using the autocorrelation increases it 20 times. In the multi-resolution scheme, the autocorrelation approach has 120 times more parameters than the Gram matrices one. The solution using the autocorrelation is the most costly in terms of the number of parameters.

The use of the spectrum or the autocorrelation does not seem to experimentally causes an important computational overheat compared to the feed-forward inference and the back-propagated gradient computation required by the method. Nevertheless

[12]Indeed, in the formulas of the autocorrelation, spectrum or random phase, the terms $(h_l * w_l)/2$ and $N = h * w/2$ can be refined because few terms of the discrete Fourier transform are real. If w_l and h_l are even, we can subtract 4 to the formula at the layer l. If we consider 256×256 or 1024×1024 images then w_l and h_l are even.

Method	Approx Formula / Remark	Number of parameters	Quadratic in the size of the image
Monoscale $256 * 256$			
[Gatys et al. 2015b] - Gram	$\sum_{l=1}^{5} m_l^2/2$	177k	No
Mean	$\sum_{l=1}^{5} m_l$	1024	No
Variance	$\sum_{l=1}^{5} m_l$	1024	No
Moments ($J = 2$)	$\sum_{l=1}^{5} J * m_l$	2048	No
Moments ($J = 4$)	$\sum_{l=1}^{5} J * m_l$	4096	No
p-norm ($Q = 4$)	$\sum_{l=1}^{5} Q * m_l$	4096	No
Autocorrelation	$\sum_{l=1}^{5} m_l * (h_l * w_l)/2$	3080k	Yes
Gram + Spectrum	$\sum_{l=1}^{5} m_l^2/2 + N/2$	209k	Yes
Random Phase	$m_L * (w_L * h_L)/2$	33k	Yes
Autocorrelation + Random Phase	$\sum_{l=1}^{5} m_l * (h_l * w_l)/2$	3080k	Yes
Multi-scale $1024 * 1024$			
Gram + MRInit ($K = 2$)	$K * \sum_{l=1}^{5} m_l^2/2$	531k	No
Gram + Spectrum + MRInit ($K = 2$)	$K * \sum_{l=1}^{5} m_l^2/2 + N/2$	1219k	Yes
Autocorr + MRInit ($K = 2$)	$\sum_{k=0}^{K} \sum_{l=1}^{5} m_l * (h_l * w_l)/2^{k+1}$	64684k	Yes
Autocorr + Random Phase + MRInit ($K = 2$)	$\sum_{k=0}^{K} \sum_{l=1}^{5} m_l * (h_l * w_l)/2^{k+1}$	64684k	Yes
Other methods			
[Portilla et al. 2000]	See [Raad et al. 2018]	710	No
[Snelgrove 2017] ($K = 3$)	They consider (pool1, block3_2)	278k	No
[De Bortoli et al. 2019] Mean of f_l	They consider 10 layers of VGG19	2688	No

Table 3.4: Numbers of parameters for different textures synthesis methods. We consider the default set of layers from [Gatys et al. 2015b]: (conv1_1 , pool1-4), then we have[a] $m_1 = m_2 = 64, m_3 = 128, m_4 = 256, m_5 = 512$ and $L = 5$ the last layer considered. m_l is the number of channel at the layer l whereas h_l and w_l are the spatial dimensions. We consider an image of size $256 * 256$ in monoscale and an image of size $1024 * 1024$ in multiscale. K is the number of scale.

[a]The number of channel per layer can be found in annex (Table D.3).

the "Gram + Spectrum" and "Autocorr" methods are based on a huge number of parameters which can cause synthesis identical to the reference.

3.5 Conclusion

In this chapter, we have shown how a multi-resolution framework and additional statistical constraints related to long-range dependency enable one to significantly improve texture synthesis results in comparison to the seminal Gatys et al.'s work [2015b], es-

pecially for high resolution and possibly regular textures. Gatys et al.'s method can be seen as a special-case of this work for a single scale ($K = 0$). We also propose two metrics to quantitatively evaluate the quality of synthesis, one based on displacement maps and the second one on wavelets coefficients. Nevertheless, these metrics are not enough to determine who will come off best. Thus, we conduct a perceptual study to arrive at a decision between methods. The methodology we can be useful to rank future methods too. Finally, we provide some hints about using statistical alternatives to the Gram matrices.

4

Transfer Learning and Analysis of Convolutional Neural Networks for Art Classification

Contents

Abstract

Transfer learning from huge natural image datasets, and the use of the corresponding pre-trained networks have become de facto the core of art analysis applications. These convolutional neural networks can be used as off-the-shelf features extractors or fine-tuned on the new dataset. Nevertheless, the effects of transfer learning are still poorly understood. In this chapter, we first compare different transfer learning approaches. We will show that a double fine-tuning involving a medium-size artistic dataset can improve the classification on smaller datasets, even when the task changes. Then we use techniques for visualizing the network internal representations for one particular architecture in order to provide clues about what the network has learned on artistic images. Finally, we provide a quantitative analysis of the changes introduced by the learning process thanks to metrics in both the feature and parameter spaces, as well as metrics computed on the set of maximal activation images. These analyses are performed on several variations of the transfer learning procedure. In particular, we observed that the network could specialize some pre-trained filters to the new image modality and also that higher layers tend to concentrate classes.

Parts of the work in this chapter have led to two publications, one in a French conference and one in an international conference:

- Gonthier N., Gousseau Y., Ladjal S. *Transfert d'apprentissage et visualisation de réseaux de neurones pour les images artistiques*; The Measurement of Images. Computational Approaches in the History and Theory of the Arts, DHNord 2020.
- Gonthier N., Gousseau Y., Ladjal S. *An analysis of the transfer learning of convolutional neural networks for artistic images*; Workshop on Fine Art Pattern Extraction and Recognition, ICPR, 2020.

4.1 Introduction

Transfer learning from large-scale natural image datasets [Russakovsky et al. 2015; Kuznetsova et al. 2020], mostly by fine-tuning large pre-trained networks, has become the *de facto* approach for art analysis applications. These datasets allow the learning of powerful deep models that recognize a wide range of visual categories. For instance, the VGG19 network [Simonyan et al. 2015] was trained on the trainval (Training + Validation) set of ILSVRC-2012 - a subset of ImageNet [Russakovsky et al. 2015] - which consists of 1.2 million natural images each annotated with one thousand categories. In the following text, we will mainly use the term *ImageNet* to designate this trainval set of ILSVRC-2012. Nevertheless, there are large differences in dataset sizes, image style and task specifications between natural images and the target artistic images, and there is little understanding of the effects of transfer learning in this context. First, we study classifiers trained on natural images or artistic ones (Section 4.3.1). Then, we compare different ways to obtain an image classifier: usual classifier trained on off-the-shelf features extracted from a pre-trained CNN, fine-tuned CNNs and CNNs trained from scratch (Section 4.3.2). Finally, we use techniques for visualizing the network internal representations in order to provide clues to the understanding of what the network has learned on artistic images (Section 4.4). We provide a quantitative analysis of the changes introduced by the learning process thanks to metrics in both the feature and parameter spaces, as well as metrics (overlapping ratio and entropy) computed on the set of maximal activation images for a given channel . In particular, we will see that CNNs can specify some pre-trained filters in order to adapt them to the new modality of images and they can also learn new, highly structured filters specific to artistic images from scratch. We also look at the set of the maximal activation images for a given channel to complete our observation.

Moreover we experimentally show that fine-tuning first a pretrained ImageNet model on an intermediate artistic dataset can lead to better performances than a direct fine tuning on the target small artistic dataset (for a different task).

Let us emphasize that the goal of this work is not to provide state-of-the-art classification performances, but rather to investigate the way CNNs are modified by classical fine-tuning operations in the context of artwork images.

4.2 Art Datasets

For our transfer learning experiments, we use two datasets which come from different research works. The first one contains the largest number of samples and comes from the WikiArt website. It contains 80,000 images tagged with one among 25 artistic styles [Lecoutre et al. 2017] and is named *RASTA*. Many other works referred to this dataset as the "WikiArt paintings" [Tan et al. 2016] but this variant contains only 25 classes instead of 27. Due to its size and large diversity, we will mainly use this dataset in the feature visualization part, Section 4.4. The second one is the *Paintings* Dataset introduced in [Crowley et al. 2014], made of 8629 British painting images with 10 different labels corresponding to common objects (ex: horse or airplane). These visual categories associated to the images can also be found in natural images especially in the

VOC dataset [Everingham et al. 2010]. It is a subset of the Art UK dataset formally named "Your Paintings".

4.2.1 The IconArt dataset

Although there has been a recent effort to increase open-access databases of artworks by academia and/or museums [Crowley et al. 2014; Mensink et al. 2014; Lecoutre et al. 2017; Mao et al. 2017; Europeana 2018; Rijksmuseum 2018; Strezoski et al. 2018; Zhang et al. 2019a], they usually do not include systematic and reliable keywords related to art history. There are two exceptions of publicly available datasets. The first one is the database from the Rijkmuseum, with labels based on the IconClass classification system, but this database of 112k elements is mostly composed of prints, photographs and drawings [Mensink et al. 2014]. The second one is the recent "Iconclass AI Test Set" [Posthumus 2020] sampled from the Arkyves database. It is a curated dataset of 80k images with IconClass labels too. Nevertheless, these databases don't include the localization of objects or characters that are useful for the detection task.

Therefore, we decided to create our own dataset called *IconArt*. In order to study the ability of deep learning model to detect iconographic elements that are interesting for art historians, we gathered 5955 painting images from Wikicommons[1], dating from the 11th to the 20th century, which are partially annotated by the Wikidata[2] contributors. We manually checked and completed image-level annotations for 7 classes. The dataset is split in training and test sets, as shown in Table 4.1. For a subset of the test set, and only for the purpose of performance evaluation, instance-level annotations have been added. This subset with bounding boxes annotations will be used for evaluating weakly supervised object detection in Chapter 5. In particular, being able to detect and localize iconographic elements is of great importance for the study of spatial configurations, which are central to the reading of images and particularly useful given the increasing importance of Semiology. The resulting database is available online[3]. Example images are shown in Figure 4.1. Some of the visual categories cannot be learned on photographs. Typical examples include iconic characters in certain situations, such as Child Jesus, the crucifixion of Jesus, Saint Sebastian (the martyr), etc.

This dataset is way more challenging compared to other datasets, such as RASTA. Indeed, the IconArt and the Paintings datasets are designed for multilabel classification[4] task with small objects whereas RASTA is designed for a multiclass classification[5].

[1] https://commons.wikimedia.org/wiki/Main_Page

[2] https://www.wikidata.org/wiki/Wikidata:Main_Page

[3] At https://wsoda.telecom-paristech.fr/downloads/dataset/.

[4] Several labels can be associated to one image. Each label represents a different classification task, but the tasks are somehow related.

[5] Each sample can only be labeled as one class.

Class	Angel	Child Jesus	Crucifixion	Mary	nudity	ruins	Saint Sebastian	None	Total
Train	600	755	86	1065	956	234	75	947	2978
Test for classification	627	750	107	1086	1007	264	82	924	2977
Test for detection	261	313	107	446	403	114	82	623	1480
Number of instances	1043	320	109	502	759	194	82		3009

Table 4.1: Statistics of the IconArt database

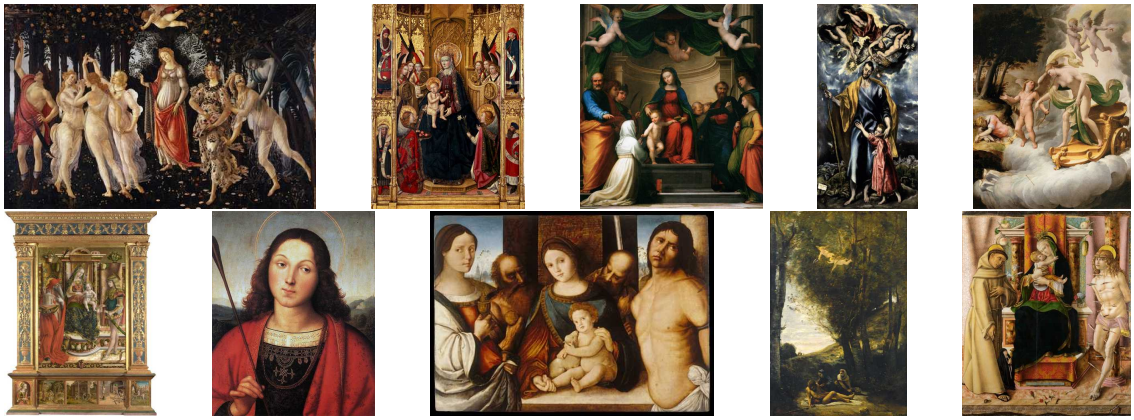


Figure 4.1: Example images from the IconArt database. Angel on the first line, Saint Sebastian on the second. We can see some of the challenges posed by this database: tiny objects, occlusions and large pose variability.

4.3 Transfer of Convolutional Neural Networks for Art Classification

This section is divided in two subsections. The first one consists in comparing classifiers trained on deep features computed on natural or artistic images. The second one is about comparing different transfer learning solutions.

4.3.1 Do We Need Artistic Images for Training?

Crowley et al. [2014] and Crowley et al. [2016] show that CNN features learned on photographs can be used for painting classification. The goal of this section is to reproduce and extend their experimental results. In a nutshell, the off-the-shelf features extraction method provides better performance when the final classifier is trained on artistic images rather than natural ones. Nevertheless, using natural images for training the classifier is an effective solution to build an online semantic recognition of arbitrary objects existing both in natural images and artistic ones (as the Oxford Painting Search [Crowley et al. 2018]).

4.3.1.1 Classification

First, we compare image-level classifiers (SVM) trained on CNN features from natural images (PASCAL VOC12) to classifiers trained on CNN features from paintings (the training set of the Paintings dataset). The two training sets have comparable sizes. In both cases these classifiers are evaluated on the test set of the Paintings dataset. All

the networks used as off-the-shelf features extractor have been pre-trained on ImageNet and then fine-tuned on VOC12 for a classification task. The features are extracted from the penultimate layer of the network. We trained one linear one-vs-rest SVM per class with optimization of the hyper-parameter using a k-fold validation. We also use a particular augmentation scheme named *Stretch*[6] which consists of taking the average feature vectors of 50 random crops from the input image[7]. It can be seen as a simple bagging of the classification network. To evaluate the different algorithms, we use the common Average Precision (AP) metric that summarizes the precision-recall curve. Results are shown in Table 4.2. First, we are able to reproduce the results from Crowley et al. [2016] except for 3.3% discrepancy in the VGG16 performances. This can be due to the fact that we may not have exactly the same pre-trained weights or to the fact that a few images from the original Paintings Dataset are missing. Second, the better the network is on ILSVRC test set the better it performs on the target domain. For instance, the InceptionResNet v2 CNN [Szegedy et al. 2017] is better than the ResNet [He et al. 2015] which is better than the VGG-based nets. The classification performance on the source domain can be seen as a good proxy to the performance on this particular dataset. A broader study as done by Kornblith et al. [2018] would be needed to confirm this conclusion. The augmentation scheme brings a constant improvement of around 3% of AP without an expensive computational cost as networks are used for inference only. Nevertheless, the classifiers learned on features from artistic images always lead to better results. Therefore it seems best to learn classifiers on artwork images than on natural images. This drop of performance training on photos has also been shown by Hall et al. [2015].

Additional results with Paintings and IconArt datasets can be found in Annex Section B.1.

4.3.1.2 Classification performance of the object detection CNNs

As an object may be relatively small and not centered, an alternative to transfer a classification CNNs is to transfer an object detection one[8]. Most of the time, detection networks process bigger resolution images and consider hundreds of regions of various sizes. This method of using an object detection CNN for doing only classification is named "Classification per detection" by Crowley et al. [2016]. As the classes we want to find in the Paintings dataset are present in object detection datasets such as VOC [Everingham et al. 2010] or MS COCO [Lin et al. 2014], we will consider using them for off-the-shelf classification without any additional training. We evaluate the performance of different detection networks pre-trained on ImageNet and fine-tuned on different natural images datasets for an object detection task. We compare the Faster R-CNN [Ren et al. 2015] with different backbones and two versions of the YOLO net [Redmon et al. 2016; Redmon et al. 2017]. We also try to use Faster R-CNN as feature extractor by considering only the region with the highest objectness score to train one linear SVM per class with the Paintings train set. In this experiment, we always use the penultimate layer of the CNNs (for instance fc7 for ResNet architecture). The

[6]This augmentation scheme is available in the MatConvNet toolbox and compared to other augmentation scheme in [Crowley et al. 2016].

[7]Note that the same augmentation scheme is applied to both training and test images.

[8]The definition of the object detection is given at the end of Section 2.1.2.2.

CNN	Training Set	Augmentation	mAP	mAP results from [Crowley et al. 2016]
VGG-M [Simonyan et al. 2015]	Paintings Dataset	none	•	50.8
		Stretch	•	52.9
VGG-16, relu7 (VD-16, 4096-D) [Simonyan et al. 2015]	VOC12	none	58.1	54.8
		Stretch	59.2	56.8
	Paintings Dataset	none	69.8	68.7
		Stretch	72.0	70.8
VGG19 relu7 [Simonyan et al. 2015]	VOC12	none	58.5	•
		Stretch	60.3	•
	Paintings Dataset	none	69.6	•
		Stretch	72.4	•
ResNet-152 fc7-2048-D [He et al. 2015]	VOC12	none	61.0	60.5
		Stretch	63.7	62.3
	Paintings Dataset	none	73.3	72.5
		Stretch	76.0	75.0
InceptionResNet v2 1536D [Szegedy et al. 2017]	VOC12	none	63.8	•
		Stretch	67.5	•
	Paintings Dataset	none	74.9	•
		Stretch	**77.6**	•

Table 4.2: Classification on **Paintings Dataset** (test set) [Crowley et al. 2014]: mean AP (%) on the 10 classes. Comparison of classifiers trained on deep features from natural images or paintings ones.

performances of these object detection CNNs transferred to a classification task in a new modality can be seen in Table 4.3.

First, we can see the influence of the quality of the network and of the quality of the training set. Newer CNN architectures perform better than older ones. Faster R-CNN ResNet achieves better performance than Faster R-CNN VGG16, and YOLO version 2 performs better than the version 1. It is the case both for natural image detection and our cross domain classification task. Then, the model generalizes better if it has been trained with a bigger dataset. A detector achieves higher performance on the Paintings test set if it has been trained on MS COCO rather than on VOC12. For the Faster R-CNN (VGG16) architecture, we have almost 4% of difference.

Second, the simplest way to transfer a detection network to a target domain is to consider that the region with the highest "objectness" score contains the object of interest (i.e. the label associated to the image). An SVM is trained on the deep features extracted from the highest "objectness" score region of each image. This method is referred as MAX in Chapter 5. This simple transfer of the detector achieves worse score than the off-the-shelf equivalent model trained on a different modality. This is due to the fact that the object of interest is not always the main object, nor is it well centered as it is implied in the Crowley et al.'s paper [2016]. This can be seen in Figure 4.2. It is therefore necessary to provide a better way to transfer those detection networks. We propose such a scheme in Chapter 5. Finally, the good performance of these detection models compared to the classification ones (Table 4.2) may also be due to the larger size of the input image, as CNNs trained with larger resolution images tend to achieve higher accuracy [Tan et al. 2019]. Smaller objects can be detected and

more details are provided to the model.

CNN	Training Set	mAP	mAP results from [Crowley et al. 2016]
Faster R-CNN (VGG16)	VOC07	56.4	•
Faster R-CNN (VGG16)	VOC12	62.2	62.7
Faster R-CNN (VGG16)	MS COCO	66.7	•
Faster R-CNN (VGG16-COCO) SVM on best objectness score region	Paintings Dataset	64.8	•
Faster R-CNN (ResNet-101)	VOC12	65.5	•
Faster R-CNN (ResNet-101)	MS COCO	**71.6**	•
Faster R-CNN (ResNet-101-COCO) SVM on best objectness score region	Paintings Dataset	69.5	•
Faster R-CNN (ResNet-152)	MS COCO	**71.6**	•
Faster R-CNN (ResNet-152-COCO) SVM on best objectness score region	Paintings Dataset	70.0	•
YOLO v1	VOC12	49.4	•
YOLO v2	VOC12	62.8	•
YOLO v2	MS COCO	65.0	•

Table 4.3: "Classification per detection" on **Paintings Dataset** (test set) [Crowley et al. 2014]: mean AP (%) on the 10 classes. Comparison of deep detection models on a classification task, without any augmentation scheme.

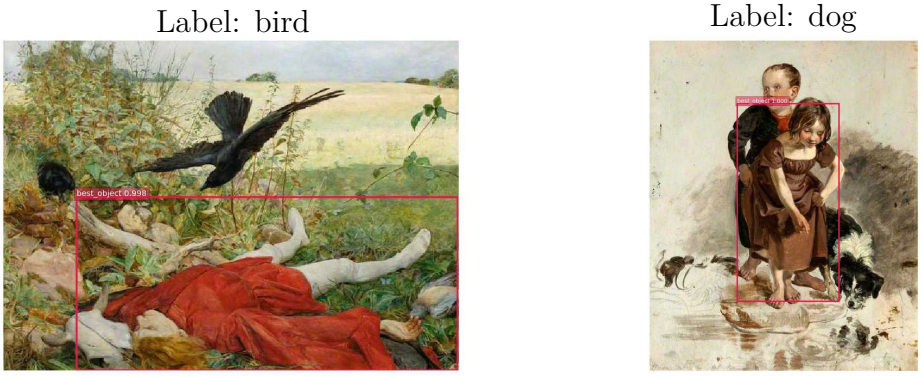


Figure 4.2: Highest objectness score box with Faster R-CNN ResNet-152 fine-tuned on MS COCO on two images from the training set of the Paintings Dataset [Crowley et al. 2014].

This section confirms that a deep model pre-trained on photographs can be efficiently transferred to automatic visual recognition in artwork, provided that the classes we are looking for are the same. This transfer can be done without fine-tuning a whole deep CNN but we will study other transfer learning methods in the following sections.

4.3.2 Comparison of the Different Transfer Learning Approaches

In this section, we will compare the three main different modalities for transfer learning mentioned before Section 2.1.2.1. The first consists in taking the penultimate output of the pre-trained network to make it the input of a simple classifier [Donahue et al. 2014]. In the following, we refer to this approach as the *off-the-shelf* method. The second option consists in *fine-tuning* FT the pre-trained network for the new task [Girshick et al. 2014]. The last one consists in training a successful architecture with a random initialization, referred to as training *from scratch*. We will present the performances on multiclass style classification and multilabel object classification, both in paintings.

Experimental Setup: We will compare three different architectures: InceptionV1 [Szegedy et al. 2015], VGG19 [Simonyan et al. 2015] and ResNet50 [He et al. 2015]. The last layer of the network is replaced by a fully connected layer with the number of outputs corresponding to the dataset at hand and the activation function is a softmax for RASTA or a sigmoid for Paintings and IconArt datasets. The loss function is the usual cross-entropy in the first case, and the sum over the classes of binary cross-entropy in the two others. We ran experiments with a varying number of hyperparameters. The hyperparameters of the different training schemes can be found in Table 4.4. The hyperparameters we have varied are the learning rate for the dense classification layer, the learning rate for the convolutional layers, the use of a deep supervision (adding auxiliary supervision branches after certain intermediate layers during training), the number of unfrozen layers (i.e. trainable layers), the maximum number of epochs and the optimizer. We also consider two different data augmentation schemes. The first one named "Small transformation" consists in small geometric transformation (horizontal flip and/or translation of 28 pixels maximum). The second one named "Random crops" consists in taking random crop of size 224×224 within an image resized to 256 for its smallest dimension. For all models, the input size is 224 × 224. Finally, the "from scratch" trainings have the same learning rate schedule as in [Szegedy et al. 2015], originally designed for the InceptionV1 model.

Mode	Learning rate on the last dense layer	Learning rate on the other layers	Deep Supervision	Maximum number of epochs	Number of unfrozen layers	Data Augmentation	Optimizer	Learning rate schedule
A	0.01	0.001	No	20	All	No	SGD	No
B	0.001	0.0001	No	20	All	No	SGD	No
C	0.001	0.001	No	20	All	No	SGD	No
D	0.001	0.0001	Yes	20	All	No	SGD	No
E	0.001	0.001	Yes	20	All	No	SGD	No
F	0.01	0.01	No	20	All	No	SGD	No
Higher layers from scratch	0.0001	0.0001	No	200	InceptionV1: 50 VGG19: 8 ResNet50: 20	Small transformation	Adam	No
From scratch	0.001	0.001	Yes if InceptionV1	200	All	Random crops	SGD	As in [Szegedy et al. 2015]

Table 4.4: Hyperparameters of the different training schemes.

For all experiments, at a fix set of hyperparameters, we selected the model with the best loss value on the corresponding validation set. The relatively small size of our dataset limits intensive search of the best hyperparameters. Maybe an intensive search

of the best hyperparameters could lead to better performances but it will certainly need a huge computational cost.

4.3.2.1 Artistic Style Recognition in Paintings

Even though the goal of this work is not to reach the best possible classification performance, we display the corresponding results in Table 4.5 to further characterize the considered fine-tuning. From this Table, one sees that a simple and short fine-tuning of a pre-trained model yields a better performance than the off-the-shelf strategy. The former method is based on extracting features from the ImageNet pretrained model and training the last layer. The features extracted may be too specific to the ImageNet classification task and the classification head too small. With training from scratch, we failed to obtain a model as efficient as with the ImageNet pretraining, on the contrary to the results in [He et al. 2019] about object detection. This can be due to the relatively small size of the RASTA dataset. Some data augmentation and a longer training was required to reach 45.29% Top1 accuracy for the InceptionV1 architecture. The conclusions are identical for the three architectures considered. This experiment confirms Lecoutre et al. [2017] conclusions for two other deep architectures. We conclude this section by observing that a simple way to improve results is to simply average the prediction of three models trained with different strategies for VGG19 and InceptionV1. In the case of ResNet50 it is not the case, but the models trained from scratch have really low performance (about 37% Top-1 accuracy) which may account for the lack of improvement.

CNN	Method	Top-1	Top-3	Top-5
InceptionV1	Off-the-shelf with pretraining on ImageNet	30.95	58.71	74.10
	FT from a pretraining on ImageNet (Mode A)	**55.18**	**82.25**	**91.06**
	Training from scratch the end of the model with pretrained frozen low-level layers	50.35	78.04	88.42
	Trained from scratch	45.29	73.44	84.67
	Ensemble of the 3 previous models	*58.76*	*83.99*	*92.23*
VGG19	Off-the-shelf with pretraining on ImageNet	37.09	64.70	77.33
	FT from a pretraining on ImageNet (Mode A)	**53.04**	**81.66**	**90.44**
	Trained from scratch the end of the model with pretrained frozen low-level layers	46.36	74.87	86.38
	Trained from scratch	46.19	73.94	85.47
	Ensemble of the 3 previous models	*54.24*	*81.03*	*90.61*
ResNet50	Off-the-shelf with pretraining on ImageNet	44.67	73.56	85.55
	FT from a pretraining on ImageNet (Mode A)	**55.99**	**82.71**	**91.70**
	Trained from scratch the end of the model with pretrained frozen low-level layers	37.08	65.53	78.54
	Trained from scratch	37.89	65.31	79.11
	Ensemble of the 3 previous models	54.02	79.71	89.14

Table 4.5: Top-k accuracies (%) on RASTA dataset [Lecoutre et al. 2017] for different methods and CNN architectures.

4.3.2.2 Object Classification in Paintings

As the notion of style is too broad, we find object classification to be more precise and challenging. Tables 4.6 and 4.7 compare the classification results obtained for the object classification task on the Paintings dataset [Crowley et al. 2014] and on the IconArt dataset when using a model pretrained on ImageNet or fine-tuned on RASTA dataset (from an ImageNet initialization or from scratch). On the contrary to [Sabatelli et al.

2018] where the authors compared the pretraining on the Rijkmuseum dataset and ImageNet for the Antwerp dataset, the task is not the same between the two artistic datasets: classification of the artistic style versus object classification. We run these comparisons for three different architectures. For the double fine-tuning, we consider the hyperparameters of the Mode F (Table 4.4).

Once again, the fine-tuning of a CNN pretrained on ImageNet outperforms off-the-shelf and training from scratch strategies. The same conclusion has been made by Tan et al. [2016] for the AlexNet architecture on style, genre and artist classification tasks. The most important observation is that the double fine-tuning (first using RASTA, then using the target dataset) outperforms the direct fine-tuning using only the target dataset for the VGG19 and InceptionV1 architectures on both datasets and for ResNet50 on the IconArt dataset. There is a gain of performance while both the small dataset and ImageNet share the same task (object classification) but do not share it with RASTA (designed for artistic style classification). The visual proximity of the images seems more important than the learning task. The filters learned on RASTA seem to be more adapted to other artistic datasets and ease the transfer in these two cases (IconArt and Paintings) where the datasets are relatively small. Finally, a model partly or entirely trained from scratch on RASTA will not provide a good initialization point for fine-tuning compared to the double pretraining, neither for IconArt, nor for Paintings. This is most probably due to the size of the RASTA dataset. But it is a better initialization point than a random initialization for these small art datasets. We have a gain of 4% for Inception and 8% for VGG19, when considering the training of the whole net from scratch.

CNN	Method	mAP
InceptionV1	Off-the-shelf with pretraining on ImageNet	56.0
	Off-the-shelf with pretraining on ImageNet and RASTA	52.4
	Off-the-shelf with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	45.6
	Off-the-shelf with training from scratch on RASTA	37.5
	FT with pretraining on ImageNet (Mode A)	64.8
	FT with pretraining on ImageNet (Mode F)	56.2
	Trained from scratch for the end of the model with pretrained frozen low-level	44.6
	Trained from scratch	46.9
	FT (Mode F) with pretraining on ImageNet and RASTA (Mode A)	**65.6**
	FT (Mode F) with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	59.6
	FT (Mode F) with training from scratch on RASTA	49.1
VGG19	Off-the-shelf with pretraining on ImageNet	63.6
	Off-the-shelf with pretraining on ImageNet and RASTA	47.1
	Off-the-shelf with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	33.2
	Off-the-shelf with training from scratch on RASTA	33.0
	FT with pretraining on ImageNet (Mode A)	55.9
	FT with pretraining on ImageNet (Mode F)	Div
	Trained from scratch for the end of the model with pretrained frozen low-level	54.4
	Trained from scratch	36.3
	FT (Mode F) with pretraining on ImageNet and RASTA (Mode A)	**65.3**
	FT (Mode F) with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	51.8
	FT(Mode F) with pretraining from scratch on RASTA	46.8
ResNet50	Off-the-shelf with pretraining on ImageNet	67.1
	Off-the-shelf with pretraining on ImageNet and RASTA	62.7
	Off-the-shelf with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	39.4
	Off-the-shelf with training from scratch on RASTA	34.3
	FT with pretraining on ImageNet (Mode A)	**70.5**
	FT with pretraining on ImageNet (Mode F)	69.4
	Trained from scratch for the end of the model with pretrained frozen low-level	36.0
	Trained from scratch	33.7
	FT (Mode F) with pretraining on ImageNet and RASTA (Mode A)	68.6
	FT (Mode F) with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	50.2
	FT (Mode F) with training from scratch on RASTA	40.8

Table 4.6: Mean Average Precision (%) on Paintings test set [Crowley et al. 2014] for three architectures and transfer learning schemes. "Div" means the model diverges for this setup.

CNN	Method	mAP
InceptionV1	Off-the-shelf with pretraining on ImageNet	53.2
	Off-the-shelf with pretraining on ImageNet and RASTA	54.4
	Off-the-shelf with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	47.5
	Off-the-shelf with training from scratch on RASTA	44.6
	FT with pretraining on ImageNet (Mode A)	59.2
	FT with pretraining on ImageNet (Mode F)	41.7
	Trained from scratch for the end of the model with pretrained frozen low-level	54.8
	Trained from scratch	46.2
	FT with pretraining on ImageNet and RASTA	**67.4**
	FT with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	59.4
	FT with training from scratch on RASTA	50.1
VGG19	Off-the-shelf with pretraining on ImageNet	54.5
	Off-the-shelf with pretraining on ImageNet and RASTA	49.3
	Off-the-shelf with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	43.6
	Off-the-shelf with training from scratch on RASTA	43.5
	FT with pretraining on ImageNet (Mode A)	56.9
	FT with pretraining on ImageNet (Mode F)	Div
	Trained from scratch for the end of the model with pretrained frozen low-level	49.4
	Trained from scratch	40.6
	FT with pretraining on ImageNet and RASTA	**64.5**
	FT with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	51.7
	FT with pretraining from scratch on RASTA	48.4
ResNet50	Off-the-shelf with pretraining on ImageNet	46.9
	Off-the-shelf with pretraining on ImageNet and RASTA	48.3
	Off-the-shelf with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	43.1
	Off-the-shelf with training from scratch on RASTA	37.6
	FT with pretraining on ImageNet (Mode A)	64.7
	FT with pretraining on ImageNet (Mode F)	66.2
	Trained from scratch for the end of the model with pretrained frozen low-level	46.2
	Trained from scratch	35.4
	FT with pretraining on ImageNet and RASTA	**66.3**
	FT with the low level pretrained on ImageNet and the high level trained from scratch on RASTA	49.5
	FT with training from scratch on RASTA	40.6

Table 4.7: Mean Average Precision (%) IconArt test set for three architectures and transfer learning schemes. "Div" means the model diverges for this setup.

4.4 Analyzing Convolutional Neural Networks trained for Art Classification

In this section, we investigate the effect of fine-tuning in the case of artistic images. In order to do so, we rely both on visualization techniques and on quantification of the change the network undergoes. Our experimental results are organized in four sections. First, we consider an Inception V1 network pre-trained on ImageNet and fine-tuned on RASTA for artistic style classification (Section 4.4.1). Then we consider the same architecture with a random initialization (from scratch) trained on RASTA (Section 4.4.2). The Section 4.4.3 is dedicated to the evaluation of the changes implied by the training on RASTA. Finally, we study the same architecture pre-trained on ImageNet and then fine-tuned first on RASTA and then on a smaller art dataset for object classification (Section 4.4.4) to see how using an intermediate art dataset can help.

Feature visualization The first visualization technique we use consists in generating *optimized images*, as introduced in [Olah et al. 2017] and defined in Section 2.2.1.1. These images are obtained by maximizing the response to a given channel. The entire feature map at a given layer has two spatial dimensions and a third depending on the convolutional kernel. A *channel* denotes one element according to this last dimension. We use the Lucid framework for visualizing convolutional channels via activation maximization. We use Lucid's 2D FFT image representation with decorrelation and 2048 iterations of the gradient ascent.

Maximal Activation Images We devise another indicator that might be useful for the analysis of the transformation that a network undergoes during its transfer to a different domain. This indicator is the set of the *maximal activation images*. For a given channel, we compute the *top 100* images in the target dataset that most trigger it. Indeed, these images are also useful to understand a CNN as shown by Borowski et al. [2020]. On this set, we compute the information *entropy* over classes for each top 100 images, in order to evaluate the clustering power of the corresponding channel. The entropy is defined as:

$$\frac{1}{maxE} \sum_{classes} -p_c log_2(p_c), \tag{4.1}$$

with p_c the fraction of images in the top 100 belonging to the class c and $maxE$ the maximal entropy with this number of classes.

Moreover, the top 100 can be computed twice, once at the beginning and once at the end of the fine-tuning. The percentage of the images that lie in both sets is an indicator of how much the channel has drifted during its adaptation. These percentages are named *overlapping ratio* in the following. They are, in many cases, much higher than what we would expect from a random reshuffling of the dataset. Besides, the combination of this indicator with the visualization technique from [Olah et al. 2017] leads to several findings that we will present thereafter.

Experimental Setup: All our visualization experiments use the InceptionV1 [Szegedy et al. 2015] CNN. This network is the classical and most expressive choice for feature visualization by optimization [Olah et al. 2017] although it no longer produces the best classification performances.

In the following sections, we analyze how the networks have been modified by the fine-tuning process. We present qualitative observations using optimized images and the maximal activation images, as well as quantitative evaluations relying on the ℓ_2 norm of the difference between convolution kernels and the linear CKA measure [Kornblith et al. 2019].

4.4.1 From Natural to Art Images

The first feature visualizations we report have been obtained by fine-tuning on the RASTA classification dataset, an InceptionV1 architecture pretrained on ImageNet with different sets of hyperparameters. The detailed architecture with all the layer names can be found in Table D.1.

Low-level layers are only slightly modified by the fine-tuning. The first observation is that low-level layers from the original network trained on ImageNet are hardly modified by the fine-tuning on RASTA (see Figure 4.3). This fact will be confirmed by the CKA measure (see Figure 4.9) and the overlapping ratio of the top 100 maximal activation images (see Figure 4.10a) in Section 4.4.3.

Note that in [Yin et al. 2016], it has been shown that the low-level layer filters have been modified by a fine-tuning on an almost monochrome drawing training dataset. This suggests that the statistics of painting images are closer to those of natural images than those of drawing ones are.

Mid-level layers adapt to the new dataset. Some of the filters have been modified to the specificity of the new dataset by the fine-tuning process, as illustrated in Figures 4.4g to 4.4i. In these figures are displayed for some channels, the optimized images defined in Section 2.2.1.1. The model learned a red and blue drapery detector, a blue mountain one and a house pediment one. It is worth mentioning that other channels are hardly modified by the fine-tuning process. When looking on maximal activation images, we can make three observations. First, among the 70k training samples, some maximal activation images are present in the top 100 both before and after fine-tuning. These images are surrounded by a green line in the last row of Figures 4.4 and 4.5. Second, in those maximal activation images, we can recognize the pattern that emerged in the optimized image (when we compare the third and last rows). For instance, in the third column of Figure 4.4, a flower-like structure is transformed into a house pediment one. Finally, we observe that the detector fine-tuned on RASTA concentrates images with this specific pattern (last row of Figures 4.4 and 4.5). The first group of images of the last row contains characters with a blue dress (as the Mary character), the second one blue mountains and the last one buildings depicted with some perspective.

On the other hand, for other channels, the pattern is already present in the optimized image and the detector is slightly adapted to the new dataset. This appears

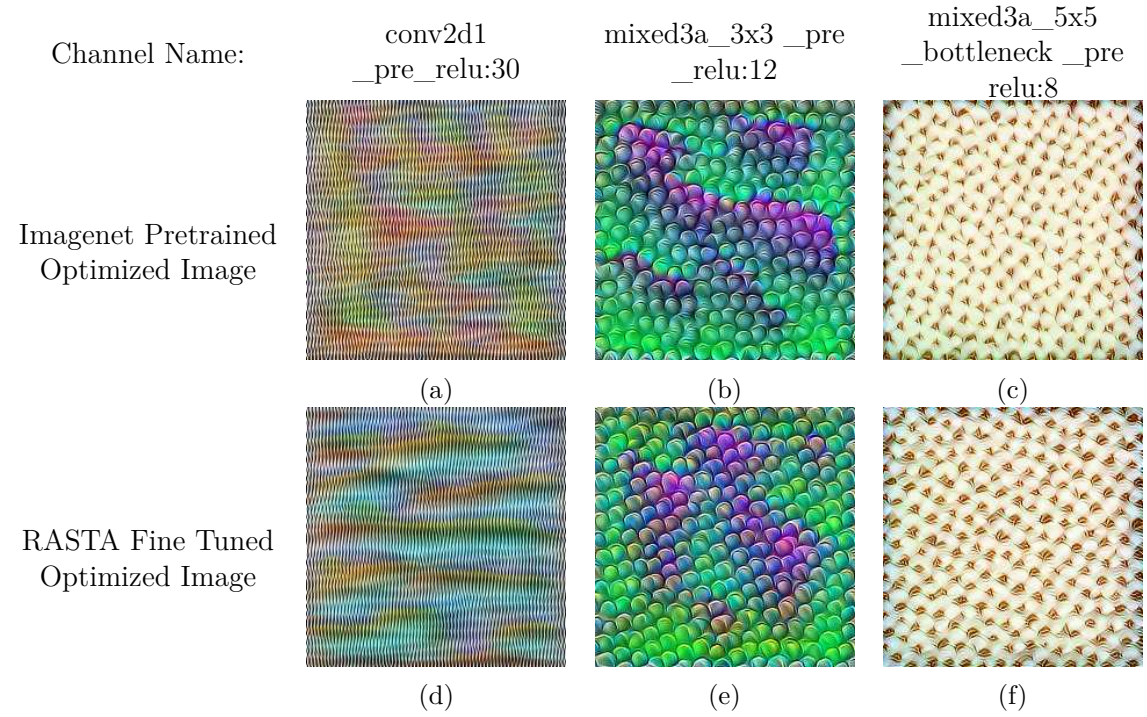


Figure 4.3: Optimized Images for one individual channel from different low-level layers. First row InceptionV1 pretrained on ImagneNet, second row fine-tuned on RASTA.

in the form of a minor modification of the optimized image. An arch detector within the pretrained ImageNet model has been modified to detect bigger arches as it can be seen in Figure 4.5g. The maximal activation images before the fine-tuning already was composed of many buildings images. In this case, the overlapping ratio between the two sets of maximal activation images is equal to 46%. Two other channels are shown in Figure 4.5. Besides the visualizations we highlighted, the reader must keep in mind that most channels are not modified by the fine-tuning and few are not interpretable at all[9].

Learned filters have a high variability. We ran 2 distinct fine-tunings for each of the 5 considered optimization schemes named Mode A to E, see Table 4.4. The initial last layer is different as well as the order of the images in the mini-batches during the training process. From a same starting point (the ImageNet weights) but for different hyper-parameters, the training process may sometimes converge to similar optimized images. On the contrary, two optimizations with the same hyper-parameters (same mode) may lead to very different detectors. These phenomena are illustrated in Figure 4.6. For this given channel, according to the mode and occurrence of the fine-tuning, one can recognize houses (Figure 4.6a), flowers (Figure 4.6e), a mix of houses or more abstract patterns (Figure 4.6i). As a temporary conclusion, we can say that ImageNet pre-trained filters seem to be a good initialization for learning useful new filters adapted to the artistic style classification and they also allow to learn a variety

[9]The reader can find more feature visualizations at https://artfinetune.telecom-paris.fr/data/

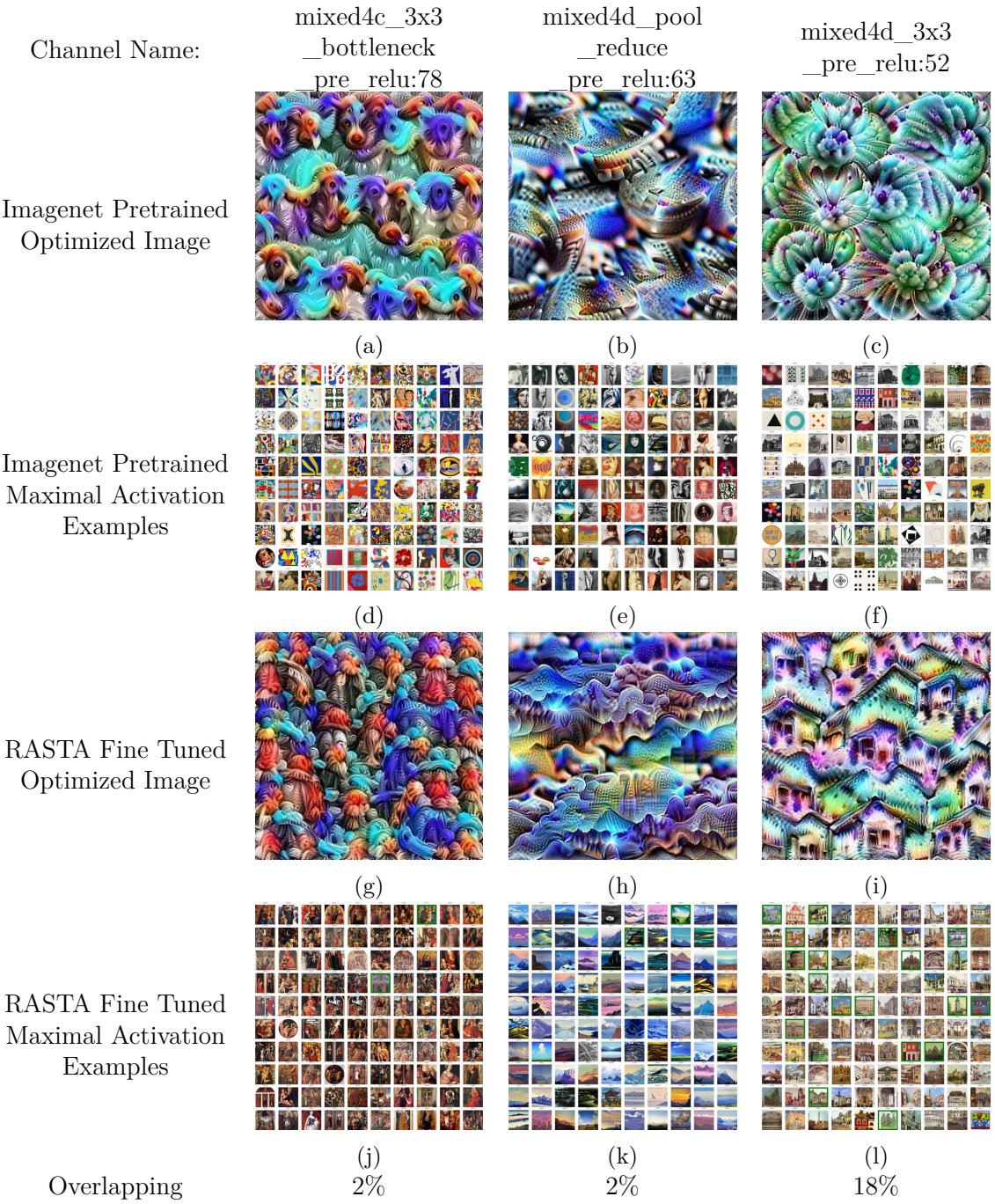


Figure 4.4: Optimized Image and Maximal Activation Examples for one Individual channel. First and second rows InceptionV1 pre-trained on ImagneNet, third and fourth rows fine-tuned on RASTA. The images surrounded by a green line are already present in the top 100 of the pre-trained model. The percentage of images in common between the two sets of maximal activation images is displayed at the bottom of each column.

of new filters. The percentage of overlap between the set of maximal activation images before and after fine-tuning seems to be correlated to the amount of visual change.

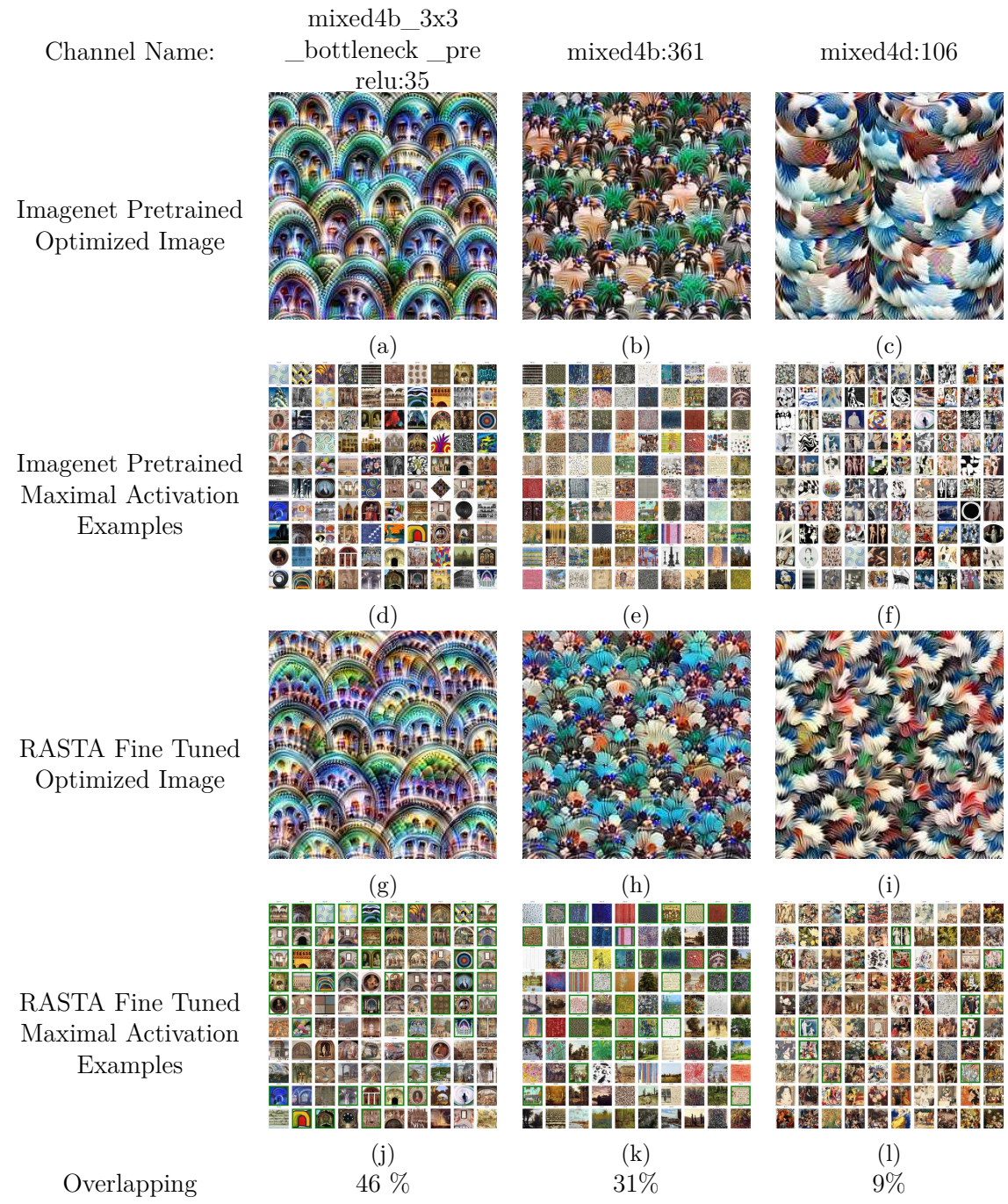


Figure 4.5: Optimized Image and Maximal Activation Examples for one Individual channel. First and second rows InceptionV1 pre-trained on ImagneNet, third and fourth rows fine-tuned on RASTA. The images surrounded by a green line are already present in the top 100 of the pre-trained model. The percentage of images in common between the two sets of maximal activation images is displayed at the bottom of each column.

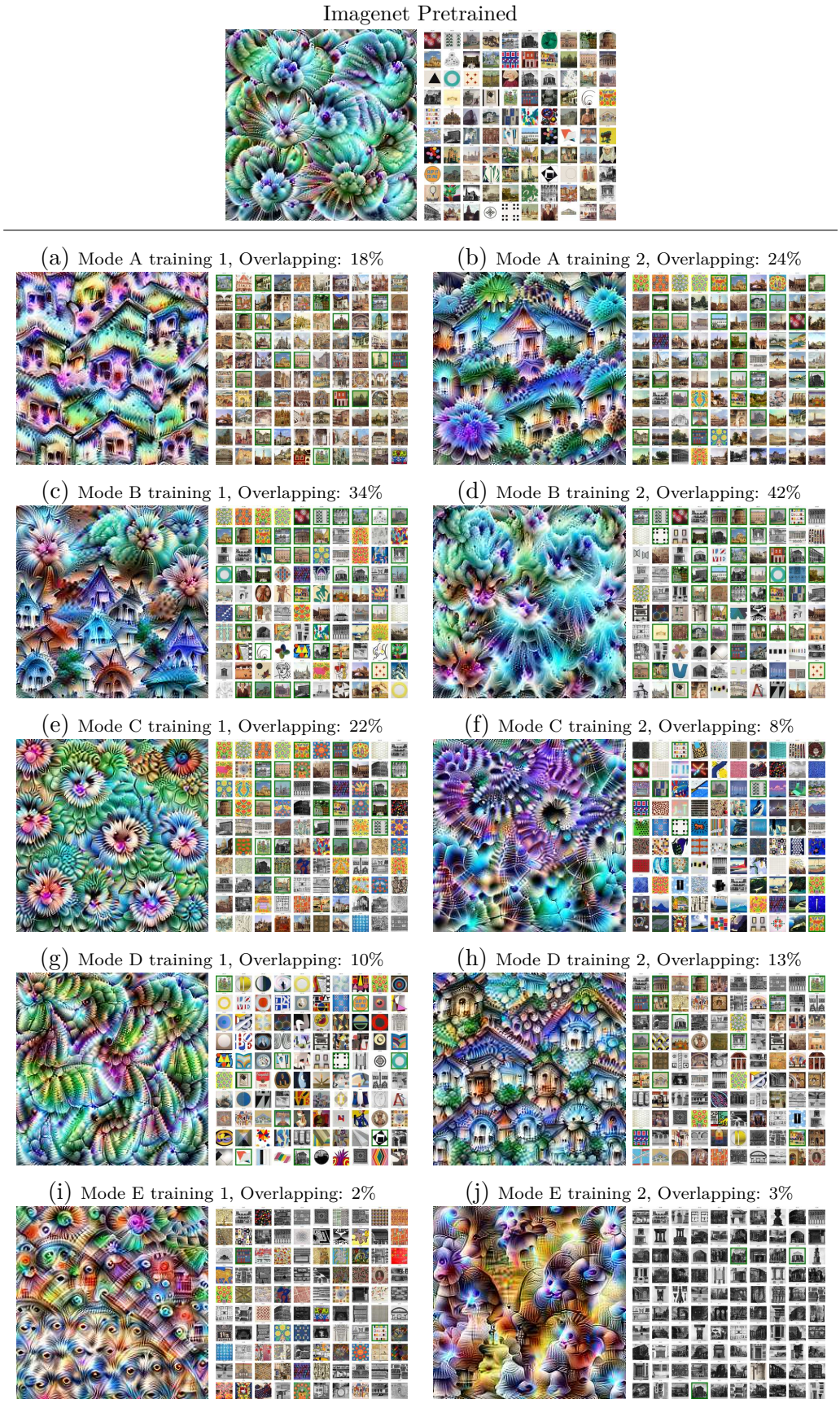


Figure 4.6: Optimized Image and Maximal Activation Examples for a given channel (mixed4d_3x3_pre_relu:52) with different training. The overlapping between the two sets of maximal activation images is displayed on top of each pair of images. The images surrounded by a green line are already present in the top 100 of the pre-trained model.

High-level filters concentrate images from the same classes. The visualizations of high-level layers (near the classification output) are more difficult to interpret, as illustrated in Figure 4.7. The network seems to mix different visual information from the previous layers. Nevertheless, the group of images with maximal activation for those 2 given channels gather images from the same artistic style after fine-tuning[10]. The first channel is mostly fired by Ukiyo-e images (82% in Figure 4.7b), the second one gathers western renaissance artworks (87% by summing up the three prevailing classes in Figure 4.7d). There is no visual clue to such clustering in the optimized images. In the last image, one may see some green tree in front of a blue sky and some drapery. The fact that Early_Renaissance, High_Renaissance and Mannerism_(Late_Renaissance) classes are clustered together maybe due to their strong visual similarity. Deep model commonly mislabel one of these as another, as mentioned in [Lecoutre et al. 2017].

(a) Optimized Image (b) Maximal activation examples: 1% (c) Optimized Image (d) Maximal activation examples: 0%

Top 100 composition:

Ukiyo-e 82 %
Northern_Renaissance 14 %
Early_Renaissance 3 %

Early_Renaissance 48%
High_Renaissance 27%
Mannerism_(Late_Renaissance) 12%

Figure 4.7: Optimized Images and Maximal Activation Examples for two high level layers for the model fine-tuned on RASTA. The overlapping ratio between the set of maximal activation images before and after fine-tuning is displayed under the images. The images surrounded by a green line are already present in the top 100 of the pre-trained model. The percentage of the 3 most common class is displayed below.

4.4.2 Training from Scratch

Mid-level detectors can be learned from scratch when low-level layers are transferred from ImageNet. The next experiment consists in fine-tuning a network whose low-level layers are initialized using the training on ImageNet and frozen whereas the mid and high-level layers are initialized randomly. In this case, the network is able to learn useful and comprehensible mid-level detectors such as drapery or checkerboard as illustrated in Figures 4.8a and 4.8b[11]. The stability of these mid-level detectors that are rebuilt despite the random initialization is somehow surprising. This

[10]The pretrained filters can be found in Annex, Section B.2.1.

[11]The corresponding feature visualizations for the initial model can be found in Annex Section B.2.2.

phenomenon is most likely made possible through the low-level layers inherited from the ImagNet training.

The optimized images are more difficult to interpret with a full training from scratch. A network trained fully from scratch seemingly yields the same kind of low-level filters that the ones pretrained on ImageNet whereas the mid and high-level layers provide optimized images that are much more difficult to interpret (see Figures 4.8c and 4.8d). A possible explanation is that the network may not need to learn very specific filters given its high capacity.

The training of the network provides filters that are able to fire for a given class such as Ukiyo-e (Figure 4.8g) or Magic_Realism (Figure 4.8h) without being interpretable for humans. Moreover, they are mid-level layer as in Figure 4.4 and not high-level one as in Figure 4.7. The class concentration appears quantitatively sooner in the network trained from scratch (see Figure B.6c) than in the fine-tuned model (see Figure 4.10b).

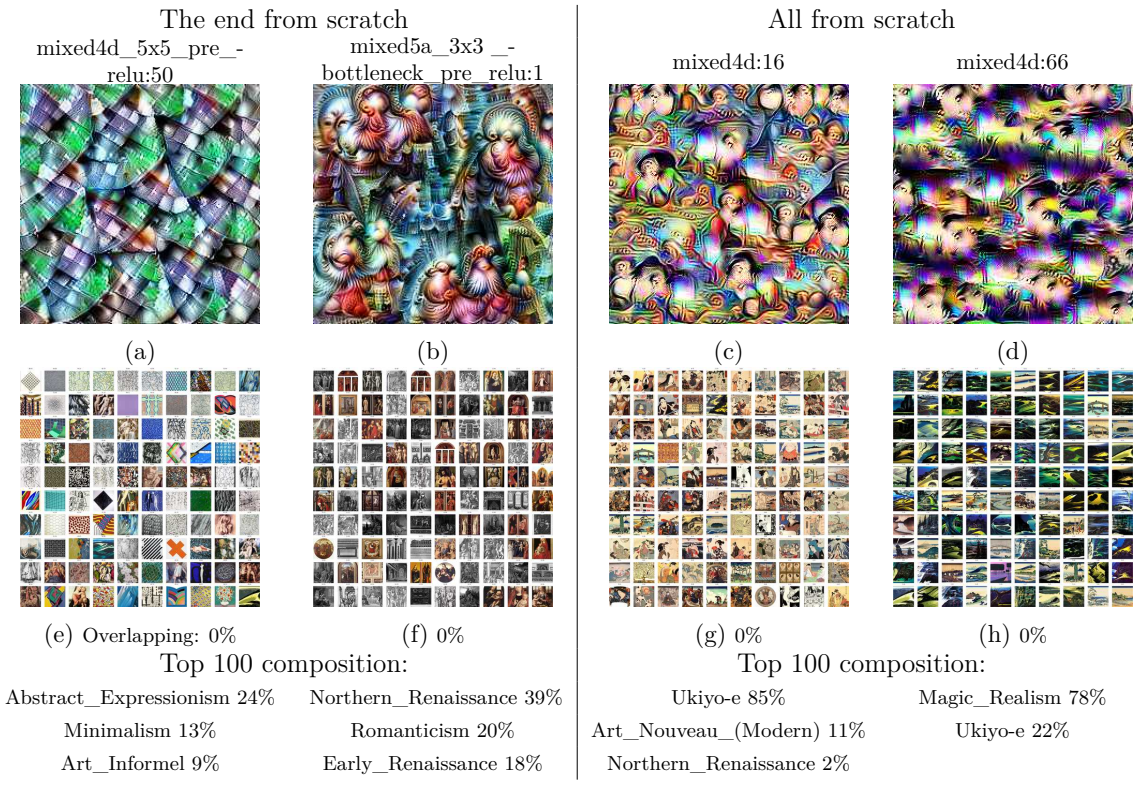


Figure 4.8: Optimized Image and Maximal activation examples from different mid-level layers. On the left: fine-tuning is performed starting from low-level layers initialized from ImagNet and upper layers initialized at random. On the right, the fine-tuning is fully performed from scratch (randomly initialized layers). The overlapping ratio between the set of maximal activation images before and after fine-tuning is displayed under the images. The percentage of the 3 most common class is displayed below.

4.4.3 Quantitative Evaluation of the CNNs Modification

In order to quantify some of the previous observations, we make use of the linear CKA [Cortes et al. 2012; Kornblith et al. 2019] as a measure of similarity between the feature maps of two different networks. This similarity index is defined in equation (2.6). For computational reasons, we summarize a channel by its global average, thus drastically reducing the dimensionality of the problem. Otherwise, the computations would have been intractable. The results are shown in Figure 4.9. The CKA values for nine pairs of models are printed in function of twelve different layers from the InceptionV1 architecture. A value of CKA near to 1 means the feature maps at this layer are similar between the two models, whereas a value near to 0 means the features are dissimilar.

First, we compare the ImageNet-pretrained model with its fine-tuned version, by looking at the dark blue line. We can observe a decrease of the CKA when the depth of the layers increases. This is a confirmation of what we observed previously with the optimized images (Section 4.4.1), the low level layers are barely modified (with a $CKA \simeq 1$ for the conv2d0 layer) whereas the high level are strongly changed (with a $CKA = 0{,}37$ for the mixed5b layer).

No matter which layer is considered, the pairs of fine-tuned models have the highest CKA values, according to the green and light blue lines. Thus, the fine-tuned models are the closest one to each other. The high level layers of these models are very similar because these models have been trained on the same dataset from the same initialization point.

The CKA also decreases with layers when we compare a model trained from scratch to its random initialization (purple and orange curves). High level layers have been more modified by the training process than the low level ones. For the model trained from scratch, the lower layer (conv2d0) has a CKA of 0,89 whereas the deepest one (mixed5b) has a CKA of 0,35. Moreover, the models trained from scratch on RASTA are closer to the model fine-tuned on RASTA than the ImageNet-pretrained model. We can observe it by comparing the brown curve to the red one and the yellow to the pink. This is clearly due to the training on the same dataset, namely RASTA.

The values of CKA presented here are higher than the ones obtained in [Neyshabur et al. 2020] for X-ray images. In the case of the model trained from scratch, we even observe several orders of magnitude of difference. This confirms and quantifies the fact that the structure of artistic images is closer to the one of natural images than the X-ray images are.

In addition to feature similarity, we also look at the distance between two models in the parameter space in Table 4.8 (as in the recent work of Neyshabur et al. [2020]). We can see that the fine-tuned models are still close one to another and also close to the ImageNet pretrained initialization. In contrast, the models trained from scratch are much farther away from their initialization.

We also observe the evolution of the overlapping ratio between the ImageNet-pretrained model and the fine-tuned one for the top 100 maximal activation images in Figure 4.10a. We can see a monotonic decrease of this ratio with the depth of the layer. This is another illustration of the fact that the high level layers are more modified by the fine-tuning. The behavior is the same if we consider the top 1000 maximal activation images. One also observes that channels with low overlapping ratio seem to correspond to optimized images that are more modified by the fine-tuning. This

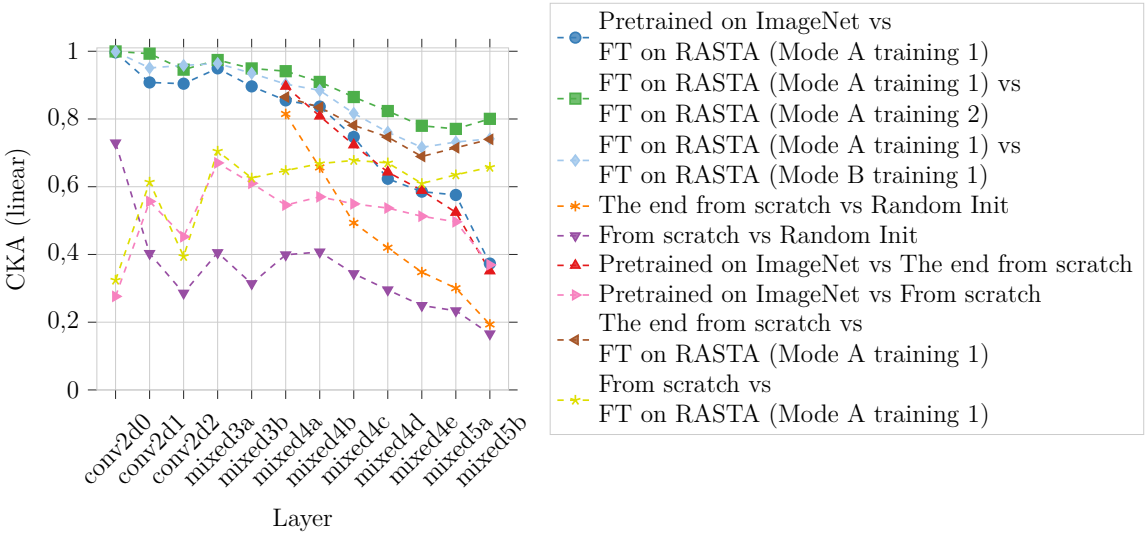
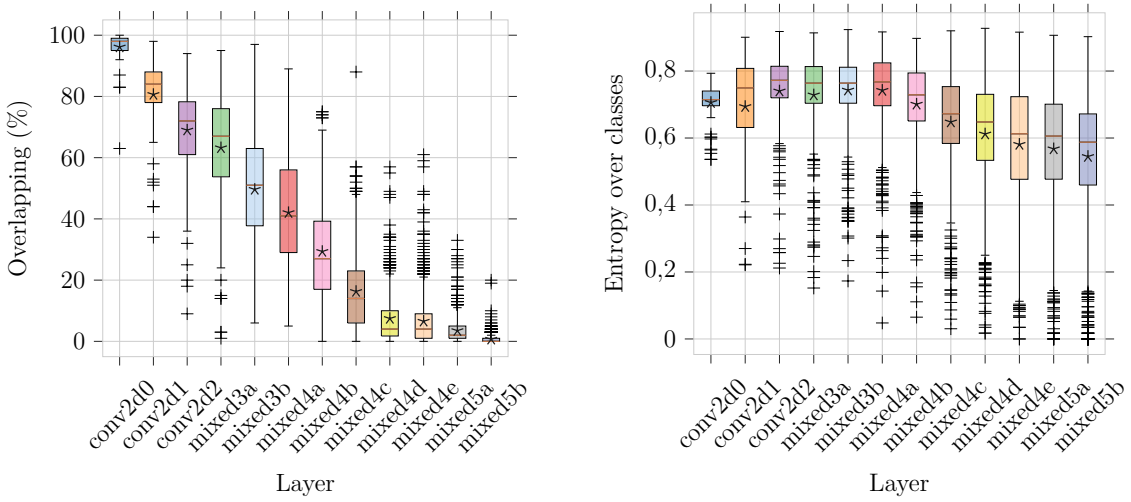


Figure 4.9: CKA computed on RASTA test set for different models trained or fine-tuned on RASTA train set.



(a) Boxplots of Overlapping ratio.

(b) Boxplots of Entropy over classes.

Figure 4.10: Boxplots of some metrics on the top 100 maximal activation images for the model fine-tuned on RASTA (Mode A1). For each box, the horizontal orange line corresponds to the average result and the star to the median. The crosses are outliers (i.e. points outside 1.5 times the interquartile range).

fact should be investigated further and could yield a simple way to browse through optimized images. In the case of a model trained from scratch the overlapping is equal to zero for almost all the layers (see Figure B.5b in Annex). As excepted from a random initialization, the initial clustering may not be relevant for the classification task. Finally, and in order to quantify the class concentration described in Section 4.4.1, we display the entropy over classes, Figure 4.10b), showing a decrease of the average entropy with the layer depth, starting roughly in the middle of the network architecture. This decrease is due to the fine-tuning process as we observe a slighty increasing of it

NetA	NetB	mean ℓ_2 norm
Pretrained on ImageNet	FT on RASTA (Mode A training 1)	1.26
FT on RASTA (Mode A training 1)	FT on RASTA (Mode B training 1)	1.24
FT on RASTA (Mode A training 1)	FT on RASTA (Mode A training 2)	1.23
The end from scratch	Its Random initialization	6.52
From scratch	Its Random initialization	8.13

Table 4.8: Mean over all layers of the ℓ_2 norm of the difference between convolutional kernels between two models.

in the case of the model pretrained on ImageNet (see Figure B.6a in Annex). Moreover the decreasing of the entropy is also observable for model trained from scratch (see Figures B.6b and B.6c in Annex).

4.4.4 From One Art Dataset to Another

In this last section, we look at the different models trained on small datasets of object classification. We compare the ImageNet model fine-tuned and the double fine-tuning strategy (first using RASTA, then using the target dataset). The performances of the different methods may be found in Section 4.3.2.2. In Table 4.9, we use the two previously mentioned metrics to compare the different models fine-tuned on the IconArt and Paintings datasets. The model fine-tuned on a small art dataset will stay similar to its ImageNet pretrained initialization (with a CKA of 0.89 or 0.91 for the IconArt and Paintings datasets). After a first fine-tuning on the large RASTA dataset, the network changes more ($CKA = 0{,}77$ and ℓ_2 norm $= 1.26$). A double fine-tuning enables to go even further from the original pretrained weights ($CKA = 0{,}73$ and $0{,}76$). As already mentioned, this method provides the best classification performance.

In the case of the model trained from scratch (two last lines of Table 4.9), the change between initialization and the final model is also large due to the randomness of the initialization but those models are worst in terms of classification. It should be noted, however, that the CKA is only a criterion of similarity and not a proxy for the performance of classification. For instance it can be dependent on the learning rate used during training as it can be seen with the Mode F case which modifies a lot the network without providing good performance.

Nets \ Small art dataset used:		IconArt		Paintings	
NetA	NetB	mean CKA	mean ℓ_2 norm	mean CKA	mean ℓ_2 norm
Pretrained on ImageNet	FT on small art dataset (Mode A)	0.90	0.14	0.91	0.15
Pretrained on ImageNet	FT on small art dataset (Mode F)	0.61	1.72	0.65	1.18
Pretrained on ImageNet	FT on RASTA (Mode A) + FT on small dataset (Mode F)	0.73	1.61	0.76	1.67
FT on RASTA (Mode A)	FT on RASTA (Mode A) + FT on small dataset (Mode F)	0.79	0.78	0.77	0.86
The end from scratch on RASTA	The end from scratch on RASTA + FT on small dataset	0.70	0.91	0.72	1.01
From scratch on RASTA	From scratch on RASTA + FT on small dataset	0.83	0.27	0.79	0.52

Table 4.9: Mean linear CKA (on IconArt or Paintings test set) and mean ℓ_2 norm between models based on InceptionV1.

The optimized images of the model pretrained on ImageNet, fine-tuned first on RASTA [Lecoutre et al. 2017] and then on IconArt, show that some of the filters learned on RASTA can be directly reused for the IconArt dataset such as a nudity detector (Figure 4.11j). Although this class does not exist in the RASTA dataset, images with

such visual pattern belong to this dataset. Some filters are adapted to focus more on a given visual pattern, for instance the blue drapery detector (Figure 4.11k). This detector is not learned by the model with only fine-tuned on IconArt (see Figure 4.11e). Other filters are completely changed such as the tree in front of a blue sky detector (Figure 4.11l). The datasets used are not representative of all the artistic production by far and are certainly biased. However, these illustrations allow us to highlight what kind of filters can be relevant for artwork classification.

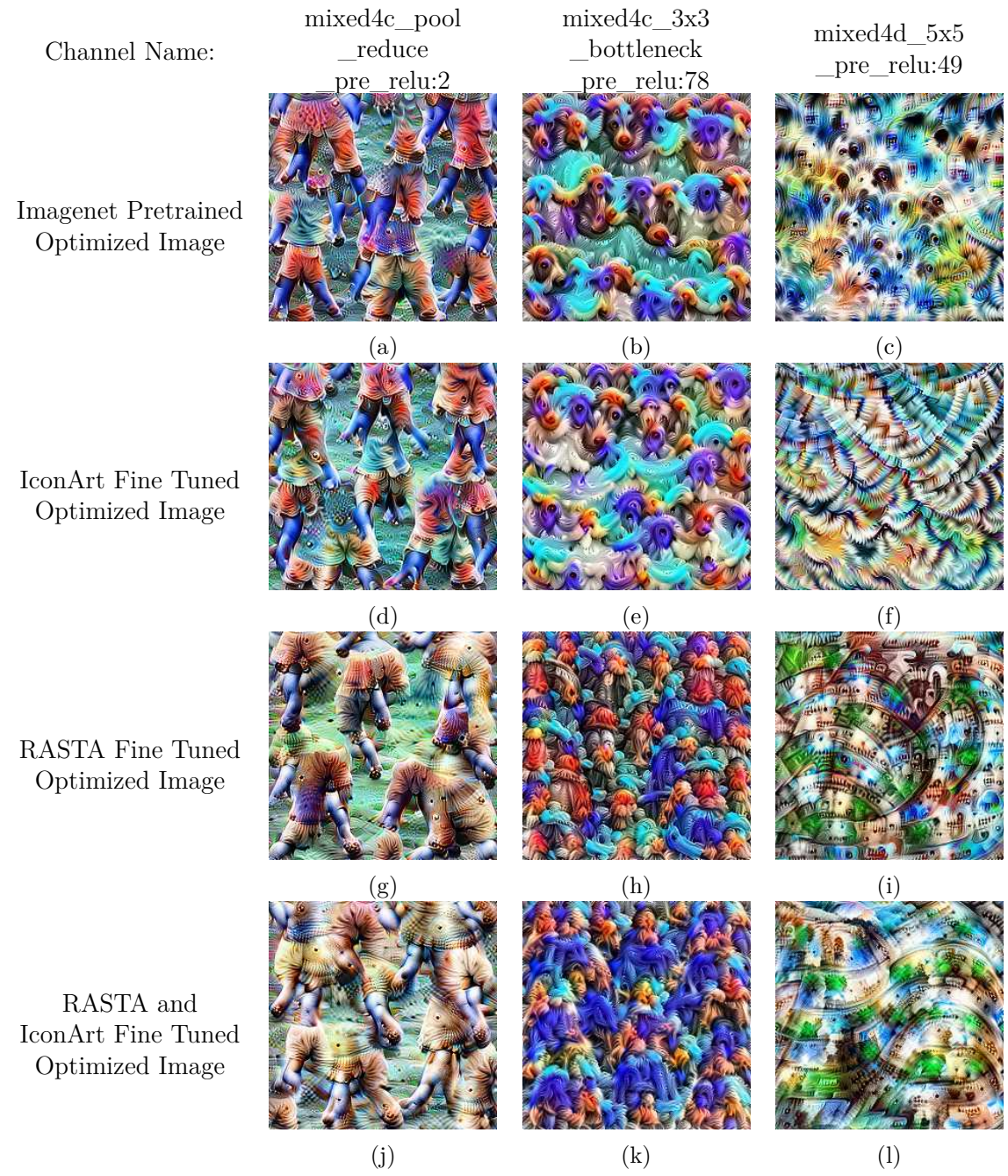


Figure 4.11: Optimized Image for one Individual channel from different mid-level layers. First row InceptionV1 pretrained on ImagneNet, second row fine-tuned on IconArt, third row on RASTA, last row fine-tuned first on RASTA [Lecoutre et al. 2017] then on IconArt.

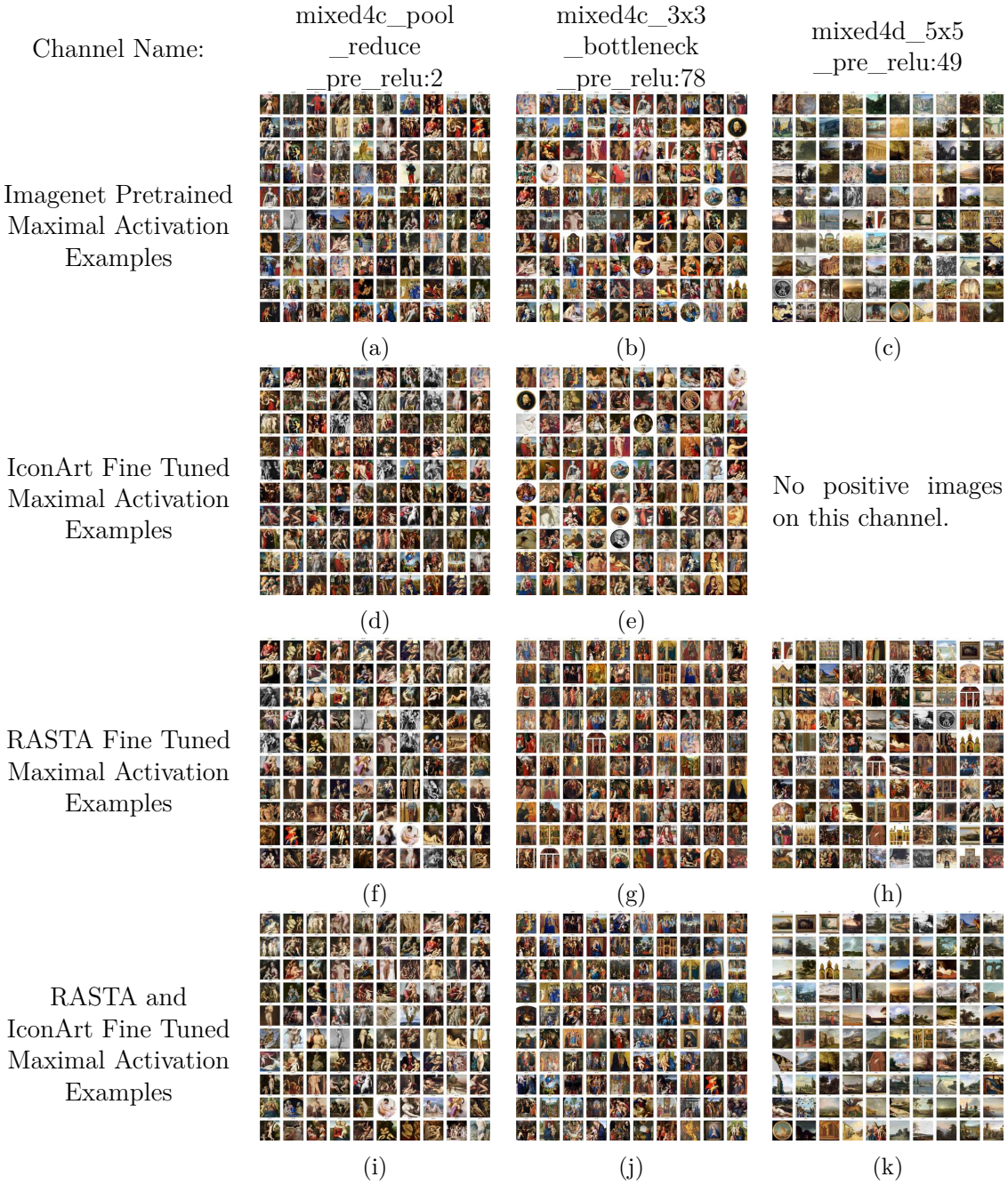


Figure 4.12: Maximal Activation Examples (from IconArt train set) the channel corresponding to Figure 4.11 channels for an InceptionV1 learned on ImagneNet then fine-tuned on RASTA [Lecoutre et al. 2017] and finally on IconArt.

4.5 Conclusion

In this chapter, we studied the different setups of transfer learning for art classification and shown that the fine-tuning of a pretrained model is the best solution. Then, we have investigated the effect of fine-tuning a network pre-trained on ImageNet using artistic dataset. We made use of visualization techniques and quantitative assessments of the changes of the networks. Among other things, we have shown that some of the mid-level layers of the networks exhibit easily recognizable patterns that appear to be more related to art images than the patterns learned on natural images, while lower layers of the network are hardly changed. We have also shown that higher layers tend to concentrate classes after fine-tuning. Eventually, we have also shown that a double fine-tuning involving a medium size artistic dataset can help the classification of small-size artistic datasets and produces visual patterns more related to the domain. The classification tasks between the two artistic datasets do not need to be identical for the double fine-tuning to be helpful. In our case, the intermediate task is style classification whereas the final one is object classification. This study provides good insights on the way networks are modified by fine-tuning in the case of artistic databases.

5

Multiple Instance Model and Features Extraction for Weakly Supervised Object Detection in Artworks

Contents

Abstract

This chapter introduces a Multiple Instance Learning method applied on deep features for Weakly Supervised Object Detection in artworks. The Weakly Supervised Object Detection task is typically addressed with a domain-specific solution focused on natural images. We show that a simple multiple instance approach applied on pre-trained deep features yields excellent performances on non-photographic datasets, possibly including new classes. This approach does not include any fine-tuning or cross-domain learning, which makes it efficient and possibly applicable to arbitrary datasets and classes. We investigate several flavors of the proposed approach, some including multi-layers networks and polyhedral classifiers.

The work in this chapter has led to the publication of a conference paper:

- Gonthier N., Gousseau Y., Ladjal S., Bonfait O. *Weakly Supervised Object Detection in Artworks*; Workshop on Computer Vision for Art Analysis, ECCV, 2018.

An extended version of it is under review:

- Gonthier N., Ladjal S., Gousseau Y. *Multiple instance learning on deep features for weakly supervised object detection with extreme domain shifts*; submitted at Computer Vision and Image Understanding in 2020.

5.1 Introduction

Given the lack (or prohibitive cost) of bounding box annotations and the proven transferability of CNN features we proposed, in this chapter, to use the Multiple Instance Learning (MIL) paradigm to tackle the Weakly Supervised Object Detection (WSOD) task on artistic images. Our simple MIL model is applied directly to the deep features of a pre-trained network. We choose not to use any fine-tuning of a big network in order to both save time and avoid overfitting. We introduce a Multiple Instance perceptron that is much lighter than the classical SVM approaches [Andrews et al. 2003]. In Section 5.2, we fully develop this approach, exploring several extensions of the model such as a multi-layer version and a polyhedral version (obtained by aggregating several linear classifiers to get a more complex classifier at a reasonable computational cost). We take interest in WSOD in the case of extreme domain shifts, namely non-photographic images especially artistic one, possibly addressing the detection of new, never seen classes. We evaluate the performances of our approach by comparing it to several state-of-the-art approaches on databases with challenging domain shifts, including paintings, drawings and cliparts, in the experimental Section 5.3.2. Our approach does not involve any cross-domain learning step and can therefore be applied to arbitrary datasets and classes. The approach outperforms methods specially developed for the considered databases, as well as classical MIL approaches and some state-of-the-art WSOD approaches, as we will see in the experimental section, such a strategy also enables one to have relatively small training times (Section 5.3.2.4). At last, our method is evaluated on classical MIL benchmarks in Section 5.3.5.

5.2 Proposed Multiple Instance Model: MI-max and its Variants

Reminder of the notations The feature vector associated to the instance k of the bag i is denoted $X_{i,k} \in \mathbb{R}^d$ and the label of the bag i is denoted $Y_i \in \{-1, 1\}$.

5.2.1 The Main Model (MI-max): a Multiple Instance Perceptron

Instead of the classical SVM generalization for MIL proposed in [Andrews et al. 2003] and based on costly iterations of SVM, we propose a simple model which is a multiple instance extension of the perceptron [Rosenblatt 1958] with a maximum taken over the instances of a bag. Our model can be seen as a Latent perceptron [Felzenszwalb et al. 2010]. It is also similar in spirit to the MI-network proposed by Zhou et al. [2002].

We denote our model **MI-max** As we consider each class individually, we focus on the case of binary classification. We build on a linear model:

$$f_w(X_{i,k}) = W^T X_{i,k} + b, \tag{5.1}$$

with $W \in \mathbb{R}^M$, $b \in \mathbb{R}$, which we combine with a maximum aggregation function $g = \max_{k \in \{1 \dots K_i\}}$ and a per example loss function equal to

$$l(y, \hat{y}) = 1 - y\,Tanh(\hat{y}) = 1 - Tanh(y\hat{y}). \tag{5.2}$$

We also use a regularization term on the norm of W and the weighting of the two classes, so that the complete loss function is:

$$\mathcal{L}(W,b) = 2 - \sum_{i=1}^{N} \frac{Y_i}{n_{Y_i}} Tanh \left\{ \max_{k \in \{1 \dots K_i\}} \underbrace{\left(W^T X_{i,k} + b\right)}_{= f_w(X_{i,k})} \right\} + C||W||^2, \tag{5.3}$$

with n_{Y_i} the number of positive examples in the training set when $Y_i = 1$ and the number of negative examples when $Y_i = -1$. The intuition behind this formulation is that minimizing $\mathcal{L}(W,b)$ amounts to seek a hyperplane separating the most positive element of each positive image from the least negative element of the negative image (i.e. from all examples in the negative bags). Also this loss seeks to maximize the margin.

The regularization term apart, this loss function equals 0 if and only if the classification is perfect. In the worst-case scenario, its value is 4 (plus the regularization term). Furthermore, if the hyperplane $W^T X + b = 0$ exactly separates the most positive examples of each positive bag from the set of all examples of all negative bags, then replacing C, W and b by $\lambda C, \frac{1}{\lambda} W$ and $\frac{1}{\lambda} b$ respectively and taking λ to 0 will lead to a loss as close to 0 as desired. This implies that if the MIL problem admits an exact linear solution, then our loss accepts it provided C is small enough. One advantage of this formulation is that it can be tackled by a simple gradient descent, therefore avoiding the very costly iterative procedures of other MIL solutions such as [Andrews et al. 2003]. Taking the max over all instances of a bag is akin to what is done in MI-SVM (mentioned in Section 2.4.1.1) when after each full training of an SVM, a new representative element of each bag is selected for the next SVM training. After each step of the gradient descent, we do a re-labeling of the instance through the max operator instead of doing so after a full optimization of a SVM.

We can switch to a Stochastic Gradient Descent (as in [Felzenszwalb et al. 2010] for the LSVM) by iterating on random batches when the dataset is too big. Of course, since our loss is not convex, the proposed method does not guaranteed to find the global minimizers of the function. To tackle this problem, we train r times the model with a random initialization and pick the best one on the training set evaluation of the loss function. When the best couple $(W^\star, b^\star)$ has been found, we compute the following score, that reflects the meaningfulness of category association:

$$S(x) = Tanh\{W^{\star T} x + b^\star\}. \tag{5.4}$$

The approach is then straightforwardly extended to an arbitrary number of categories, by computing $W^\star$, $b^\star$ and the score separately for each category. Observe, however, that this leads to non-comparable scores between categories.

Some previous works [Alain et al. 2017] shown that CNNs learn to find a low-dimensional representation in which the data are linearly separable, so the linear model is adapted to our approach.

Our method shares the idea of using a maximum function as aggregation function with numerous works [Zhou et al. 2002; Nguyen et al. 2009; Oquab et al. 2015] including the Nguyen et al.'s model [2009] which alternates between updating the SVM parameters and the regions label by taking the maximum over regions only for positive images.

5.2.2 One Hidden Layer Multiple Instance Network

In this extension, called **MI-max-HL**, the bare features $X_{i,k}$ are transformed by a hidden layer before the MI-max approach is applied. This can be summarized by modifying the function f_w as follows:

$$f_w(X_{i,k}) = \Omega^T \left(Tanh \left(W^T X_{i,k} + b \right) \right) + \beta, \tag{5.5}$$

with $W \in \mathbb{R}^{L \times M}$, $b \in \mathbb{R}^L$, $\Omega \in \mathbb{R}^L$, $\beta \in \mathbb{R}$ and L the dimension of the hidden layer.

When compared with MI-max the parameters to be learned are Ω, β, W, b for a total dimension of $L + 1 + L \times M + L = L \times (M + 2) + 1$ compared to the original $M + 1$ scalars. Here again we run r times the model and pick the best one on the training set evaluation of the loss function. For the moment, we do not provide an effective implementation of this method. This method is hundred times longer than the MI-max model because we are not able to train at the same time all the classes and all the initialization. Moreover this version has a bigger memory footprint.

5.2.3 Multiple Linear Classifier: a Polyhedral Model

5.2.3.1 Our model

Another possible improvement of the linear model is to learn several hyperplanes in parallel, so that the binary classification is performed in a collaborative manner instead of selecting the best hyperplane. The contributions of r hyperplanes are gathered with a maximum function, so that the model can be defined as:

$$f_w(X_{i,k}) = \max_{j \in \{1 \dots r\}} \left(W_j^T X_{i,k} + b_j \right). \tag{5.6}$$

At each iteration of the gradient descent, only one of the couples (W_j, b_j) is updated. For the inference the r hyperplanes are also used. The maximum function on the r classification scores is one of the simplest ways to aggregate the contributions of the different hyperplanes but there are no guarantees that hyperplanes will be distinct. Nevertheless on some datasets the performances are better with this variant compared to the plain model (Section 5.3.2.3). This model allows use to have more complex boundaries with a small extra cost compared to an SVM. This model defines a concave polyhedral boundary between the two classes, a particular case of a piece-wise linear boundary. Thus, this model is named **Polyhedral MI-max**.

5.2.3.2 Related Work about Polyhedral and Piecewise Linear Classifiers

In this section, we will present a short review of the polyhedral and piecewise linear classifiers proposed by the machine learning community.

Linear or piecewise linear boundaries between classes are ubiquitous in machine learning. In fact, many algorithms yield such a separation between the classes either directly (SVM, random trees...) or indirectly by aggregating linear classifiers to build a more complex one (Adaboost [Freund et al. 1995], Bootstrap aggregation [Breiman 1996]). Even a complex CNN only separates space by linear frontiers, as long as the only nonlinearities are ReLUs and classical poolings. Chu et al. [2018a] even propose

to transform a Piecewise Linear Neural Network[1] to its mathematically equivalent set of linear classifiers in order to study it.

The simplest algorithm that provides a piecewise linear frontier is the k-nearest-neighbor algorithm [Cover et al. 1967]. The axis-parallel rectangle classifier also provides a piecewise linear boundary [Dietterich et al. 1997].

In [Astorino et al. 2002], the concept of convex *polyhedral separability* is studied, i.e. a class is approximated by a convex polytope and the rest of the space is used to approximate the second class. An algorithm is proposed to learn the englobing polytope based on the iterative solution of a linear programming descent direction finding problem. In [Orsenigo et al. 2007], the optimization of the polyhedral separability is achieved by solving a mixed integer programming model that extends the notion of discrete SVM.

The related concept of *max-min separability* was introduced in [Bagirov 2005]. In this approach two sets are separated using a continuous piece-wise linear function, generalizing polyhedral separability since the convexity requirement is dropped. These two kinds of separability are illustrated in Figure 5.1. In [Bagirov et al. 2005] an efficient algorithm for solving the max-min separability problem is proposed. This algorithm is based on the use of a discrete gradient of the objective function to make the computation tractable.

Breiman [1993] proposes the hinging hyperplanes, which is a linear combination of basis functions of the form $max\{0, \ell_m(x)\}$ with ℓ_m an affine function. This also leads to a piecewise linear boundary. This model has been improved by Wang et al. [2005] in a generalized hinging hyperplanes model of the following form $\max\{0, \ell_1(x), \ldots, \ell_{k_n}(x)\}$. Huang et al. [2013] combine a piecewise linear feature mapping and an SVM to create an efficient classifier.

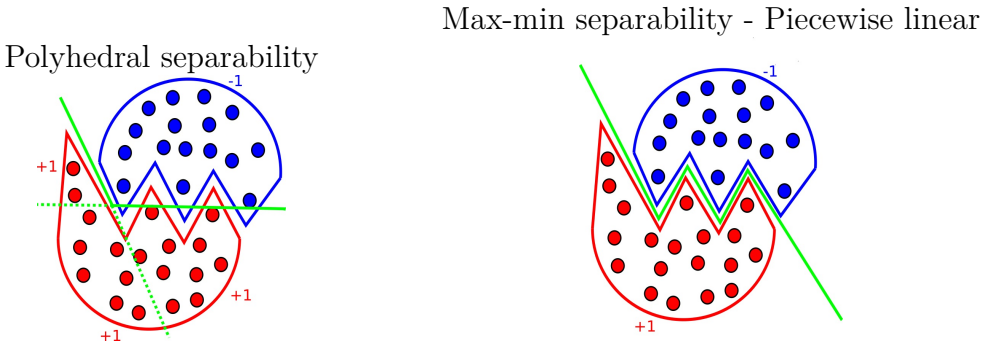


Figure 5.1: Illustration of the different kind of separability.

These three different models (linear, multi-layers and polyhedral) will be experimentally compared in Section 5.3.

[1]Only piecewise or linear activation are used.

5.2.4 From MIL to WSOD

In the context of WSOD, each bag i corresponds to an image and each instance k corresponds to a candidate region to be labeled. In a sense, we assume that the Region Proposal Network (RPN) is robust enough to transfer detection from photography to the target domain. One can see good bounding boxes for the category angel in Figure 5.2. The goal is then to decide which boxes correspond to the category. Following this "MIL assumption", our problem boils down to the classic multiple-instance classification problem [Dietterich et al. 1997]. Our work is focused on the instance-level classification and not bag level one.

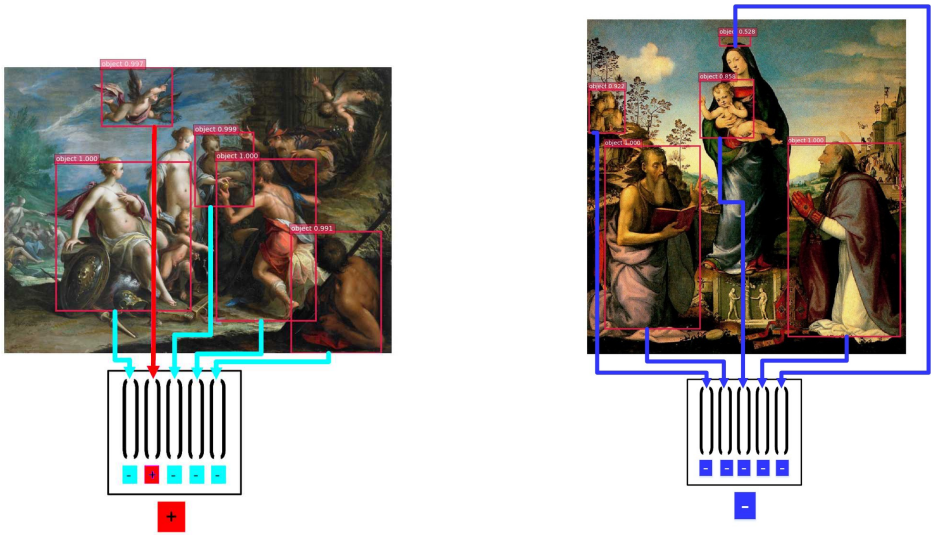


Figure 5.2: Illustration of positive and negative sets of instances (bounding boxes) for the *angel* category.

We use a CNN trained for a object detection task on natural images to provide candidate regions with an off-the-shelf feature vector per region. Training a WSOD model from scratch, especially when the database is rather small face overfitting and convergence problem. Thus, reusing as much as possible models that have been trained on large datasets is advisable. Referring to the WSOD standard pipeline mentioned in Section 2.4.2, we only focus on the multiple instance classification task and not on the box proposals algorithms, feature extraction or refinement methods. Our previously described approach enable to learn a linear separation between boxes. The classification score of a given box is defined in equation (5.4). A Non Maximum Suppression (NMS) algorithm is applied to avoid redundant detections. This algorithm takes into account the spatial information between boxes and the classification score.

Moreover, we here assume that candidate regions are returned by the detection CNN, together with a feature vector and a class-agnostic objectness score, but without any label. We ignore the classification predictions of the detection network. The idea is to give more importance to the classification of boxes with the highest objectness score. We observe that using the class-agnostic objectness score attached to each proposed box consistently give better results (see Section 5.3.3.1). Before taking the maximum,

each $W^T X_{i,k} + b$ is multiplied by the objectness score of the region k:

$$f_w(X_{i,k}) = (s_{i,k} + \epsilon) \left(W^T X_{i,k} + b \right), \tag{5.7}$$

with $\epsilon \geq 0$ and where $s_{i,k}$ is a class-agnostic objectness score of the region k, as returned by the detection network. Hence, the loss function defined in equation (5.3) can be written as:

$$\mathcal{L}(W, b) = 2 - \sum_{i=1}^{N} \frac{Y_i}{n_{Y_i}} Tanh \left\{ \max_{k \in \{1 \ldots K_i\}} \underbrace{(s_{i,k} + \epsilon) \left(W^T X_{i,k} + b \right)}_{= f_w(X_{i,k})} \right\} + C||W||^2. \tag{5.8}$$

The motivation behind this formulation is that the score $s_{i,k}$, roughly a clue that there is an object in box k, provides a prioritization between boxes. The same idea is used in the WSDDN model [Bilen et al. 2016] or in MELM [Wan et al. 2018]. At test time, the instance-level decision is made as before according to the sign of $\left(W^{\star T} x + b^\star \right)$, since multiplication by a positive score does not change the sign. Indeed, the hyperplane $W^\star, b^\star$ is chosen to separate two classes and the loss $\mathcal{L}$ aims at maximizing the margin with respect to this hyperplane. It stands to reason that the instance-level classification must be related to the relative position of the instance and the hyperplane. Nevertheless, we will propose in Section 5.3.2 a non-maximal suppression strategy that will once again use the objectness score to filter the boxes proposed for each class. More precisely the non-maximal suppression algorithm will use the following score:

$$S(x) = Tanh \left\{ (s(x) + \epsilon) \left(W^{\star T} x + b^\star \right) \right\} \tag{5.9}$$

which mixes the objectness score $s(x)$ and the signed distance from the learned hyperplane $W^{\star T} x + b^\star$. This resulting multiple instance learning model is our default model, previously called MI-max. Please note that otherwise mentioned, the model is used with the objectness score when it is available for WSOD case. For classical MIL tasks, there is no objectness score available.

The objectness score could be used in different manners. For example, we could multiply this score with the $Tanh$ operator instead of doing the multiplication inside of it:

$$f_w(X_{i,k}) = (s_{i,k} + \epsilon) \, Tanh \left(W^T X_{i,k} + b \right). \tag{5.10}$$

In this case, a region will need to have a strong objectness score and a strong classification score to be considered as a positive or negative instance for the learning process. It will discard the low objectness score region even if they have a high classification score.

Another way is to compute a linear combination between the objectness score and the classification one to force the model to select regions with both a high objectness score and a high classification one as in the following equation:

$$f_w(X_{i,k}) = \lambda \, Tanh \left(W^T X_{i,k} + b \right) + (1 - \lambda) \, s_{i,k} \, \text{sign} \left(W^T X_{i,k} + b \right), \tag{5.11}$$

with $\lambda \in [0, 1]$. This linear interpolation between the objectness score and the classification one have already been used in other WSOD models ([Tang et al. 2017c; Uijlings et al. 2018] during training and [Zhong et al. 2020] during inference).

The different use of the objectness score will be compared in Section 5.3.3.1.

5.2.5 Concerning the *Tanh* Based Loss Function

The goal of our *Tanh* based loss function defined in equation (5.2) is to mimic the margin of the SVM because the function saturates at -1 and 1. Thus the elements far from the border will not contribute to the gradient. Rosasco et al. [2004] show that assuming the loss function can be written as $L(y, f) = L(yf)$, is convex and is decreasing in a neighborhood of 0 then the minimizer of the expected risk equals the Bayes optimal solution (in case of infinite data). The hinge loss is the one for which they obtained the tighter bounds (in case of finite data). As our loss function can be rewritten as $1 - Tanh(yf(x))$ and respects the condition of the Theorem 2.1 of Rosset et al. [2004] then our loss function is margin-maximizing. Moreover as *Tanh* is 1-Lipschitz continuous as the hinge loss, it has the same theoretical convergence rate according to Rosasco et al. The usual loss functions and our *Tanh* based one can be seen in Figure 5.3. Our *Tanh* based loss function is a convex upper bound of the classification $Misclassification\ (0-1)\ loss : L(f(x), y) = I(yf(x) \leq 0)$ such as hinge loss and squared error. It will asymptotically be as the hinge loss for the well classifier point (above 1) where it will treat all the wrong classifier equally (under -1) whereas the hinge loss function will treat them linearly.

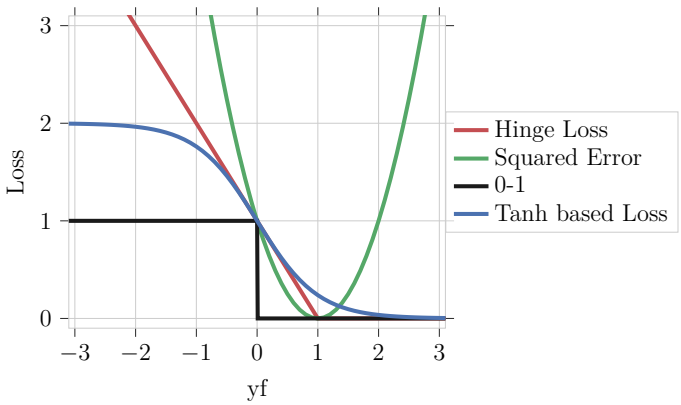


Figure 5.3: Common loss functions used in machine learning.

We experimentally show that the *Tanh* based loss function is as good as the hinge loss (Section 5.3.3.4).

5.3 Evaluation

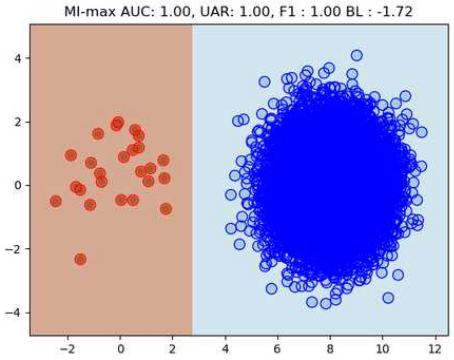
Reminder of the notations The names of our three models are:

- MI-max: the multiple instance perceptron
- MI-max-HL: the multiple instance neural network with one hidden layer
- Polyhedral MI-max: the multiple instance polyhedral perceptron

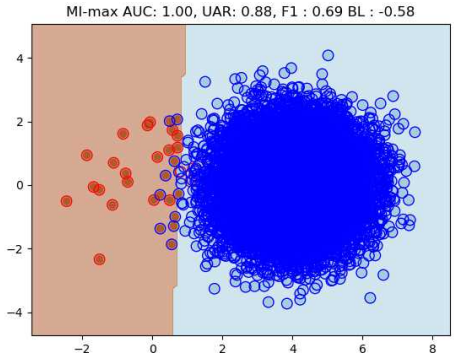
5.3.1 Toy Problem

First, we try our models on a simple Gaussian toy problem in two dimensions without or with overlapping. Two point clouds are generated and the bags of elements are created with one positive element out of one hundred for positive bags and none for the negative bags. The train and test sets are the same size: 25 positive bags and 125 negative ones. Figure 5.4 shows that our model MI-max is able to linearly separate two classes even when there is a little bit of overlapping. As expected, a linear model is not able to deal with not linearly separable data. It can be noticed in Figure 5.5 the MI-max-HL model provides a non-linear border as expected. The model Polyhedral MI-max can provide multiple distinct hyperplanes to split the space (see Figure 5.6) but fails when the two classes are too close. Moreover the same kind of result can be obtained even if the training set only contains one positive bag during the training.

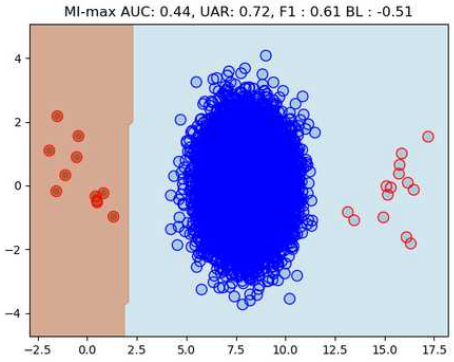
(a) Without overlapping



(b) With overlapping



(c) Without overlapping and non-linearly separable points



(d) With overlapping and non-linearly separable points

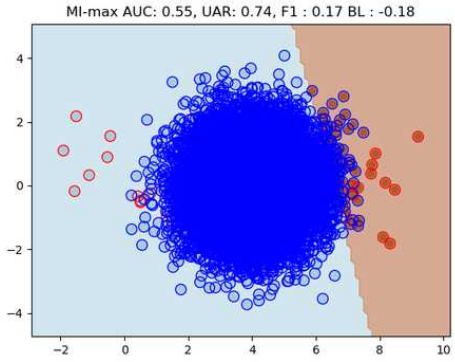


Figure 5.4: Toy problem with MI-max. Only the test set is shown. The background color is the class provided by the model whereas the circle color is the true label (red for positive and blue for negative).

(a) Without overlapping

(b) With overlapping and non-linearly separable points

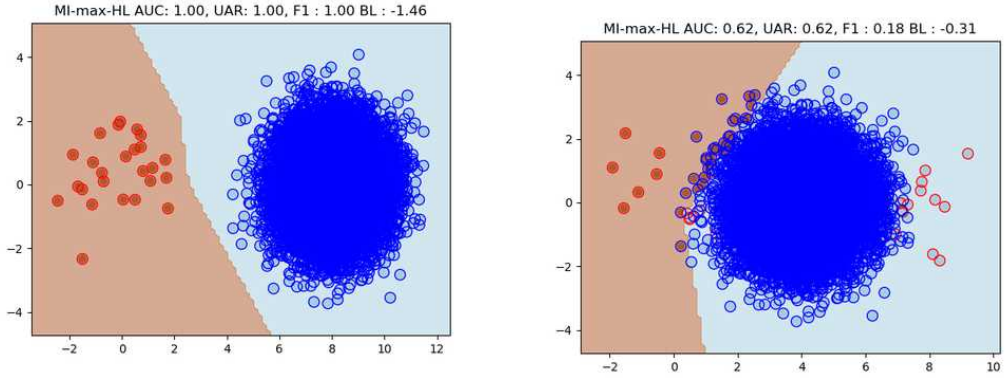


Figure 5.5: Toy problem with MI-max-HL. Only the test set is shown. The background color is the class provided by the model whereas the circle color is the true label (red for positive and blue for negative). The non-linear boundaries can be observed.

(a) Without overlapping and non-linearly separable points

(b) With overlapping and non-linearly separable points

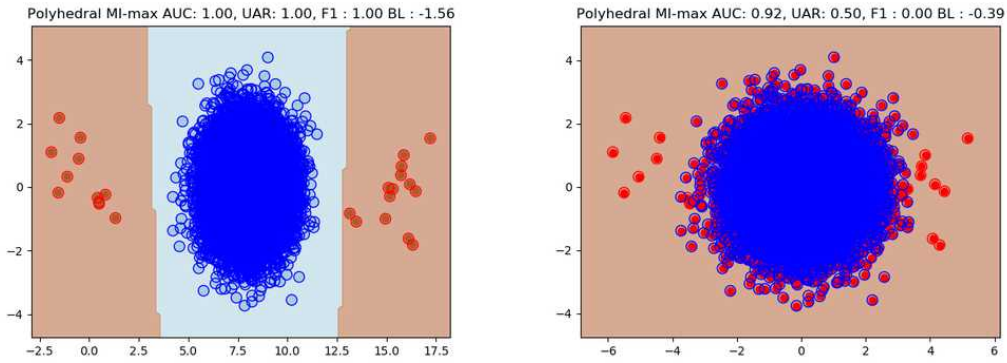


Figure 5.6: Toy problem with Polyhedral MI-max. Only the test set is shown. The background color is the class provided by the model whereas the circle color is the true label (red for positive and blue for negative). On the left, we can see the two hyperplanes which provide a good separation.

5.3.2 Benchmark on the Weakly Supervised Object Detection Task

5.3.2.1 Implementation Details

Faster R-CNN We use the detection network Faster R-CNN [Ren et al. 2015]. For our method we only use its RPN and the features corresponding to each proposed region. Recall that in order to yield an efficient and flexible learning of new classes, we decide to avoid retraining or even fine-tune the network. We explore the transferability of these learned representations in the scenario where the user gives simple hints about the presence of an object in a small number of artworks. The images are resized to a

maximal size of 600 by 1000 pixels before applying the Faster R-CNN model. After the NMS phase, the 300 boxes with the best "objectness" scores are kept among the boxes returned by Faster R-CNN along with their high-level features[2]. An example of extracted boxes is shown in the Figure 5.7. Observe, however, that since we are relying on Faster R-CNN, our system uses a subpart trained on databases including bounding boxes, but without using any class information. To be in the "common" WSOD setup evaluated on natural images benchmark (as in [Bilen et al. 2016; Zhu et al. 2017b; Tang et al. 2018a]) the model should have never seen any bounding boxes even on a different modality. In some sense, we are halfway between the "common" WSOD setup, the cross domain (as in [Inoue et al. 2018]) one and the knowledge transfer one. Indeed, our method allows considering new visual categories that are not present in the source domain (here MS COCO which has been used for training the detection CNN before its transfer), in contrast to Inoue et al. [2018]. Thus, our approach belongs to the weakly supervised paradigm as we tried to infer information that is not explicitly provided in the training set (i.e. the bounding box around our object of interest in the artworks), but we build on the knowledge from a previous and related task thus our method belongs to the WSOD-with-transfer paradigm mentioned in Section 2.4.2.5. Nevertheless, we do not explicitly exploit the relation between the two tasks which is specific to the knowledge transfer paradigm.

According to Kornblith et al. [2018], ResNets appears to be the best architecture for transfer learning by feature extractions among the different ImageNet models, and we therefore choose these networks as backbone for the Faster R-CNN algorithm. One of them (denoted RES-101-VOC07) is obtained with a ResNet 101 layers, trained for the detection task on PASCAL VOC2007 trainval set. The other one (denoted RES-152-COCO) is a ResNet 152 layers trained on MS COCO training set [Lin et al. 2014]. This last backbone seems to provide better features for the classification task as one can see in Section 4.3.1.2. We will also compare our approach to the plain application of these networks trained on detection tasks in a fully supervised manner when applicable. Thus they were trained on classes we want to detect and on images from the same domain than the test set. We refer to these approaches as Fully Supervised Detection (FSD) in our experiments. For implementation, we build on the Tensorflow implementation of Faster R-CNN of Chen et al. [2017][3].

MI-max and variants When a new class is to be learned, the user provides a set of weakly annotated images. The MI-max framework described before is then run to find a linear separator specific to the class. Note that both the database and the library of classifiers can be enriched very easily. Indeed, adding an image to the database only requires running it through the Faster R-CNN network and adding a new class only requires a MIL training.

Parameters of the models For training our MIL models, we use a batch size of 1000 examples (for smaller sets, all features are loaded into the GPU), 300 iterations of gradient descent for the linear model, performed with a constant learning rate of 0.01

[2]The layer $fc7$ of size $M = 2048$ in the ResNet case, often called 2048-D.

[3]This implementation code can be found on GitHub: `https://github.com/endernewton/tf-faster-rcnn`.

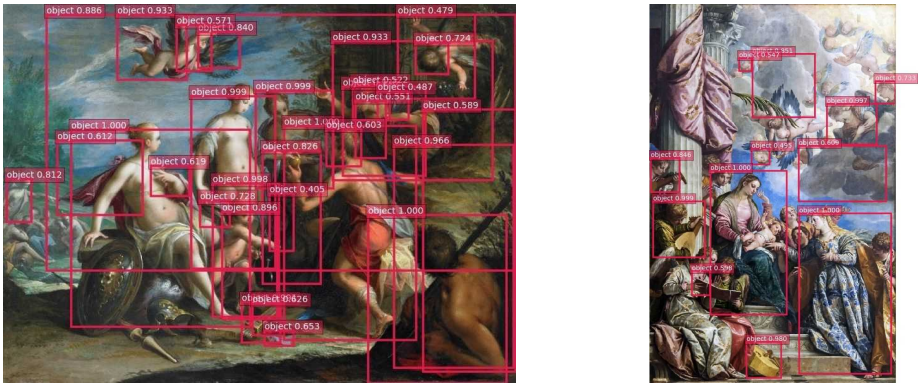


Figure 5.7: Some of the regions of interest generated by the RPN of Faster R-CNN.

and $\epsilon = 0{,}01$ and $C = 1$ (equations (5.7) and (5.3)). The complete training takes about 6 minutes for 7 classes on the IconArt dataset (Section 4.2.1) with 12 random starting points per class using a consumer GPU (GTX 1080Ti). In the case of Polyhedral MI-max we used 3000 iterations which increase the training time to 1 hour. For MI-max-HL, we use a maximum batch size of 500 elements. Actually, for the MI-max and Polyhedral MI-max models, the random restarts and classes are performed in parallel to take advantage of the presence of the features in the GPU memory, thus reducing the GPU-Central Processing Unit (CPU) transfer times. Typically, the different classes can be learned in parallel on a standard GPU, due to the light weight of the model. One of the others advantages of not fine-tuning the network is that there is no need to store the weights of the newly trained model. It is sufficient to store the features that are computed once for all the images.

Comparison to the multiple instance neural network Our model MI-max-HL is the one of our models that is closest to the multiple instance neural networks proposed by Ramon et al. [2000] and Zhou et al. [2002], but these models involve a sigmoid activation and they are trained with a quadratic loss $l(y, \hat{y}) = (y - \hat{y})^2$ and no re-initialization ($r = 0$).

5.3.2.2 Sanity Check on PASCAL VOC 2007

Before testing our method on paintings, we start with a sanity check experiment on PASCAL VOC2007 [Everingham et al. 2010]. We compare our weakly supervised approach, MI-max and its variants, to the plain application of the fully supervised Faster R-CNN [Ren et al. 2015] and two simple weakly supervised procedures (MAX and MAXA). The method MAX keeps one feature vector per image and learns a linear SVM classifier that separates the positive vectors from the negative ones [Crowley et al. 2016]. The variant MAXA also keeps one vector per positive image but uses all vectors from the negative ones. In both cases a 3-fold cross validation is performed. We perform the comparison using two different pre-trainings, RES-101-VOC07 and RES-512-COCO, as explained in the previous section. We also compared our approach to the Fully Supervised[4] Faster R-CNN Detector fine-tuned on VOC07 or MS COCO,

[4]Fine-tuned with bounding boxes.

this approach is denoted FSD. In a variant of this approach, denoted "FSD w/o B", we remove the final layer of the Faster R-CNN in charge of the box regression and take the localization of the bounding box from the RPN.

Net	Method	mean
RES-101-VOC07	FSD [He et al. 2015]	**75.0**
	FSD w/o B	67.4
	MAX [Crowley et al. 2016]	36.2
	MAXA	58.9
	MI-max w/o score	61.8 ± 0.7
	MI-max	68.3 ± 0.2
	Polyhedral MI-max w/o score	62.5 ± 0.9
	Polyhedral MI-max	**69.0** ± 0.2
	MI-max-HL w/o score	60.7 ± 0.9
	MI-max-HL	**69.2** ± 0.4
RES-152-COCO	FSD [He et al. 2015]	**82.7**
	FSD w/o B	78.7
	MAX [Crowley et al. 2016]	44.1
	MAXA	65.5
	MI-max w/o score	73.9 ± 0.3
	MI-max	81.6 ± 0.3
	Polyhedral MI-max w/o score	76.7 ± 1.7
	Polyhedral MI-max	**82.3** ± 1.9
	MI-max-HL w/o score	71.0 ± 2.0
	MI-max-HL	78.8 ± 1.8

Table 5.1: **VOC 2007 test** Average precision (%). Comparison of the Faster R-CNN detector (trained in a fully supervised manner: FSD) and our MI-max algorithm (trained in a weakly supervised manner) for two networks RES-101-VOC07 and RES-152-COCO. In blue the best fully supervised method and in red the best weakly supervised one. When it was computationally acceptable we calculated the standard deviation on 10 runs.

In Table 5.1, one can see that our MI-max, Polyhedral MI-max and MI-max-HL proposed models are better than the two baselines MAX and MAXA, with or without the objectness score use. Especially the baseline MAX procedure (used for transfer learning on paintings in [Crowley et al. 2014]) yields notably inferior performances. MAXA is better than MAX because it is trained on a lot more negative examples. That's why it is more able to discriminate background. Moreover this shows that our MI-max model is better than the MI-max w/o score thanks to the weighting by the objectness score in the loss function. Similarly, the model with the objectness score is better than the one without it for the Polyhedral MI-max and MI-max-HL models. On the average, the loss is only 1.1% of mAP when using RES-152-COCO between the FSD and MI-max whereas it is 7% in the case of RES-101-VOC07. The gap is even lower between the Polyhedral MI-max approach and the FSD one, with 0.4% in the first case and 6% in the second. The MI-max-HL and Polyhedral MI-max models give the best weakly supervised performance with the RES-101-VOC07 features. The

features and the boxes extracted in the first case seem to be of better quality. This may be due to the fact that there are more examples in the MS COCO dataset. The MI-max, Polyhedral MI-max and MI-max-HL models are able to learn a classifier equivalent to the last layer that has been removed to the Faster R-CNN. We have to mention that our models are not completely weakly supervised and cannot be fairly compared to state-of-the-art WSOD methods (described in Section 2.4.2) because we are relying on Faster R-CNN. However, our system only uses a subpart trained using class agnostic bounding boxes. Our approach can be seen an unsupervised Knowledge Transfer because we don't take into account the knowledge about the transferred model as in [Uijlings et al. 2018]. This experiment is promising for the power of our method.

5.3.2.3 Results and Comparison to Other Methods on Artistic Datasets

In this section, we perform weakly supervised object detection experiments on different non-photorealistic databases, in order to illustrate different assets of our approach. We compare our different models MI-max, Polyhedral MI-max, MI-max-HL to three types of methods.

The first group of methods are those specifically targeted at WSOD using fine-tuned networks. We have included state-of-the-art methods for which a source code is available: Soft Proposal Network (SPN) [Zhu et al. 2017b] and Proposal Cluster Learning (PCL)[5] [Tang et al. 2018a]. For some of the datasets, we also include results from the WSDDN [Bilen et al. 2016] from [Inoue et al. 2018]. For three datasets we also show the performance obtained by the mixed supervised method with domain adaptation proposed by Inoue et al. [2018]. This method needs the bounding boxes for objects of the same classes in another modality. Moreover the domain adaptation for unknown classes is out of the scope of this work.

The second family of methods are generic MIL-methods directly applied to the set of deep features vectors generated by Faster R-CNN. Observe that these methods ignore the objectness scores returned by the detection network. The first ones are MI-SVM and mi-SVM from Andrews et al. [2003]. These two methods require to train several SVMs and are therefore costly. In some cases (for PeopleArt and IconArt) we were forced to perform a PCA on the training set to reduce the number of components from 2048 to around 650 dimensions by keeping 90% of the variance (to fit the SVM in CPU memory). We experimentally observe on the other datasets that this dimensionality reduction does not reduce the performances. Eventually, the computationally lighter MI_Net, MI_Net with Deep Supervision (DS) or Residual Connection (RC) and mi_-Net from Wang et al. [2018b] are also considered. Although these models are designed for a bag-level classification, we used them for instance-level prediction. This can be seen as variants on our proposed method (the weakly detection of objects is not addressed in [Wang et al. 2018b]).

The last type of methods are these which before any training use the objectness score of the proposed regions to keep only one feature vector for each positive image. They are the MAX and MAXA approaches defined before, in Section 5.3.2.2.

Observe that our method lies in between the generic MIL methods and the ones that take advantage of objectness information from deep models. At test time, the

[5]Trained with the following hyperparameters: batch size = 2, learning rate = 0.001, epochs = 13 and number of clusters by default.

labels and the bounding boxes are used to evaluate the performance of the methods in terms of AP per class. The generated boxes are filtered by a NMS with an IoU threshold of 0.3 and a confidence threshold of 0.05 for all methods.

Reference	Dataset	# Images in train set	# Images in test set	# Instances in test set	# Classes	Min # Images per class	Classes from natural images	Classes from Pascal VOC
[Westlake et al. 2016]	PeopleArt	3007	1616	1137	1	968	Yes	All
[Inoue et al. 2018]	Watercolor2k	1000	1000	3315	6	27	Yes	All
[Inoue et al. 2018]	Clipart1k	500	500	3615	20	21	Yes	All
[Inoue et al. 2018]	Comic2k	1000	1000	6389	6	87	Yes	All
[Thomas et al. 2018]	CASPA paintings	1045	1033	1486	36	8	Yes	6 out of 8
Section 4.2.1	IconArt	2978	1480	3009	7	75	No	No

Table 5.2: Overall information of the evaluated artistic datasets used for WSOD task.

As explained above, we concentrate on non-photographic databases for which a ground truth is available for object detection on the test set. We report in Table 5.3 the performances for the Weakly Supervised Object Detection task for 6 different non-photographic datasets: PeopleArt [Westlake et al. 2016], Watercolor2k, Clipart1k, Comic2k [Inoue et al. 2018], IconArt (Section 4.2.1) and "CASPA paintings" [Thomas et al. 2018]. "CASPA paintings" is the painting subset of the CASPA dataset[6] proposed in [Thomas et al. 2018] with bounding boxes associated to 8 visual categories (only animals) for most of the images. When the method is not too slow, we provide standard deviation and mean score computed on 10 runs of it. The detailed results can be found in Annex, Section C.2.

First, for all databases, the end-to-end weakly supervised methods (WSDDN, SPN and PCL) yield relatively poor results. Possible explanations are that the model overfits on the training set or that the model is stuck in bad local minima, so that the weakly supervised setting is not adequate with a relatively small training dataset. Maybe it should be necessary to add more regularization during the training of those models and to freeze some part of the deep network due to the small size of the training sets but this is out of the scope of this work. Moreover in the case of PCL, the boxes are proposed by the Selective Search algorithm [Uijlings et al. 2013] which, as shown in Table 5.8, completely fails on the considered non-photographic datasets. That alone can explain the poor results of PCL on those datasets. Moreover, these methods do not have the important advantage of using features from Faster R-CNN pretrained on MS COCO (thus with bounding box annotations).

When comparing the performances of the different multiple instance neural networks, we can see that MI_Net (Maximum Bag Margin Formulation) outperforms the other MIL networks on four datasets. Moreover this multiple instance neural network outperforms the multiple instance SVM (mi-SVM and MI-SVM), which can be due to the fact that linear SVM are not complex enough.

We can notice that the Maximum Pattern margin methods (mi-SVM and mi_Net) never perform better than the bag margin ones. This is rather unexpected since those models are designed to better take into account the whole positive bag by assigning an individual label per instance. These models appear to be badly suited for the task of weakly supervised detection in non-photographic databases. When comparing our MI-max and Polyhedral MI-max models to the baseline MAX and MAXA, we observe that our models consistently perform better. Nevertheless the MAXA model

[6]This dataset can be found here: http://people.cs.pitt.edu/~chris/artistic_objects/

Network	Method	Model	People-Art	Watercolor2k	Clipart1k	Comic2k	CASPA paintings	IconArt
VGG16-IM	Fully supervised fine-tuning [Westlake et al. 2016]	Fast R-CNN	**59***	•	•	•	•	•
SSD	Mixed supervised	DT+PL [Inoue et al. 2018]	•	**54.3***	**46.0***	**54.3***	•	•
Yolov2	with domain	DT+PL [Inoue et al. 2018]	•	•	39.9*	•	•	•
Faster RCNN	adaptation	DT+PL [Inoue et al. 2018]	•	•	34.9*	•	•	•
VGG16-IM	Weakly supervised fine-tuning	WSDDN [Bilen et al. 2016]	•	12.7	4.4	12.7	•	•
		SPN [Zhu et al. 2017b]	10.0	7.1	3.8	1.2	0.7	7.7
		PCL [Tang et al. 2018a]	3.4	0.0	1.2	0.0	0.0	5.9
RES-152-COCO	Off-the-shelf Features extraction	MAX [Crowley et al. 2016]	25.9	34.3	16.9	11.9	9.8	3.7
		MAXA	48.9	43.9	22.0	19.8	14.6	12.0
		MI-SVM [Andrews et al. 2003]	13.3	21.8	19.3	13.0	2.5	4.0
		mi-SVM [Andrews et al. 2003]	5.6	5.3	6.2	4.6	1.2	2.8
		MI_Net [Wang et al. 2018b]	33.0 ± 6.0	34.1 ± 1.0	29.7 ± 1.5	22.8 ± 1.1	11.7 ± 1.6	**15.1** ± 1.5
		MI_Net_DS [Wang et al. 2018b]	19.5 ± 11.4	26.8 ± 2.4	18.9 ± 2.4	19.6 ± 1.6	7.6 ± 1.2	13.0 ± 1.7
		MI_Net_RC [Wang et al. 2018b]	12.5 ± 8.3	16.7 ± 6.3	0.9 ± 0.8	11.4 ± 4.4	5.6 ± 2.1	12.7 ± 1.6
		mi_Net [Wang et al. 2018b]	26.5 ± 8.5	25.8 ± 3.5	29.5 ± 1.2	19.5 ± 2.1	8.4 ± 1.7	12.9 ± 1.2
		MI-max	55.5 ± 1.0	**49.5** ± 0.9	**38.4** ± 0.8	**27.0** ± 0.8	**16.2** ± 0.4	12.0 ± 0.9
		Polyhedral MI-max	**58.3** ± 1.2	46.6 ± 1.3	30.5 ± 2.3	23.3 ± 1.6	14.4 ± 0.7	13.0 ± 2.2
		MI-max-HL	57.3 ± 2.0	47.0 ± 1.6	33.0 ± 1.2	23.6 ± 0.5	14.4 ± 0.9	14.5 ± 1.8

Table 5.3: Detection Mean Average Precision (%). Comparison on six art datasets of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. The best fully supervised method is highlighted in blue, the best mixed supervised method is highlighted in green and the best weakly supervised one in red. Thus, the red, green and blue number are not fairly comparable. We use a grid search for MAX and MAXA approaches. When it was computationally acceptable we calculated the standard deviation on 10 runs. The Fast R-CNN performance comes from the Westlake et al.'s paper [2016]. DT+PL and WSDDN performances come from the Inoue et al.'s paper [2018].

performs well especially on the IconArt or CASPA paintings databases, probably for the previously mentioned reason. The MAX baseline sometimes provides equivalent performances to more complex methods (such as MI-SVM or MI_Net), this illustrates the fact that the objectness score (used for selecting candidates in MAX) contains useful information. Also observe that it is faster to train a multiple instance perceptron than several linear SVMs, as is needed for MI-SVM or mi-SVM. This is quantified in Section 5.3.2.4. Finally, we observe that both our models MI-max and Polyhedral MI-max provide better results than the other methods on PeopleArt, CASPA paintings, Comic2k, Clipart1k and Watercolor2k datasets. The dataset IconArt appears to be much more challenging. In this case, our multiple instance methods provide equivalent performances compared to the multiple instance networks. The best performance is obtained by the MI_Net, the MI-max-HL performance being very similar.

We propose a Multiple Instance Learning approach to transfer off-the-shelf deep features to datasets with extrem domain shift (on the contrary to the experiment from the previous Section 5.3.2.2).

5.3.2.4 Execution Time

One advantage of our method is the relative short time needed for training, as can be seen in Table 5.4. As can be expected, the SPN and PCL methods are the longest to train due to the fine-tuning of the whole network. Observe also that the training time for our method MI-max is almost independent of the number of classes and restarts, which is a strong advantage compared to the MI-SVM, mi-SVM, MI_Net and mi_Net models. These models all need one full training per class and per re-initialization. The SVM based methods are more costly because they don't take advantage of GPU computational power. Nevertheless, due to the aggregation of several hyperplanes with a maximum operator in the Polyhedral MI-max model, we need to do 10 times more epochs that when using MI-max, which explain the strong overload.

Method	Training Duration Duration	Linear to the number of classes	Linear to the number of restarts
No Boxes proposals			
SPN [Zhu et al. 2017b]	3000s (20 epochs)	No	•
Selective Search Bounding Boxes proposal	6600s		
PCL [Tang et al. 2018a]	12000s (13 epochs)	No	•
Faster R-CNN Features and boxes proposals	200s		
MAX with hyperoptimization	52s	Yes	•
MAXA with hyperoptimization	2000s	Yes	•
MI-SVM [Andrews et al. 2003]	3000s	Yes	Yes
mi-SVM [Andrews et al. 2003]	30000s	Yes	Yes
MI_Net [Wang et al. 2018b]	1200s (20 epochs)	Yes	Yes
MI_Net_DS [Wang et al. 2018b]	1800s (20 epochs)	Yes	Yes
MI_Net_RC [Wang et al. 2018b]	1600s (20 epochs)	Yes	Yes
mi_Net [Wang et al. 2018b]	1800s (20 epochs)	Yes	Yes
MI-max	130s (300 epochs)	No	No
Polyhedral MI-max	1100s (3000 epochs)	No	No
MI-max-HL	3000s (300 epochs)	No	Yes

Table 5.4: Execution time of the different models for datasets Watercolor2k and Comic2k, with 1000 images in the training set and 6 visual categories.

5.3.3 Fine Models Analysis

In this section we discuss the details of our model and some variations. In particular, an ablation study is provided where we analyze how the choices of a different losses, different set of features and use of the objectness score impact the performances of our models. In Sections 5.3.3.1, 5.3.3.2 and 5.3.3.4 a thorough investigation of the main parameters' influence is conducted. From this study we are able to recommend a set of parameters that are suited for our models, thus providing the user with a safe baseline for reusing them. Then, some limitations of our Polyhedral MI-max model are highlighted in Section 5.3.3.5. In Section 5.3.3.6, the generalization ability of our models are also evaluated across different modalities of images (using classes shared by the different datasets). We experimentally show that our method also enables to easily transfer the knowledge between datasets and artistic modalities. Finally, in Section 5.3.3.7 some visual results are commented to give an insight on the strengths and shortcomings of our model.

5.3.3.1 Concerning the Faster R-CNN Objectness Score Use

Dataset	MI-max				Polyhedral MI-max			
	Main model	Without score	With score mul outside *Tanh* (5.10)	With score addition (5.11)	Main model	Without score	With score mul outside *Tanh* (5.10)	With score addition (5.11)
PeopleArt	55.5 ± 1.0	0.9 ± 0.4	**57.5** ± 0.5	53.7 ± 3.3	**58.3** ± 1.2	10.1 ± 3.3	53.4 ± 2.3	55.0 ± 4.8
Watercolor2k	49.5 ± 0.9	32.8 ± 2.2	**50.7** ± 0.8	50.1 ± 0.8	**46.6** ± 1.3	18.3 ± 4.7	37.9 ± 3.4	42.8 ± 2.3
Clipart1k	**38.4** ± 0.8	24.2 ± 1.6	38.3 ± 0.9	35.4 ± 2.0	**30.5** ± 2.3	11.9 ± 2.6	24.6 ± 1.1	24.2 ± 1.9
Comic2k	27.0 ± 0.8	17.4 ± 1.5	**27.2** ± 0.8	24.0 ± 1.9	**23.3** ± 1.6	11.6 ± 2.8	18.6 ± 1.1	20.3 ± 1.8
CASPA paintings	16.2 ± 0.4	18.7 ± 0.8	**16.4** ± 0.1	16.0 ± 0.6	**14.4** ± 0.7	8.6 ± 1.4	12.1 ± 0.8	13.6 ± 1.0
IconArt	**12.0** ± 0.9	6.7 ± 2.5	**12.0** ± 1.1	11.7 ± 1.5	**13.0** ± 2.2	6.4 ± 2.3	8.6 ± 1.3	10.8 ± 1.9

Table 5.5: Mean average precision over the classes of the different datasets (%). Comparison of the proposed MI-max and Polyhedral MI-max methods with different ways to use the objectness score: this score can be used in a multiplicative way inside the *Tanh* operator referred as "Main model" (equation (5.9)), outside the *Tanh* operator (equation (5.10)) or in an additive way (equation (5.11)) with $\lambda = 0{,}5$. We also consider the baseline without objectness score. Standard deviation is computed on 10 runs of the method.

The first conclusion that can be drawn is that the use of objectness score almost always significantly increases the performances of our models. This is especially true for the PeopleArt dataset where the performances decrease drastically without using the objectness score. For the other datasets, the performances are always significantly lower without the objectness score. Note that for some classes the detection score drops due to the fact that the model detects parts of the object instead of the whole object when the objectness score is ignored. Among the three ways to use the objectness score (defined in Section 5.2.4), the main model is better or comparable to the two other methods especially for the Polyhedral MI-max model. The additive solution seems under the two other solutions. The landscape of the aggregated score provided by those three possibilities can be seen in Figure 5.8. The multiplication with the objectness score inside the *Tanh* is less restrictive than the multiplication outside. The second solution gives more importance to the objectness score. Whereas the additive solution can be seen as an intermediate solution (for $\lambda = 0{,}5$).

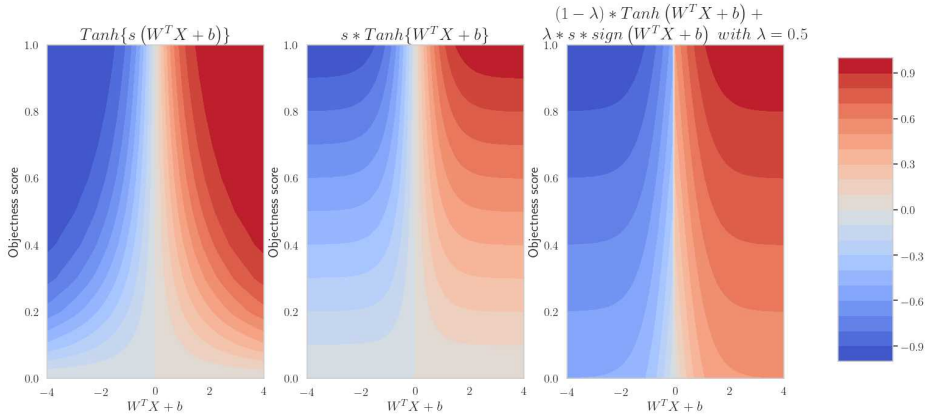


Figure 5.8: Different ways to use the objectness score: value of the global score. From left to right, we consider the multiplication with the objectness score outside the $Tanh$ (equation (5.9)), inside it (equation (5.10)) and the additive approach (equation (5.11)) with $\lambda = 0{,}5$.

We study this gain on performance especially on the PeopleArt dataset because the drop of performance is really significant. When we look at the best detected region per image on the train set of PeopleArt, MI-max w/o score model seems to detect upper body whereas the MI-max model gets the full body (see Figure 5.9). The same observation can be made on the test set Figure 5.10. Moreover we can see that the MI-max model will provide higher score for the same region sometimes (last row of Figure 5.10). It also outputs higher number of bounding boxes around the object of interest.

MI-max w/o score provides a classification score equivalent to the MI-max's one whereas it leads to bad detection score (see in Annex, Table C.7). As shown by Vanwinckelen et al. [2016], the bag-level classification score (here the image level classification score) is a bad proxy for the instance-level classification score (the detection score). However the objectness score is not a cure-all. In Figure 5.11, both models fail on the same image maybe due to the Faster R-CNN features.

The way the RPN works is not yet well understood as illustrated in [Rosenfeld et al. 2018]. Double an object within the image may completely change the predictions. However the RPN provides a good measure of the importance of a given box in the case of artistic images. It would be interesting to quantify the objectness score before using it as a ranking weight in order to avoid providing arbitrary too much weight to a given box. Moreover, developing an object proposal that generalizes well for artistic images as it's done for natural ones in [Wang et al. 2020] is a promising research direction.

5.3.3.2 Choice of the Loss Function

In Table 5.6, we gather different versions of the two models MI-max and Polyhedral MI-max with two possible modifications. We replace the $Tanh$ based loss in equation (5.2) by the hinge loss, and we suppress the objectness score in the loss function (see section 5.2.4) as in the previous paragraph. The conclusion is that replacing the $Tanh$ based loss function in equation (5.2) by a hinge loss $l(y, \hat{y}) = 1 - \max(0, 1 - y\hat{y})$ generally hinders the performances, except for two cases among the 12 cases studied. In

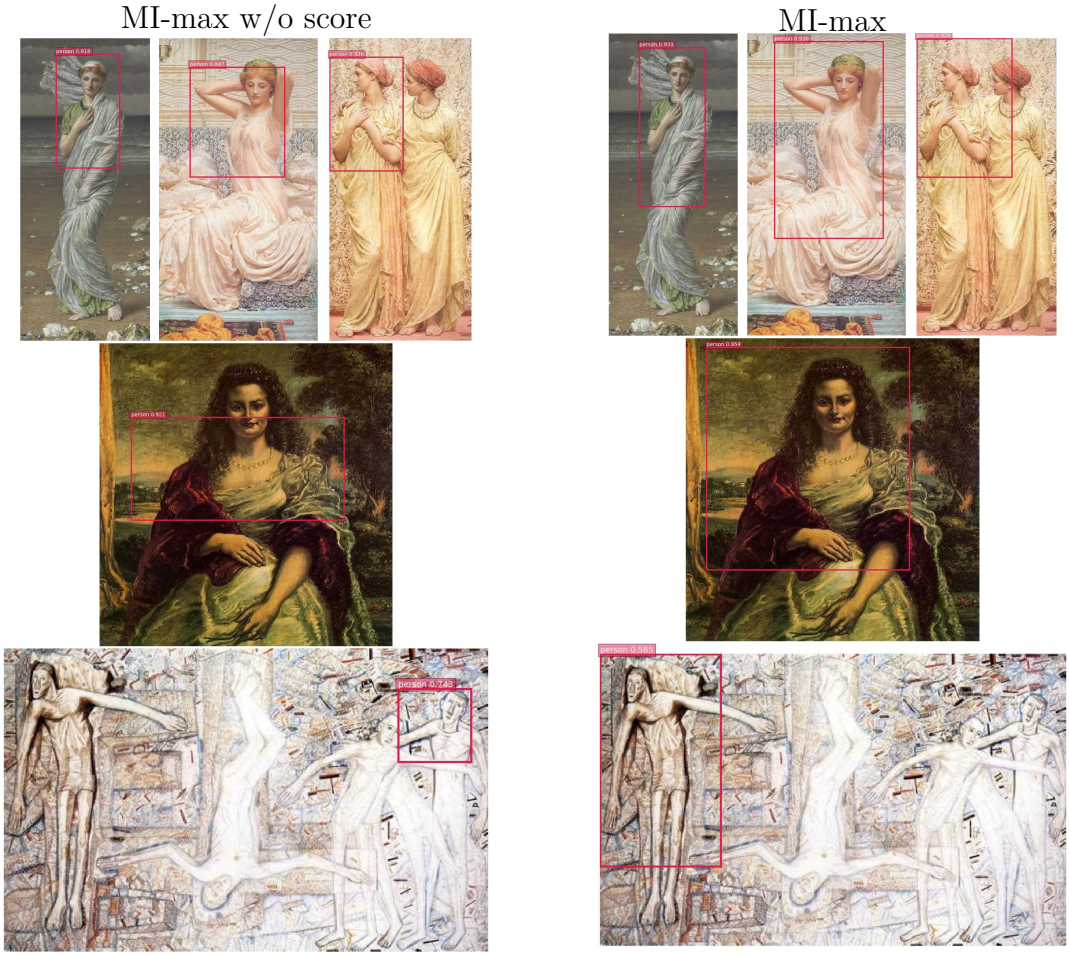


Figure 5.9: Boxes of the best element of the class on training examples with the knowledge of the ground truth class on PeopleArt dataset. In the case of the MI-max w/o score model, the upper body is detected whereas it is the full body with MI-max.

particular the Polyhedral MI-max method rarely benefits from a different loss function. Given the difficulty of the task, errors are likely to happen. The *Tanh* based loss may be more robust and forgiving than the hinge loss. Indeed, the hinge loss tries hard to correct any errors, especially those with a high negative margin.

Dataset	MI-max				Polyhedral MI-max			
	Main Model	Without score	Hinge loss	Without score and hinge loss	Main Model	Without score	Hinge loss	Without score and hinge loss
PeopleArt	55.5 ± 1.0	0.9 ± 0.4	**57.6** ± 1.0	1.7 ± 0.9	**58.3** ± 1.2	10.1 ± 3.3	56.6 ± 4.4	18.1 ± 8.6
Watercolor2k	**49.5** ± 0.9	32.8 ± 2.2	46.7 ± 1.5	33.8 ± 1.6	**46.6** ± 1.3	18.3 ± 4.7	37.5 ± 2.1	24.8 ± 3.3
Clipart1k	38.4 ± 0.8	24.2 ± 1.6	**34.8** ± 1.2	22.2 ± 1.8	**30.5** ± 2.3	11.9 ± 2.6	16.5 ± 1.2	5.1 ± 1.1
Comic2k	**27.0** ± 0.8	17.4 ± 1.5	25.5 ± 1.1	17.3 ± 1.1	**23.3** ± 1.6	11.6 ± 2.8	15.0 ± 1.8	9.5 ± 1.8
CASPA paintings	**16.2** ± 0.4	18.7 ± 0.8	16.1 ± 0.5	12.6 ± 0.9	**14.4** ± 0.7	8.6 ± 1.4	9.0 ± 0.9	3.2 ± 0.6
IconArt	12.0 ± 0.9	6.7 ± 2.5	**14.3** ± 2.1	8.2 ± 2.3	13.0 ± 2.2	6.4 ± 2.3	**13.3** ± 2.8	8.3 ± 2.0

Table 5.6: Mean average precision over the classes of the different datasets (%). Comparison of the proposed MI-max and Polyhedral MI-max methods with different settings. Standard deviation is computed on 10 runs of the method.

Figure 5.10: Comparison of MI-max w/o score and MI-max on PeopleArt test set. The first model only detects upper body.

5.3.3.3 Features Extraction and Region Proposals Choices

In this section, we investigate an alternative choice for the features and box proposals. The unsupervised algorithm EdgeBoxes [Zitnick et al. 2014] is used for box proposals and a ResNet-152 trained on ImageNet applied to each box for extracting features. By doing so we must drop the objectness score that is not included in the output of EdgeBoxes. Ideally, we would like to have an unsupervised feature and boxes proposals to get rid of the pretraining process. We can see in Table 5.7 the performances of the model MI-max (without the objectness score) using these features/boxes compared to the Faster R-CNN's features/boxes (without objectness score for fair comparison).

MI-max w/o score

MI-max

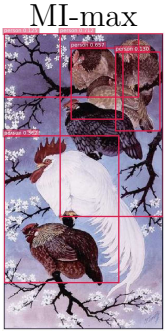


Figure 5.11: Boxes on test set: wrong detection for MI-max and MI-max w/o score models.

Regarding the detection task, the performances clearly drop when using EdgeBoxes. To further investigate this drop of performance we present in Table 5.8 the recall score of three box proposals methods (the percentage of ground-truth boxes that are present in the set of all proposed boxes with respect to an IoU > 0.5). We can see that EdgeBoxes performs very poorly on a dataset like PeopleArt and never matches the boxes proposed by Faster R-CNN. For the classification task, the MI-max method without objectness score performs honorably in this setting when compared to the use of Faster R-CNN's boxes/features (even slightly better on the IconArt database). This is another proof that bag-level classification (the aim of the training of a MIL algorithm) is not a good proxy for instance-level classification (which is the aim of a detection algorithm). The objectness score can be seen as a very helpful cue to guide the training of a WSOD method. Similar to the conclusions of Donahue et al. [2014] about the classification task, transfer learning of deep models trained for the detection task is the best way to obtain a detector on new domains even when we do not have the bounding box.

Dataset	Metric	Faster R-CNN	EdgeBoxes
PeopleArt	AP IuO $\geqslant$0.5	0.9 $\pm$ 0.4	0.0 $\pm$ 0.0
	Classif AP	92.5 $\pm$ 0.3	92.1 $\pm$ 0.2
Clipart1k	AP IuO $\geqslant$0.5	24.2 $\pm$ 1.6	3.1 $\pm$ 0.3
	Classif AP	59.4 $\pm$ 1.7	42.8 $\pm$ 1.3
Comic2k	AP IuO $\geqslant$0.5	17.4 $\pm$ 1.5	1.8 $\pm$ 0.3
	Classif AP	54.9 $\pm$ 2.0	47.9 $\pm$ 1.5
Watercolor2k	AP IuO $\geqslant$0.5	32.8 $\pm$ 2.2	2.7 $\pm$ 0.5
	Classif AP	78.0 $\pm$ 1.2	71.8 $\pm$ 1.3
CASPA	AP IuO $\geqslant$0.5	12.6 $\pm$ 0.5	0.3 $\pm$ 0.1
	Classif AP	48.6 $\pm$ 0.6	45.0 $\pm$ 1.2
IconArt	AP IuO $\geqslant$0.5	6.7 $\pm$ 2.5	5.3 $\pm$ 0.3
	Classif AP	60.4 $\pm$ 1.1	69.2 $\pm$ 0.3

Table 5.7: Average precision for detection and classification (%). Two different feature extraction methods are considered in this table (both without objectness score).

EdgeBoxes computes the edge of the images and then try to fit boxes that cover the highest regions without cutting too much edges with the bounding box boundaries. In some sense, this unsupervised method try to get regions with the maximum of energy from the gradient of the image. One can see images from the IconArt dataset with 30 or 300 boxes generated with EdgeBoxes. It seems that on most of the images the box proposals are not good enough for a detection task (Figure 5.12), the characters are not selected, the animals neither.

Dataset \ Box proposal	RPN of Pre-trained Faster R-CNN [Ren et al. 2015]	EdgeBoxes [Zitnick et al. 2014]	Selective Search [Uijlings et al. 2013]
PeopleArt	94.0	15.4	55.7
Clipart1k	91.4	14.4	49.4
Comic2k	82.7	54.1	46.2
Watercolor2k	93.6	61.4	56.8
CASPA	76.6	34.3	51.6
IconArt	75.9	60.0	56.9
Number of boxes	300	300	3000-5000

Table 5.8: Recall (%) at IuO $\geqslant$0.5 of the boxes proposals for the different methods and databases. Mean over the classes.

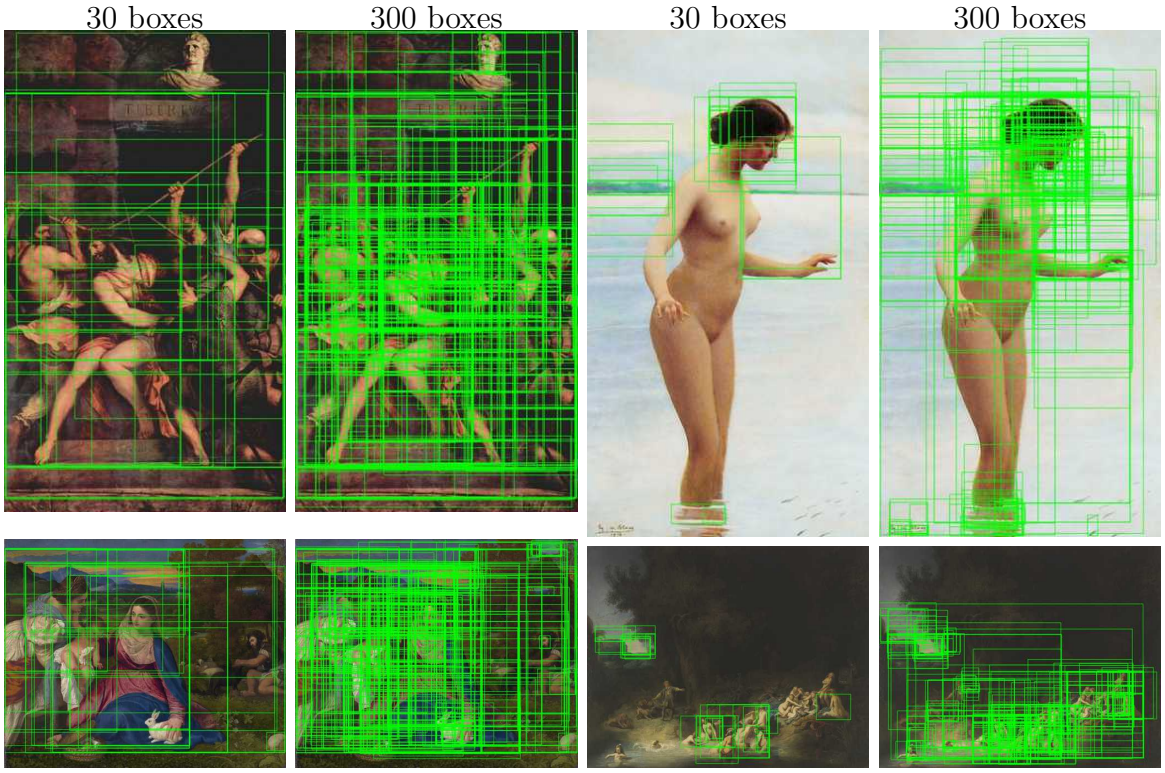


Figure 5.12: Boxes extracted by the EdgeBoxes method: selection of the 30 or 300 "best" boxes on images from IconArt.

If we compared to boxes proposed by the RPN of a pre-trained Faster R-CNN, one can clearly see the pertinence of the boxes (Figure 5.13).

5.3.3.4 Influence of the Parameters of the Model

In this section, we analyze the influence of the different hyperparameters of our MI-max model. We show in Figure 5.14 the performances with respect to each of the three following parameters: the number of restarts, the batch size and the regularization term C. We vary one parameter at a time while keeping the others fixed to the already mentioned values (for instance 11 for the number of restarts, 1000 for the batch size and 1.0 for C). Although the study by Doran et al. [2014] shows that restarts from random points is not always useful for nonconvex models, we find that having about

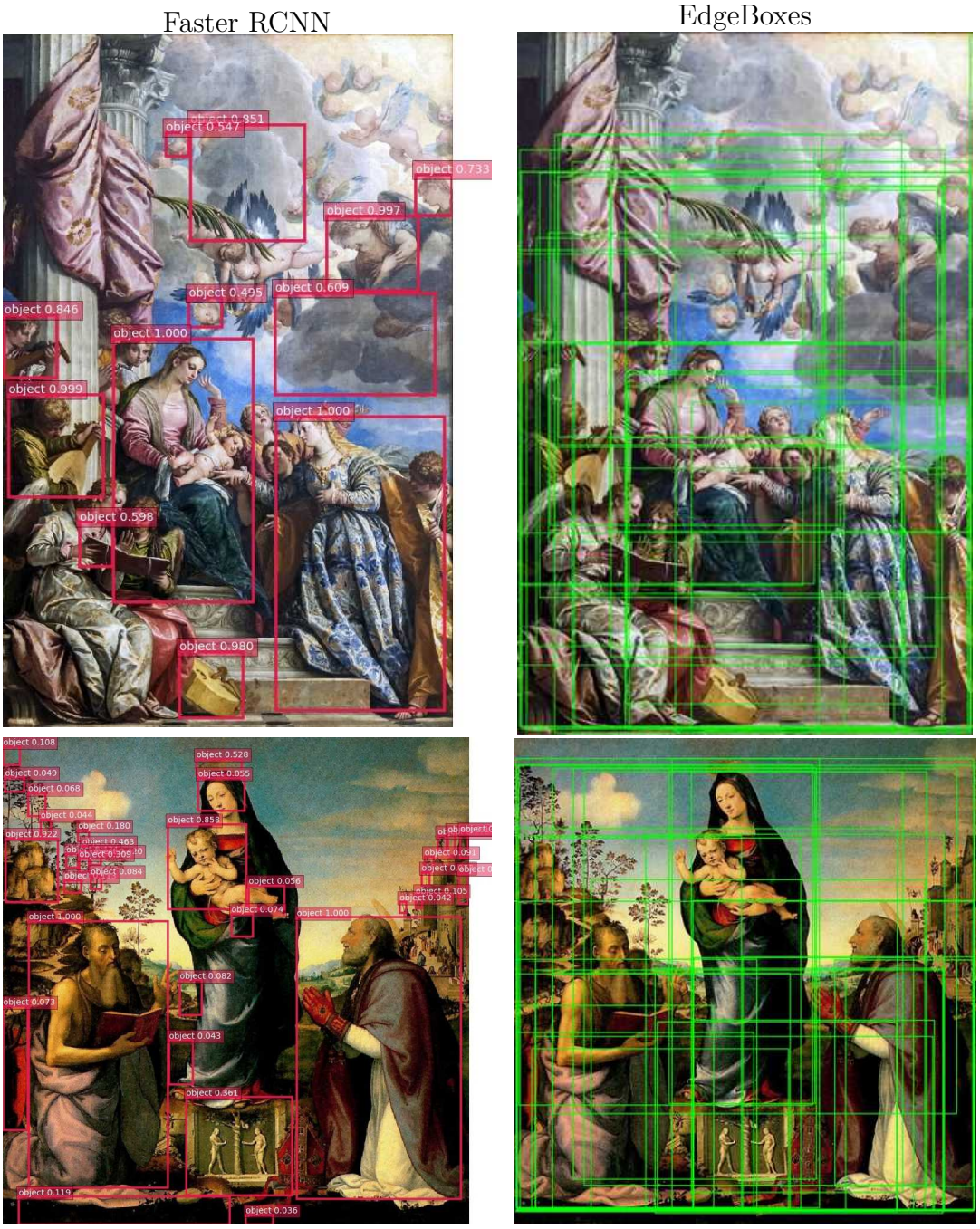


Figure 5.13: Illustration of the boxes extracted by the RPN of Faster R-CNN (red) or by the EdgeBoxes model (green) on images from IconArt.

10 restarts slightly improves the performances and can be taken as a rule of thumb for our models. Notice that the variance of the outcomes is also reduced for such a parameter choice. We also found experimentally that restarts for mi-SVM or MI-SVM reduce the performance in accordance with the experiments in [Doran et al. 2014]. Then, we remark that increasing the batch size provides better results and often yields a reduction of the variance. For the regularization term, we observe relatively constant performances between 1.0 and 2.0. The value 0.5 seems to be the best for 2 of the datasets (PeopleArt and IconArt, but with a bigger variance). These experiments also show the necessity of using a regularization term in the loss function.

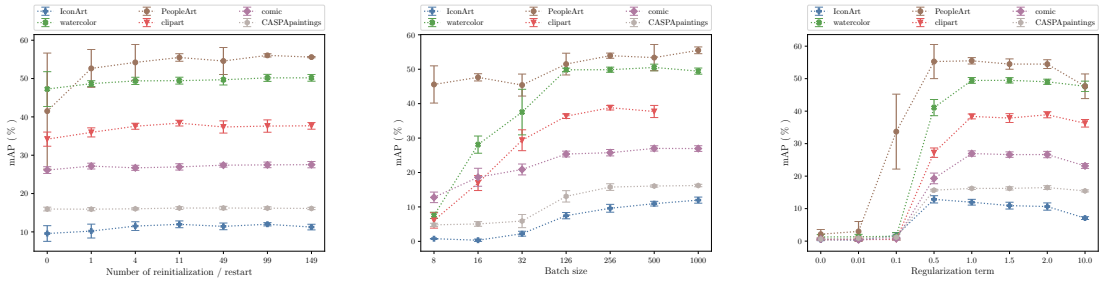


Figure 5.14: Impact of the different hyperparameters on the MI-max model.

Although the best C hyperparameter value seems to be consistent, we also tried to do the minimization of the loss function with different values of the parameter C (equation (5.3)) and to pick up the couple (W,b) that reaches the best values of the loss function on a validation set. This is a naive but cost-free way to a selection of the hyperparameter of the model. The results are shown in Table 5.9, the performance increases for 2 datasets over 6 (including IconArt) but it decreases for the four others. With a score of 16,3, this approach beats the previously best method on IconArt (namely MI_Net). The increase is due to one single class (JCchild with a mean AP of 35,4 this class due to a better C value). This implies that it is still possible to improve the performance score but this approach is too naive. This approach could be more robust if we replace the parameter C selection by a cross-selection with a k-fold procedure and a retraining on the full training plus validation sets.

Dataset	mean AP
PeopleArt	56.5 ± 2.2
Watercolor2k	46.8 ± 2.2
Clipart1k	34.6 ± 1.9
Comic2k	22.6 ± 1.2
CASPA paintings	**16.3** ± 0.7
IconArt	**16.3** ± 1.6

Table 5.9: Mean average precision over the classes of the different datasets (%). Performance of the MI-max with an optimization of the C. Standard deviation is computed on 10 runs of the method.

5.3.3.5 Polyhedral MI-max Limits

In Figure 5.15, one can see the correlation between the vectors W for two different classes of the IconArt dataset and two models MI-max and Polyhedral MI-max. The twelve independently trained vectors are highly correlated in the case of the angel class (Figure 5.15a) but not in the case of the ruins one (Figure 5.15b). Even with a random initialization, the optimization process may lead to the same local minimum, without a surprise. Indeed, we are using exactly the same training set and the same optimization parameters while only the vector initialization changes. When the vectors are trained by taking the maximum over the twelve scalar products, some resulting vectors are still highly correlated for the class angel (Figure 5.15c). It could be interesting to look at the contribution of each individual vector to the final classification decision. The same phenomenon of highly correlated vectors can be observed without the use of the objectness score. It could be possible to impose the fact that the vectors have to be de-correlated with geometric constraints on the set of hyperplanes. Boosting or bagging methods to obtain more diverse hyperplanes should be investigated.

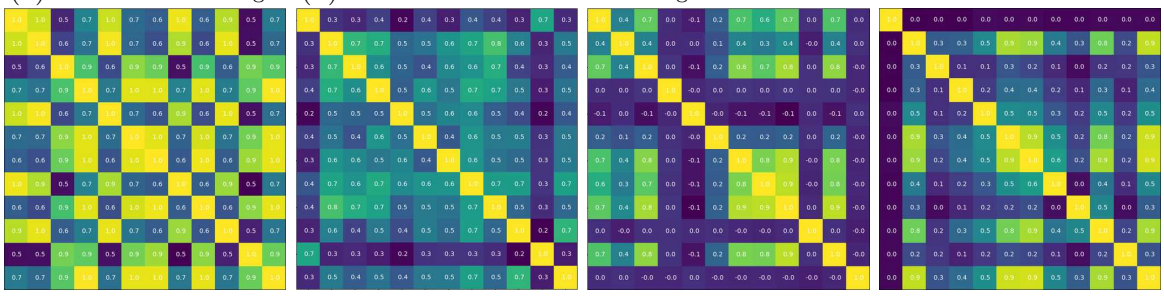


Figure 5.15: Two first figures: correlation matrix for 12 independently trained vectors (for hyperplane) on IconArt with the MI-max model.
The two others: correlation matrix for 12 commonly trained vectors on IconArt with Polyhedral MI-max model.

5.3.3.6 Cross Modalities Knowledge Transfer

Tables 5.10 and 5.11 present cross-domain performance for two of our models Polyhedral MI-max and MI-max. We compute the performances of detection for the classes that are shared between the different datasets. These performances (run once) are compared to the mean performance on the same modality (several runs as before). This experiment illustrates the fact that our method can be transferred to other modalities of images. This is sometimes called the "Cross-Depiction Problem" [Hall et al. 2015]: recognizing visual objects regardless of whether they are painted or depicted in different artistic styles. First, we can see that the Polyhedral MI-max model trained on PeopleArt outperforms the one learned on the target modality for 2 of the 3 datasets (first line). This can be due to the fact the PeopleArt dataset contains many different artistic style. The fact that the class person is well detected can also be due to the Faster R-CNN features that have been trained on a dataset (MS COCO) containing this class for natural images. Nevertheless, we observe that the MI-max trained on

Source set \ Target set	PeopleArt	Watercolor2k	Comic2k	Clipart1k	CASPA paintings
PeopleArt	-	60.0 (59.2)	42.1 (39.5)	54.3 (55.4)	/
Watercolor2k	56.0 (57.3)	-	23.1 (24.1)	11.2 (24.6)	13.8 (18.3)
Comic2k	48.9 (57.3)	42.4 (46.6)	-	7.2 (24.6)	12.5 (18.3)
Clipart1k	52.0 (57.3)	36.7 (46.6)	19.6 (24.1)	-	7.7 (13.6)
CASPA paintings	/	27.5 (39.0)	9.9 (18.1)	4.2 (12.5)	-

Table 5.10: Mean AP (%) at IuO $\geqslant$0.5 for the common classes between the source and target sets with the Polyhedral MI-max model. The results in parentheses, is the mean performance obtained by learning the detection on the same set (modality).

source set \ target set	PeopleArt	Watercolor2k	Comic2k	Clipart1k	CASPA paintings
PeopleArt	-	0.0 (58.2)	0.0 (37.0)	0.0 (55.5)	/
Watercolor2k	47.4 (55.5)	-	25.8 (27.0)	12.2 (33.4)	15.6 (18.3)
Comic2k	50.4 (55.5)	47.3 (49.5)	-	10.0 (33.4)	15.0 (18.3)
Clipart1k	36.2 (55.5)	44.3 (49.5)	25.2 (27.0)	-	10.8 (14.0)
CASPA paintings	/	33.4 (35.4)	12.2 (15.2)	4.7 (22.5)	-

Table 5.11: Mean AP (%) at IuO $\geqslant$0.5 for the common classes between the source and target sets with the MI-max model. The results in parentheses, is the mean performance obtained by learning the detection on the same set (modality).

PeopleArt model fails badly on those three datasets. Finally, we can notice that some datasets such as CASPA paintings and Clipart1k are more challenging than the others. It may be due to the difference in the modality in the second case. This experiment illustrates the fact that our model Polyhedral MI-max generalizes well but also that providing a diverse and numerous training set can help to get a better detector trained in a weakly supervised manner.

Thus, the best solution to have a good detector for a given visual category may be to train a MI-max model on the most stylistically various dataset among the datasets with the required annotation. Then, it is possible to gather MIL detector trained on different datasets for different classes, to build a search engine capable of recognizing multiple visual categories. The quest of generic features is a necessity to build a detector which is efficient at the same time on different modalities.

5.3.3.7 Visual Results

In order to give some intuitive insights on the ability of the proposed method, we show more visual illustrations of the performance of the proposed models (in particular from MI-max and Polyhedral MI-max), both in successful and failure cases.

Successful detections We show successful results on various datasets for the Polyhedral MI-max detection scheme. In Figures 5.16 and 5.17 we show various examples of the visual categories our models are able to detect, respectively on Watercolor2k and CASPA painting datasets. In Figure 5.18, we can see the large stylistic diversity that the model is able to detect for a same class, namely person, on the PeopleArt dataset. In Figure 5.19, one can see some detections on the challenging IconArt dataset.

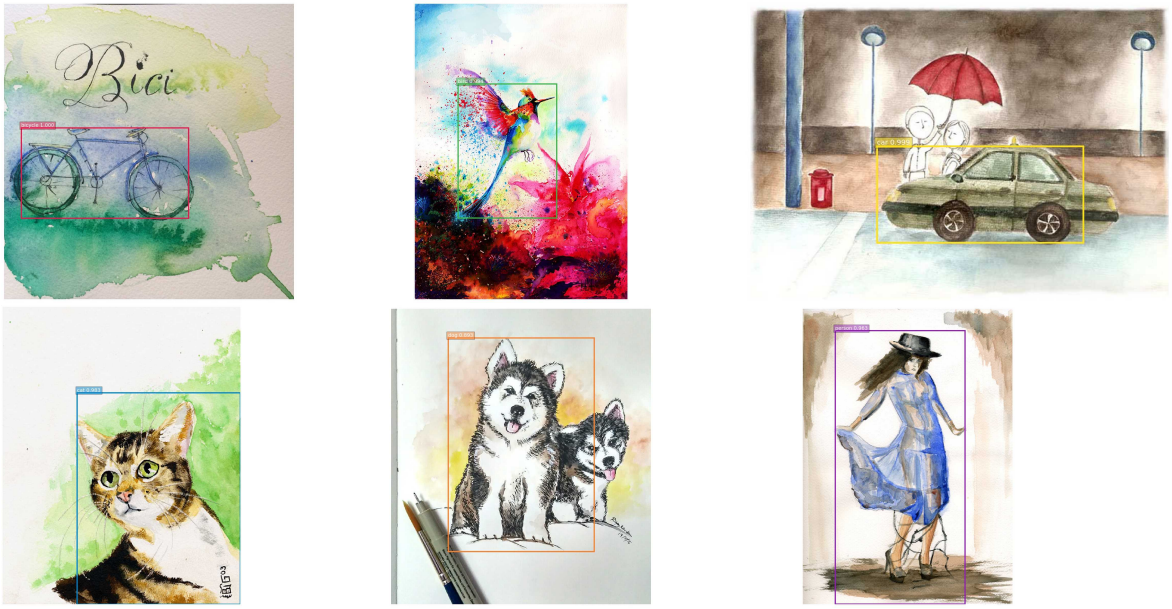


Figure 5.16: One successful example per class using our Polyhedral MI-max detection scheme on Watercolor2k test set. We only show boxes whose scores are over 0.75.

Figure 5.17: One successful example per class using our Polyhedral MI-max detection scheme on CASPA paintings test set. We only show boxes whose scores are over 0.75, except for the elephant image.

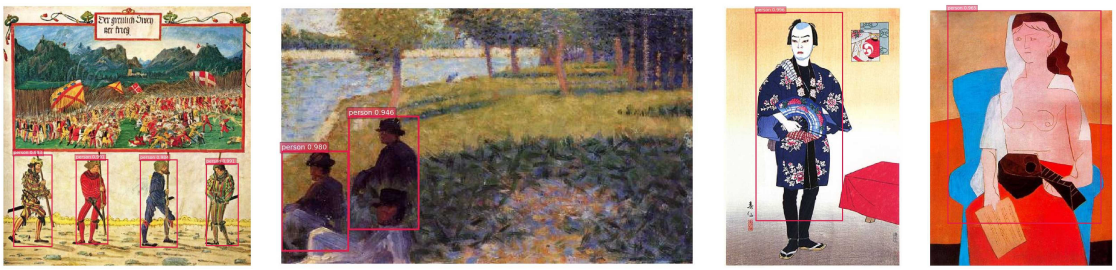


Figure 5.18: Successful examples using our Polyhedral MI-max detection scheme on PeopleArt test set. One can observe the strong stylistic differences between the images. We only show boxes whose scores are over 0.75.

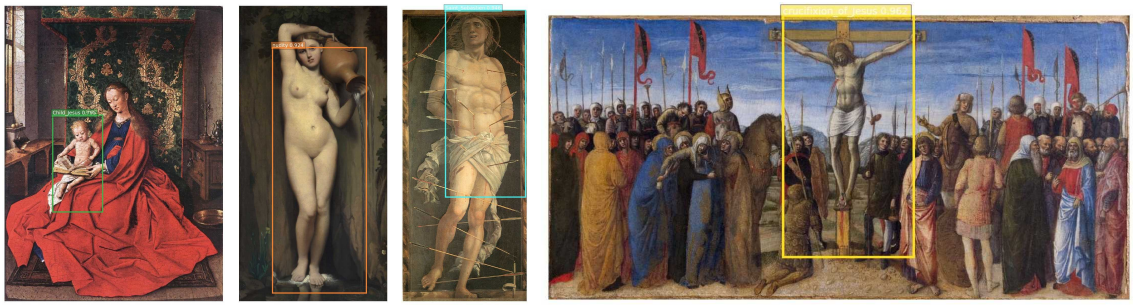


Figure 5.19: Successful examples of detection of iconographic characters using our Polyhedral MI-max detection scheme on IconArt test set. We only show boxes whose scores are over 0.75.

Failures examples We can categorize the failures cases into five main categories that occurred on the different datasets and for the different models, these failure cases may be related to the characteristic of the MIL problem mentioned in Section 2.4.2.1. We also show some images from the training sets to investigate some of the failures. In that case, only the boxes with the highest score for the labels associated with the image are shown.

The discriminative elements are detected instead of the whole object To detect the most discriminative parts of an object seems to be the most common problem in the WSOD setup. It occurs for all our models. For instance, the MI-max w/o score model only detects the legs in Figure 5.20 for the visual category nudity or the arrows instead of Saint Sebastian in Figure 5.21. The Polyhedral MI-max without score model detects the hand in Figure 5.22. This failure already occurs on the training examples and will only be reproduced on the test set. It is clearly due to the use of a bag-level classification loss as proxy for training an instance-level classifier. The use of the objectness score may reduce this phenomenon as mentioned before in Section 5.3.3.1 (see the training images of PeopleArt in Figure 5.9) and illustrated in Figure 5.21 but it will not completely solve the problem.

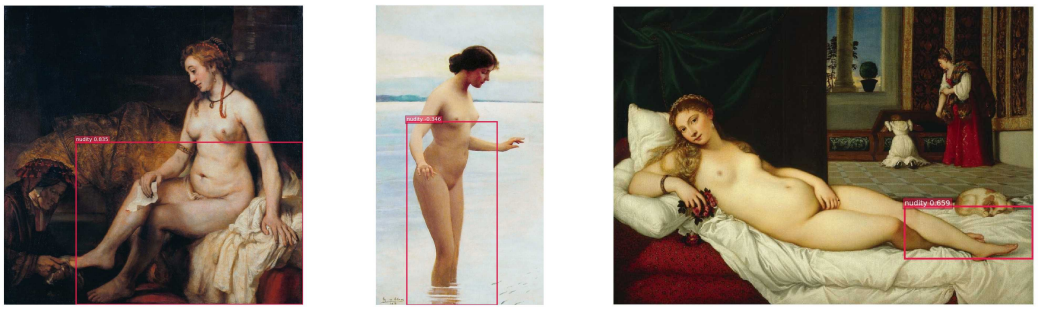


Figure 5.20: Boxes of the best element of the class on train examples with the knowledge of the ground truth class: discriminative part for nudity with MI-max w/o score.

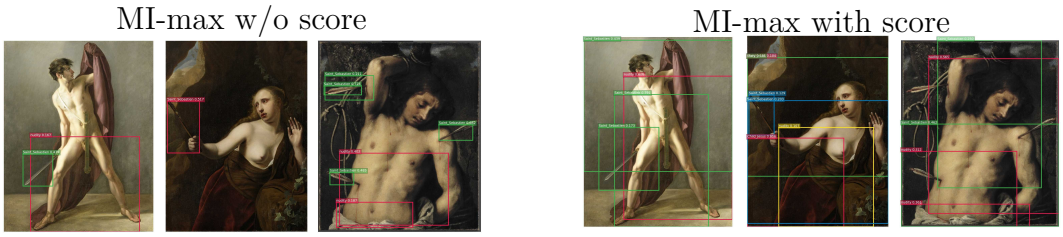


Figure 5.21: An example of wrongly detected object at test time, when using MI-max without or with the objectness score. In the first case, arrows or spike are detected instead of Saint Sebastian.

Detection of a whole group instead of individual instances Another common failure case is the detection of a whole group of instances instead of the individual instances. Such failures cases can be seen in Figure 5.23 for the Polyhedral MI-max without score model. This can lead to a high error rate as the metric demands to

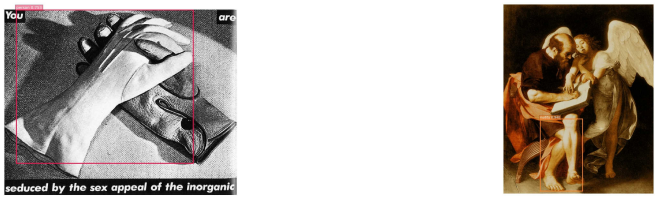


Figure 5.22: Failure examples using our Polyhedral MI-max detection scheme on different datasets. We only show boxes whose scores are over 0.75. The most discriminative boxes correspond to parts of the whole objects. On the first image, the gloves are detected instead of a person. On the second one, the legs are detected as nudity.

have one box per instance. A related failure case can be observed in Figure 5.24, the MI-max w/o score model provides too big boxes for the Mary visual category.

This kind of failure can be due to the use of a loss function defined at an image level. Thus, it may be more effective for the model to use the whole image or group of objects to correctly classify the input image. This can be related to the "Instance Co-occurence" characteristic of the WSOD problem, mentioned in Section 2.4.2.1.

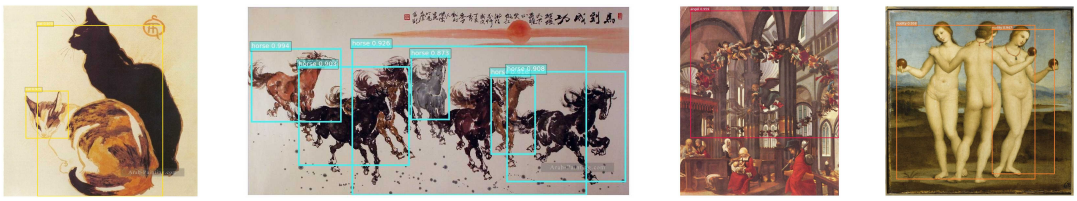


Figure 5.23: Failure examples using our Polyhedral MI-max detection scheme on different datasets. We only show boxes whose scores are over 0.75. Whole groups are detected instead of the instances.

Figure 5.24: Boxes of the best element of the class on train examples with the knowledge of the ground truth class: Boxes too big for Mary with MI-max w/o score.

Misclassification of correct bounding box In Figure 5.25, one can observe bounding boxes that correctly wrap the object of interest but for the wrong class.

Missing mode Sometimes, at the end of the training, for a given class, none of the regions are classified as positive although the image label is positive. However the box with the highest (but negative) score is the box containing the object of interest. One can see it for the nudity visual category in Figure 5.26. This can be a hint that our model is not complex enough to capture several modes in the visual category. The

Figure 5.25: Failure examples using our Polyhedral MI-max detection scheme on different datasets. We only show boxes whose scores are over 0.75. Mis-classified boxes: on the first image the bird is classified as a dog and on the second one the dog is detected as a cat.

model is not able to capture the multiple modes in the positive instance distribution. The same conclusion can be made with the fact that the head-and-shoulders portraits of angels are completely ignored by the hyperplane separator as in the two last images of Figure 5.26, whereas full-length portrait of angel or head surrounded by wings are well detected as in Figure 5.27. According to the example in Figure 5.28, it could be the wings that are detected for the class angel and not the whole character, once again the most discriminative element is detected.

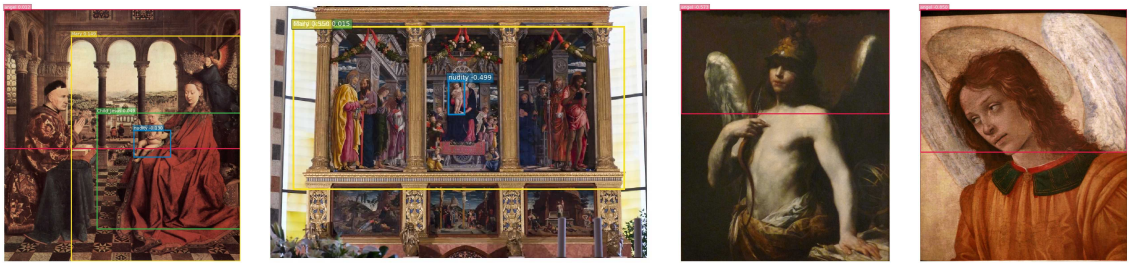


Figure 5.26: Boxes of the best element of the class on train examples with the knowledge of the ground truth class, for the MI-max w/o score model. The best region is correct but associated to a negative score: for the nudity class in the two first images and for the angel class in the two others.

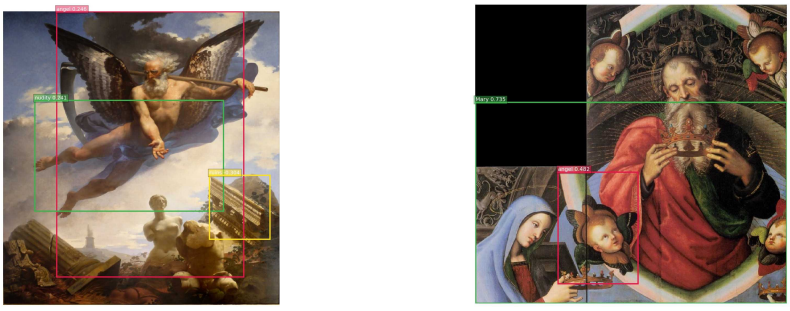


Figure 5.27: Boxes of the best element of the class on train examples with the knowledge of the ground truth class: Angel correct detection with MI-max w/o score.

Confusing images Relatively advanced knowledge in art history is needed to know that the child on the left of Figure 5.29 is Saint John the Baptist whereas the

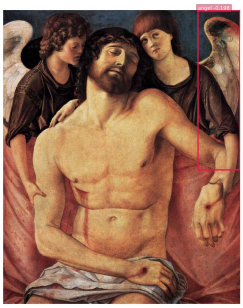
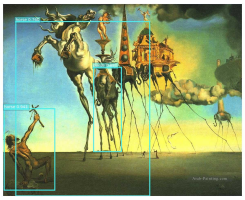


Figure 5.28: Boxes of the best element of the class on train examples with the knowledge of the ground truth class: Angel wings detection with MI-max w/o score.

one on the right is Jesus Child (the element of interest within the image).



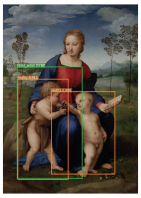


Figure 5.29: Failure examples using our Polyhedral MI-max detection scheme on different datasets. We only show boxes whose scores are over 0.75. Those are confusing images. In the first one, a bear in an human posture is detected as a person. In the middle, the horse, the man and other animals are deformed. The last one is a confusing case between Saint John the Baptist and Jesus children who are visually similar.

In Figure 5.30, the previously shown training images are displayed with the MI-max model predictions for the curious reader. We are facing we face a similar problem for the class Mary (i.e. prediction of box too big), but the boxes selected for the nudity category are bigger.

5.3.4 Concerning Classification

As mentioned before (Section 5.3.3.3), a good classification score is not correlated to a good detection score. The classification scores are available in Annex (Section C.3). Hence, we can observe that we only have 56.7 % with the MI-max model and 60.4 % with the MI-max w/o score for the IconArt dataset (Table C.12) whereas the MI-max model provides better detection performance. The Polyhedral MI-max leads to a classification performance of 61.9% better than our other approaches. These scores are comparable to the "Feature extraction + SVM" based methods for this dataset (Table B.1). The use of the objectness score increases the classification performances for the six datasets in the case of the Polyhedral MI-max model and it does not decrease the performances for the MI-max model on the five other datasets. Moreover, on the six datasets, the MI_Net model trained on the deep features provides the best classification performance. This model has been designed for a bag-level classification task which could explain the high scoring.

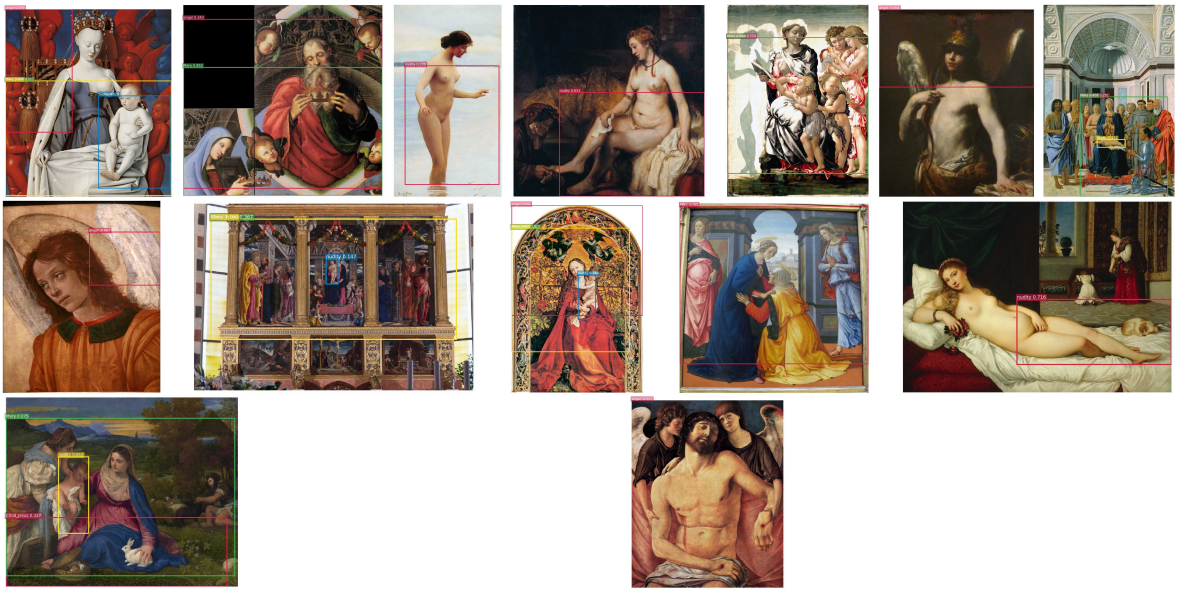


Figure 5.30: Boxes on train set with MI-max model for comparison with the detection obtained with the MI-max w/o score model in Figures 5.20, 5.24 and 5.26 to 5.28.

5.3.5 Classical MIL Benchmarks

As our proposed method is generic, we tried to evaluate our models on three classical MIL benchmarks: **Birds** [Briggs et al. 2012], **SIVAL** [Rahmani et al. 2005] and **Newsgroups** [Craven et al. 2008] for an instance classification task. We follow the benchmark protocol detailed in [Carbonneau et al. 2016a], i.e. 10 repetitions of a 10-fold cross validation of each class (one-vs-the rest). The dataset that is the most relevant to our case is the Newsgroups one because the positive bags contain an average of 3.7% of positive instances. In the case of the 300 regions proposed by a Faster R-CNN, we can consider that 1% of the regions contain the object of interest. Nevertheless the feature vectors from Newsgroups are sparse and discretized because they represent the frequency of some words in text. The CNN features are not discretized and maybe not sparse. The performance of our models can be found in Annex, in Tables C.13 to C.15 with three different metrics: Unweighted Average Recall (UAR), F1-score and Area Under receiver operating characteristic Curve (AUC). We compared 3 different variants of our model: MI-max, MI-max-HL and Polyhedral MI-max.

Our models are not able to beat the state-of-the-art ones even if sometimes on one of the metrics they do better. Moreover our methods have an important standard deviation on their mean performance, which means they are less stable than the classical ones. On the Newsgroups dataset, the Polyhedral MI-max model is competitive compared to the other methods (according to the AUC and the UAR metrics).

But the F1-score is more pertinent compared to the AUC and the UAR metrics because the data are heavily unbalanced. However, our models are way faster and much simpler to design. Surprisingly, our models achieve better results than Single Instance SVM, mi-SVM and MI-SVM methods on the WSOD task on several datasets but not those classical MIL datasets. We have to mention that the state-of-the-art methods are not linear and that their optimal hyperparameters have been found by cross validation.

5.4 Conclusion

In this chapter, we confirm that transfer learning of pretrained CNN can provide a good model to automatically analyze artistic image databases. This was previously shown for classification and fully supervised detection tasks, and was here investigated in the case of weakly supervised object detection. We proposed a simple and quick model to solve the multiple instance problem we are facing. Our solution is based on features and boxes proposals from a pretrained CNN detector coupled with a multiple instance perceptron. We intensively compared different variants of this model. This framework is particularly suited to develop tools helping art historians, because it avoids tedious annotations and opens the way to learning on large datasets. We also show, in this context,experiments dealing with iconographic elements that are specific to art history and cannot be learned on natural images.

6

Conclusion and Perspectives

6.1 Summary of Contributions

In this thesis, we have proposed a methodology for a better understanding of CNNs internal representations and have explored some of their potentialities for image analysis and image synthesis. We tackled various ways in which these internal representations can be used to solve computer vision problems, as summarized here after.

One of the key challenges of computer vision is how to transfer the features learned on natural images to other tasks. We illustrated the versatility of deep features from pretrained CNNs in three different problems: texture synthesis, art classification and weakly supervised object detection. In all three cases, we relied on transferring models pretrained on large scale databases of natural images.

We have shown that the internal representations of CNNs can be used to synthesize high resolution images even through they were trained on low resolution images. For this problem, we proposed a multi-resolution framework to address high resolution texture synthesis with CNNs (Chapter 3). Our multi-resolution strategy can be effectively coupled with a Fourier frequency constraint or the auto-correlation of the CNN features as statistics for the synthesis of structured textures.

We confirmed that features extracted from CNNs can be effective for off-the-shelf art classification (Chapter 4). Nevertheless, fine-tuning a pretrained CNN is more effective than using the off-the-shelf approach or training from scratch. It provides an excellent initialization point to the learning process even for a different modality. We also illustrated that a double fine-tuning involving an intermediate artistic dataset can improve the classification on smaller datasets.

Finally, we proposed a simple but efficient multiple instance model on deep features for the WSOD task in artworks (Chapter 5). Our method does not require any fine-tuning or cross-domain learning and is therefore efficient and possibly applicable to arbitrary datasets and classes. It is an effective way to learn to locate objects of interest with only image-level supervision.

Before proceeding to the perspectives our work opens, let us focus particularly on two contributions.

Analysis of Fine-tuned CNN We studied the effect of transfer learning of a CNN from ImageNet to art dataset. In this respect, we used feature visualization by maximization to see the internal representations changes for one particular architecture. We also provides a quantitative analysis of the changes introduced by the learning process thanks to metrics in both feature and parameter spaces, as well as metrics computed on the set of maximal activation images. This allowed us to point that mid-level layers learned recognizable patterns from the new modality whereas higher layers tend to concentrate classes.

Texture synthesis evaluation Another challenging problem in texture synthesis is how to evaluate the output of the different methods. We proposed two unsupervised ways to rank them. The first metric is based on displacement maps and allows discarding methods prone to excessive copy-paste. The second one relies on Kullback-Leibler divergence between the marginal distributions of wavelet coefficients. It provides a ranking that is coherent with visual assessment. To confirm our subjective observations, we ran a relatively large user study. We asked almost one hundred participants to vote between pairs of synthesized textures with respect to the reference image. We asked them to focus on local scale aspects and on the global structure of the image. This perceptual study has shown the benefit of the multi-resolution scheme for high resolution textures and the interest of combining it with correlation constraints for regular textures.

6.2 Future Work

Several limitations or open questions remain in our work, that deserve attention and are partly listed here as our prospects.

6.2.1 Texture Synthesis

Using CNN trained with a multi-resolution strategy Texture synthesis with a CNN as feature extractor illustrates the fact than the VGG filters are versatile and provide good features at different resolutions, although the network has been trained on 224×224 images. An interesting experiment in this respect would be to synthesize textures using the ensemble of networks trained at different scales from van Noord et al. [2017]. Our multi-resolution strategy could be used straightforwardly with these CNNs. Another solution could be to use networks that are able to focus on higher resolution crops within the images as in [Bianco et al. 2019]. These networks may have a more complete and multi-scale representation of images.

Reducing the number of parameters used A strong limitation of the neural methods investigated in this work is the unreasonably large number of parameters of the models, about 200k parameters for the method based on Gram matrices. In this respect, the next question is not "What set of statistical constraints is sufficient?", but "What is the minimal set of statistical constraints needed to produce realistic synthesis?" Using moments or p-norms with the multi-resolution strategy may be the

first step in this direction. Besides, reducing the number of parameters is a step towards a more understandable inner-working of the algorithms used.

Automatic estimation of the weighting of the spectrum constraint The wavelets based metric that we considered in Section 3.3.3.2 could be used as a metric to automatically determine the best weighting when defining a loss function accounting for several constraints, e.g. the β parameter between the Gram based term and the spectrum constraint (equation (3.4)) for a given reference image. A complete study is needed to estimate the pertinence of this claim.

Neural style transfer A natural extension would be to investigate the use of such multi-resolution strategies for neural style transfer of high-resolution images, following Gatys et al. [2015a]. Another experiment is to replace the Gram Matrices by the autocorrelation for the style transfer process to better understand what is captured by this kind of statistics.

Using another architecture than VGG19 Although VGG is still the best model for texture synthesis with CNNs, the robust deep models seem to be a promising alternative. First, it could allow to use deeper networks with bigger receptive field to generate high resolution images without using "semantic" layers. Second, investigating why VGG architecture still provides the best results for texture synthesis and InceptionV1 for feature visualization by optimization seems to be a path to gain a better understanding of CNNs internal representations.

6.2.2 Transfer Learning and Analysis of Convolutional Neural Networks for Art Classification

Working with higher resolution images Using higher resolution images may help to provide better classifiers [Tan et al. 2019]. First, it can be simply done by using fully convolutionnal networks such as the WILDCAT model [Durand et al. 2017]. It is also possible to use an ensemble of CNNs trained at different scale as in [van Noord et al. 2017]. Nevertheless, it may be interesting to have a model or an ensemble of models that is both scale-invariant and scale-variant. This is necessary to be able to detect both a central object in the foreground and the same object smaller in the background. Moreover, the depiction of the object may be different according to the scale. These higher resolution or multi-resolution models could be analyzed with a similar methodology to the one developed in this thesis.

Extension to other art datasets The study of Section 4.4 provides some insights on the way networks are modified by fine-tuning in the case of artistic databases. Several of the findings in this work necessitate further confirmation, possible on larger databases [Wilber et al. 2017; Strezoski et al. 2018] annotated with concepts from art history such as the IconClass labels [Strezoski et al. 2018; Posthumus 2020] or with visual pattern specific to art as faces of pre-modern Japanese characters [Tian et al. 2020].

Further Feature Visualizations Another perspective would be to go further in the use of feature visualization, as it is done in [Olah et al. 2018; Olah et al. 2020a]. For instance, it could be more informative to look at the patches that fire a given channel rather than the whole input image. In [Olah et al. 2020a], the authors claim for universality within the hidden representations among different deep convolutional architectures and it is of interest to check if the same is true for artistic datasets with the different architecture we studied. For instance, we found a "drapery" channel in InceptionV1 architecture similarly to [Offert 2018] who considered the InceptionV3 architecture. Additional optimized images may be seen in Annex (Section B.2.4). It would be necessary to develop automatic tools for analyzing a large number of optimized images and see the redundancy between differently-initialized networks. Using texture similarity models may be an effective solution [Ding et al. 2020] in this aspect.

Finally, we could also look at the principal directions (thanks to the covariance matrices as in PCA) within the feature maps blocks instead of looking at each channel individually. These principal directions could be used to extract groups of channels kernels correlated with a particular output class. This could enable to have a more generic understanding of the hidden representations learned by a layer.

Developing new tool for art historians CNNs can be used to classify images of artistic dataset. These classification predictions allow analyzing large scale corpora. Offert [2018] propose to include the optimized images obtained with the classification CNN in addition to the classification predictions to boost reflection. Offert et al. [2020] named these optimized images: *technical metapictures*. These images can be seen as "representations of representation", they are informative on the machine perception and on the regularity within the training set. They can be considered as archetypes of the training corpora and could be useful for digital art historians.

6.2.3 Weakly Supervised Object Detection in Artworks

WSOD supervised by a classification network A perspective is to supervise the training of a weakly supervised detector with a fully-trained classifier in order to remove some obvious misclassified box candidates as it can be done in classical WSOD methods [Wan et al. 2018]. Feature maps contain a coarse localization information as mentioned previously in Section 2.4.2.3. This could provide better detection performances.

Improving the multiple instance polyhedral perceptron by forcing hyperplanes to be distinct We showed our Polyhedral MI-max model is still prone to learn several times the same separator hyperplane. This problem may be solved in two possible ways. One way is to add some geometric constraint in the polyhedral case to force the hyperplanes to be as distinct as possible and to get better boundaries. The other one is to add some data mining of hard examples as in classical object detection algorithm [Felzenszwalb et al. 2010; Girshick et al. 2014].

Moreover, it might be beneficial to take more than one instance per bag to learn better detectors and uncover multi-modal visual category.

Using deep features learned on art dataset A more extensive investigation of the different possible feature extractors and boxes proposal algorithms could show the flexibility of our model. As it has been shown before (Section 4.3.2.2), natural images pretraining may be less effective for transfer learning than a pretraining on natural plus artistic datasets, but there is no artistic dataset equivalent to MS COCO [Lin et al. 2014] for pretraining a detector as Faster R-CNN. For that reason, several works promote domain adaptation mixed with weakly supervised learning [Inoue et al. 2018] for visual categories present both in photographic and artistic images. Another solution should be to use an unsupervised fine-tuning of the classification part of the Faster R-CNN without breaking the region proposal part.

Another exciting direction is to investigate the potential of weakly supervised learning on large databases with only image-level annotations [Wilber et al. 2017; Strezoski et al. 2018] or even to let the user provide dozens of positive training examples to learn a new visual category on the fly as done by Crowley et al. [2018]. This framework could be used to develop a versatile search engine for diverse modalities of images, avoiding the time-consuming annotation task. This kind of pre-iconographical objects detector could support the study of the evolution of iconography, enabling advanced queries over artistic images, for instance "Find all images where the angel appears to the right of Mary", "Find all the images where Mary appears with Jesus Child on her lap". Mixing those visual recognition models with metadata associated to the images is a necessity to ask questions related to space and time, such as "Find the earliest crucifixion in the Italian collection". Nevertheless, the field of data science has been especially good at identifying visual patterns and correlations in large datasets but taking a step further would require social and contextual knowledge [Kitchin 2014] in addition to sources and tools criticism [Bishop 2018; Offert et al. 2020].

Bibliography

Ackermann, Sandro, Kevin Schawinski, Ce Zhang, Anna K. Weigel, and M. Dennis Turp (2018). "Using Transfer Learning to Detect Galaxy Mergers". In: *Monthly Notices of the Royal Astronomical Society* 479.1, pp. 415–425 (cit. on pp. 18, 35).

Aguerrebere, Cecilia, Yann Gousseau, and Guillaume Tartavel (2013). "Exemplar-Based Texture Synthesis: The Efros-Leung Algorithm". In: *Image Processing On Line* 3, pp. 223–241 (cit. on pp. 75, 89, 95, 96, 103).

Alain, Guillaume and Yoshua Bengio (2017). "Understanding Intermediate Layers Using Linear Classifier Probes". In: *ICLR Workshop* (cit. on p. 154).

Andrews, Stuart, Ioannis Tsochantaridis, and Thomas Hofmann (2003). "Support Vector Machines for Multiple-Instance Learning". In: *Advances in Neural Information Processing Systems*, pp. 577–584 (cit. on pp. 57, 62, 153, 154, 165, 167, 168, 303–313).

Artrendex (2018). *ArtPi Is the Art World Discovery Engine*. Artrendex. URL: `https://www.artpi.co/` (visited on 11/20/2017) (cit. on pp. 19, 54).

Arun, Aditya, C. V. Jawahar, and M. Pawan Kumar (2019). "Dissimilarity Coefficient Based Weakly Supervised Object Detection". In: *CVPR* (cit. on p. 70).

Astorino, A. and M. Gaudioso (2002). "Polyhedral Separability Through Successive LP". In: *Journal of Optimization Theory and Applications* 112.2, pp. 265–293 (cit. on p. 156).

Aubry, Mathieu, Bryan C. Russell, and Josef Sivic (2014). "Painting-to-3D Model Alignment via Discriminative Visual Elements". In: *ACM Transactions on Graphics* 33.2, pp. 1–14 (cit. on p. 44).

Badea, Mihai, Corneliu Florea, Laura Florea, and Constantin Vertan (2018). "Can We Teach Computers to Understand Art? Domain Adaptation for Enhancing Deep Networks Capacity to De-Abstract Art". In: *Image and Vision Computing* 77, pp. 21–32 (cit. on pp. 46, 50).

Bagirov, Adil M. (2005). "Max–Min Separability". In: *Optimization Methods and Software* 20.2-3, pp. 277–296 (cit. on p. 156).

Bagirov, Adil M. and Julien Ugon (2005). "Supervised Data Classification via Max-Min Separability". In: *Continuous Optimization: Current Trends and Modern Applications*. Applied Optimization. Boston, MA: Springer US, pp. 175–207 (cit. on p. 156).

Basha, S. H. Shabbeer, Sravan Kumar Vinakota, Shiv Ram Dubey, Viswanath Pulabaigari, and Snehasis Mukherjee (2020a). *AutoFCL: Automatically Tuning Fully Connected Layers for Handling Small Dataset*. Version 3. arXiv: `2001.11951 [cs, eess]` (cit. on p. 33).

Basha, S. H. Shabbeer, Sravan Kumar Vinakota, Viswanath Pulabaigari, Snehasis Mukherjee, and Shiv Ram Dubey (2020b). "AutoTune: Automatically Tuning Convolutional Neural Networks for Improved Transfer Learning". In: *Neural Networks* 133, pp. 112–122 (cit. on p. 33).

Bender, K. (2015). "Distant Viewing in Art History. A Case Study of Artistic Productivity". In: *International Journal for Digital Art History* 1 (1) (cit. on pp. 19, 54).

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert, and Jason Weston (2009). "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. Montréal, Canada: ACM Press, pp. 41–48 (cit. on p. 64).

Berger, Guillaume and Roland Memisevic (2017). "Incorporating Long-Range Consistency in CNN-Based Texture Generation". In: *ICLR* (cit. on pp. 82, 83, 93, 94).

Bergmann, Urs, Nikolay Jetchev, and Roland Vollgraf (2017). "Learning Texture Manifolds with the Periodic Spatial GAN". In: *Proceedings of the 34th International Conference on Machine Learning*. ICML. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 469–477 (cit. on p. 84).

Bernstein, Jeremy, Arash Vahdat, Yisong Yue, and Ming-Yu Liu (2020). "On the Distance between Two Neural Networks and the Stability of Learning". In: *Advances in Neural Information Processing Systems 33*. NeurIPS (cit. on p. 40).

Bianco, Simone, Davide Mazzini, Paolo Napoletano, and Raimondo Schettini (2019). "Multitask Painting Categorization by Deep Multibranch Neural Network". In: *Expert Systems with Applications* 135, pp. 90–101 (cit. on pp. 46, 50, 188).

Bianco, Simone, Davide Mazzini, and Raimondo Schettini (2017). "Deep Multibranch Neural Network for Painting Categorization". In: *Image Analysis and Processing - ICIAP 2017*. Vol. 10484. Cham: Springer International Publishing, pp. 414–423 (cit. on pp. 46, 49).

Bilen, Hakan (2018). "CVPR18: Tutorial: Part 1: Weakly Supervised Learning for Computer Vision" (CVPR2018) (cit. on pp. 55, 67).

Bilen, Hakan, Marco Pedersoli, and Tinne Tuytelaars (2014). "Weakly Supervised Object Detection with Posterior Regularization". In: Proceedings BMVC 2014, pp. 1–12 (cit. on pp. 63–65).

Bilen, Hakan and Andrea Vedaldi (2016). "Weakly Supervised Deep Detection Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR. IEEE, pp. 2846–2854 (cit. on pp. 60, 64, 65, 67, 68, 158, 162, 165, 167, 303–305).

Bishop, Claire (2018). "Against Digital Art History". In: *International Journal for Digital Art History* 3 (3) (cit. on pp. 54, 191).

Bongini, Pietro, Federico Becattini, Andrew D. Bagdanov, and Alberto Del Bimbo (2020). "Visual Question Answering for Cultural Heritage". In: *IOP Conf. Series: Materials Science and Engineering*. HERITECH. Vol. 949 (cit. on p. 48).

Borowski, Judy, Roland S. Zimmermann, Judith Schepers, Robert Geirhos, Thomas S. A. Wallis, Matthias Bethge, and Wieland Brendel (2020). *Exemplary Natural Images Explain CNN Activations Better than Feature Visualizations*. Version 1. arXiv: 2010.12606 [cs.CV] (cit. on pp. 38, 136).

Bozinovski, Stevo (2020). "Reminder of the First Paper on Transfer Learning in Neural Networks, 1976". In: *Informatica* 44.3, pp. 291–302 (cit. on p. 31).

Brachmann, Anselm, Erhardt Barth, and Christoph Redies (2017). "Using CNN Features to Better Understand What Makes Visual Artworks Special". In: *Frontiers in Psychology* 8 (cit. on p. 52).

Bradley, Ralph Allan and Milton E Terry (1952). "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons." In: *Biometrika* 39.3/4, pp. 324–345 (cit. on pp. 106, 268).

Braun, Julia, Seyed Ali Amirshahi, Joachim Denzler, and Christoph Redies (2013). "Statistical Image Properties of Print Advertisements, Visual Artworks and Images of Architecture". In: *Frontiers in Psychology* 4 (cit. on p. 42).

Breiman, Leo (1993). "Hinging Hyperplanes for Regression, Classification, and Function Approximation". In: *IEEE Transactions on Information Theory* 39.3, pp. 999–1013 (cit. on p. 156).

— (1996). "Bagging Predictors". In: *Machine Learning* 24.2, pp. 123–140 (cit. on p. 155).

Brendel, Wieland and Matthias Bethge (2019). "Approximating CNNs with Bag-of-Local-Features Models Works Surprisingly Well on ImageNet". In: *International Conference on Learning Representations* (cit. on pp. 39, 81).

Bressan, M., C. Cifarelli, and F. Perronnin (2008). "An Analysis of the Relationship between Painters Based on Their Work". In: *2008 15th IEEE International Conference on Image Processing*. 2008 15th IEEE International Conference on Image Processing, pp. 113–116 (cit. on p. 43).

Briggs, Forrest, Xiaoli Z. Fern, and Raviv Raich (2012). "Rank-Loss Support Instance Machines for MIML Instance Annotation". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '12*. The 18th ACM SIGKDD International Conference. Beijing, China: ACM Press, p. 534 (cit. on pp. 185, 311).

Cai, Xiuxia, Bin Song, and Zhiqian Fang (2019). "Exemplar Based Regular Texture Synthesis Using LSTM". In: *Pattern Recognition Letters* (cit. on p. 85).

Cammarata, Nick, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah (2020). "Curve Detectors". In: *Distill* 5.6, e00024.003 (cit. on p. 38).

Carbonneau, Marc-André, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon (2016a). "Multiple Instance Learning: A Survey of Problem Characteristics and Applications". In: *Pattern Recognition* 77, pp. 329–353 (cit. on pp. 59–61, 185, 311).

Carbonneau, Marc-André, Eric Granger, Alexandre J. Raymond, and Ghyslain Gagnon (2016b). "Robust Multiple-Instance Learning Ensembles Using Random Subspace Instance Selection". In: *Pattern Recognition* 58, pp. 83–99 (cit. on pp. 58, 59, 62, 311–313).

Carter, Shan, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah (2019). "Activation Atlas". In: *Distill* 4.3, e15 (cit. on p. 38).

Castellano, Giovanna, Eufemia Lella, and Gennaro Vessio (2020a). *Visual Link Retrieval and Knowledge Discovery in Painting Datasets*. Version 1. arXiv: 2003.08476 [cs] (cit. on pp. 49, 53).

Castellano, Giovanna and Gennaro Vessio (2020b). *Deep Convolutional Embedding for Digitized Painting Clustering*. Version 1. arXiv: 2003.08597 [cs] (cit. on p. 49).

Cetinic, Eva and Sonja Grgic (2013). "Automated Painter Recognition Based on Image Feature Extraction". In: *Proceedings ELMAR-2013*. Proceedings ELMAR-2013, pp. 19–22 (cit. on p. 43).

Cetinic, Eva, Tomislav Lipic, and Sonja Grgic (2018). "Fine-Tuning Convolutional Neural Networks for Fine Art Classification". In: *Expert Systems with Applications* (cit. on pp. 33, 46).

— (2019a). "A Deep Learning Perspective on Beauty, Sentiment, and Remembrance of Art". In: *IEEE Access* 7, pp. 73694–73710 (cit. on pp. 52, 53).

— (2019b). "Learning the Principles of Art History with Convolutional Neural Networks". In: *Pattern Recognition Letters* (cit. on pp. 47, 53).

Chatfield, Ken, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman (2014). "Return of the Devil in the Details: Delving Deep into Convolutional Nets". In: *BMVC* (cit. on p. 33).

Chen, Liyi and Jufeng Yang (2019). "Recognizing the Style of Visual Arts via Adaptive Cross-Layer Correlation". In: *Proceedings of the 27th ACM International Conference on Multimedia*. Nice,France, p. 9 (cit. on p. 51).

Chen, Xinlei and Abhinav Gupta (2015). "Webly Supervised Learning of Convolutional Networks". In: IEEE, pp. 1431–1439 (cit. on p. 55).

— (2017). *An Implementation of Faster RCNN with Study for Region Sampling*. arXiv: 1702.02138 [cs] (cit. on p. 162).

Cheplygina, Veronika (2019). "Cats or CAT Scans: Transfer Learning from Natural or Medical Image Source Datasets?" In: *Current Opinion in Biomedical Engineering* 9, pp. 21–27 (cit. on pp. 33, 34).

Chu, Brian, Vashisht Madhavan, Oscar Beijbom, Judy Hoffman, and Trevor Darrell (2016a). "Best Practices for Fine-Tuning Visual Classifiers to New Domains". In: *Computer Vision–ECCV 2016 Workshops*. Springer, pp. 435–442 (cit. on p. 33).

Chu, Lingyang, Xia Hu, Juhua Hu, Lanjun Wang, and Jian Pei (2018a). "Exact and Consistent Interpretation for Piecewise Linear Neural Networks: A Closed Form Solution". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. London, United Kingdom: ACM, pp. 1244–1253 (cit. on p. 155).

Chu, Wei-Ta and Yi-Ling Wu (2016b). "Deep Correlation Features for Image Style Classification". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. Amsterdam, The Netherlands: Association for Computing Machinery, pp. 402–406 (cit. on p. 51).

— (2018b). "Image Style Classification Based on Learnt Deep Correlation Features". In: *IEEE Transactions on Multimedia* 20.9, pp. 2491–2502 (cit. on p. 51).

Cicalese, Pietro Antonio, Aryan Mobiny, Pengyu Yuan, Jan Becker, Chandra Mohan, and Hien Van Nguyen (2020). "StyPath: Style-Transfer Data Augmentation for Robust Histology Image Classification". In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 351–361 (cit. on pp. 50, 81).

Cinbis, Ramazan Gokberk, Jakob Verbeek, and Cordelia Schmid (2014). "Multi-Fold MIL Training for Weakly Supervised Object Localization". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, pp. 2409–2416 (cit. on pp. 64, 65).

— (2016). "Weakly Supervised Object Localization with Multi-Fold Multiple Instance Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.1, pp. 189–203 (cit. on p. 65).

Clarke, Alasdair, Fraser Halley, Andrew Newell, Lewis Griffin, and Mike Chantler (2011). "Perceptual Similarity: A Texture Challenge". In: *Procedings of the British Machine Vision Conference 2011*. British Machine Vision Conference 2011. Dundee: British Machine Vision Association, pp. 1–10 (cit. on pp. 85, 89, 106).

Collomosse, John, Tu Bui, Michael J Wilber, Chen Fang, and Hailin Jin (2017). "Sketching With Style: Visual Search With Sketches and Aesthetic Context". In: *ICCV*, p. 9 (cit. on pp. 48, 81).

Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR, pp. 3213–3223 (cit. on p. 54).

Cortes, Corinna, Mehryar Mohri, and Afshin Rostamizadeh (2012). "Algorithms for Learning Kernels Based on Centered Alignment". In: *Journal of Machine Learning Research* 13, pp. 795–828 (cit. on pp. 41, 144).

Cortes, Corinna and Vladimir Vapnik (1995). "Support-Vector Networks". In: *Machine Learning* 20.3, pp. 273–297 (cit. on p. 57).

Courtiol, Pierre, Eric W. Tramel, Marc Sanselme, and Gilles Wainrib (2018). *Classification and Disease Localization in Histopathology Using Only Global Labels: A Weakly-Supervised Approach*. Version 1. arXiv: `1802.02212 [cs, stat]` (cit. on p. 66).

Cover, T. and P. Hart (1967). "Nearest Neighbor Pattern Classification". In: *IEEE Transactions on Information Theory* 13.1, pp. 21–27 (cit. on p. 156).

Craven, Burr Settles Mark and Soumya Ray (2008). "Multiple-Instance Active Learning". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1289–1296 (cit. on pp. 185, 311, 312).

Cross, George R. and Anil K. Jain (1983). "Markov Random Field Texture Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5.1, pp. 25–39 (cit. on pp. 20, 72, 74).

Crowley, Elliot J., Ernesto Cot, and Andrew Zisserman (2018). *Oxford Painting Search*. URL: http://zeus.robots.ox.ac.uk/artsearch/ (visited on 03/02/2018) (cit. on pp. 19, 54, 127, 191).

Crowley, Elliot J. and Andrew Zisserman (2013). "Of Gods and Goats: Weakly Supervised Learning of Figurative Art". In: *In Proceedings British Machine Vision Conference 2013*. Vol. 39. BMVA Press, pp. 1–11 (cit. on p. 43).

— (2014). "In Search of Art". In: *Workshop at the European Conference on Computer Vision*. Springer, pp. 54–70 (cit. on pp. 18, 20, 35, 45, 47, 125–127, 129, 130, 132, 134, 164).

— (2016). "The Art of Detection". In: *European Conference on Computer Vision*. Springer, pp. 721–737 (cit. on pp. 45, 127–130, 163, 164, 167, 302–310).

Csurka, Gabriela (2017). "A Comprehensive Survey on Domain Adaptation for Visual Applications". In: *Domain Adaptation in Computer Vision Applications*. Advances in Computer Vision and Pattern Recognition. Cham: Springer International Publishing, pp. 1–35 (cit. on p. 35).

Csurka, Gabriella, Christopher R Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray (2004). "Visual Categorization with Bags of Keypoints". In: *Workshop on Statistical Learning in Computer Vision*. ECCV, pp. 1–22 (cit. on p. 18).

Cubuk, Ekin Dogus, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le (2019). "AutoAugment: Learning Augmentation Policies from Data". In: *CVPR* (cit. on p. 31).

Dai, Jifeng, Yi Li, Kaiming He, and Jian Sun (2016). "R-FCN: Object Detection via Region-Based Fully Convolutional Networks". In: *Advances in Neural Information Processing Systems*, pp. 379–387 (cit. on p. 68).

Dalal, N. and B. Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1, pp. 886–893 (cit. on pp. 18, 29, 44).

Darzi, Anna, Itai Lang, Ashutosh Taklikar, Hadar Averbuch-Elor, and Shai Avidan (2020). *Co-Occurrence Based Texture Synthesis*. Version 1. arXiv: 2005.08186 [cs] (cit. on p. 84).

De Bortoli, Valentin, Agnès Desolneux, Bruno Galerne, and Arthur Leclaire (2019). "Macrocanonical Models for Texture Synthesis". In: *International Conference on Scale Space and Variational Methods in Computer Vision*. Springer, pp. 13–24 (cit. on pp. 72, 82, 83, 103, 116, 120).

Del, Bimbo and P. Pala (1997). "Visual Image Retrieval by Elastic Matching of User Sketches". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.2, pp. 121–132 (cit. on p. 44).

Del Chiaro, R., A Bagdanov, and A. Del Bimbo (2019a). "NoisyArt: A Dataset for Webly-Supervised Artwork Recognition:" in: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, pp. 467–475 (cit. on p. 55).

Del Chiaro, Riccardo, Andrew D. Bagdanov, and Alberto Del Bimbo (2019b). "Webly-Supervised Zero-Shot Learning for Artwork Instance Recognition". In: *Pattern Recognition Letters* 128, pp. 420–426 (cit. on p. 48).

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255 (cit. on p. 19).

Deselaers, Thomas, Bogdan Alexe, and Vittorio Ferrari (2012). "Weakly Supervised Localization and Learning with Generic Knowledge". In: *International Journal of Computer Vision* 100.3, pp. 275–293 (cit. on p. 70).

Diba, Ali, Vivek Sharma, Ali Pazandeh, Hamed Pirsiavash, and Luc Van Gool (2017). "Weakly Supervised Cascaded Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR. IEEE, pp. 5131–5139 (cit. on pp. 60, 66, 68).

Dietterich, Thomas G., Richard H. Lathrop, and Tomás Lozano-Pérez (1997). "Solving the Multiple Instance Problem with Axis-Parallel Rectangles". In: *Artificial Intelligence* 89.1, pp. 31–71 (cit. on pp. 56, 62, 156, 157).

Di Lenardo, Isabella, Benoît Laurent Auguste Seguin, and Frédéric Kaplan (2016). "Visual Patterns Discovery in Large Databases of Paintings". In: Digital Humanities 2016 (cit. on pp. 48, 54).

Ding, Keyan, Kede Ma, Shiqi Wang, and Eero P. Simoncelli (2020). *Image Quality Assessment: Unifying Structure and Texture Similarity*. Version 2. arXiv: 2004.07728 [cs] (cit. on p. 190).

Do, M. N. and M. Vetterli (2002). "Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback-Leibler Distance". In: *IEEE Transactions on Image Processing* 11.2, pp. 146–158 (cit. on pp. 89, 103).

Donahue, Jeff, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell (2014). "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *International Conference on Machine Learning*. ICML, pp. 647–655 (cit. on pp. 18, 32, 65, 131, 173).

Dong, Xinghui and Mike J. Chantler (2013). "The Importance of Long-Range Interactions to Texture Similarity". In: *Computer Analysis of Images and Patterns*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 425–432 (cit. on pp. 85, 106).

Dong, Xinghui, Junyu Dong, and Mike Chantler (2020). "Perceptual Texture Similarity Estimation: An Evaluation of Computational Features". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1 (cit. on pp. 85, 94, 106).

Dong, Xuanyi, Deyu Meng, Fan Ma, and Yi Yang (2017). "A Dual-Network Progressive Approach to Weakly Supervised Object Detection". In: *Proceedings of the 25th ACM International Conference on Multimedia*. MM '17. New York, NY, USA: ACM, pp. 279–287 (cit. on p. 68).

Doran, Gary and Soumya Ray (2014). "A Theoretical and Empirical Analysis of Support Vector Machine Methods for Multiple-Instance Classification". In: *Machine Learning* 97.1-2, pp. 79–102 (cit. on pp. 58, 174, 176).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2020). *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. Version 1. arXiv: 2010.11929 [cs] (cit. on p. 35).

Durand, Thibaut, Taylor Mordan, Nicolas Thome, and Matthieu Cord (2017). "WILDCAT: Weakly Supervised Learning of Deep ConvNets for Image Classification, Pointwise Localization and Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*. Honolulu, HI, United States: IEEE (cit. on pp. 62, 66, 189).

Durand, Thibaut, Nicolas Thome, and Matthieu Cord (2015). "MANTRA: Minimum Maximum Latent Structural SVM for Image Classification and Ranking". In: *IEEE International Conference on Computer Vision (ICCV15)*. Santiago, Chile (cit. on pp. 66, 68).

— (2016). "WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks". In: *29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. Las Vegas, NV, United States (cit. on pp. 61, 66).
— (2018). "SyMIL: MinMax Latent SVM for Weakly Labeled Data". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.12, pp. 6099–6112 (cit. on p. 58).
— (2019). "Exploiting Negative Evidence for Deep Latent Structured Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.2, pp. 337–351 (cit. on pp. 58, 66).
Efros, Alexei A. and William T. Freeman (2001). "Image Quilting for Texture Synthesis and Transfer". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, pp. 341–346 (cit. on pp. 75, 80, 89, 95, 97–100, 104, 228–262).
Efros, Alexei A. and Thomas K. Leung (1999). "Texture Synthesis by Non-Parametric Sampling". In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference On*. Vol. 2. IEEE, pp. 1033–1038 (cit. on pp. 20, 72, 73, 75, 89, 95, 97–100, 104, 228–262).
Eilhauer, Alex, Alice Pritikin, Dylan Weed, and Steven J. Gortler (2000). *Combining Textures and Pictures with Specialized Texture Synthesis*. URL: https://www.cs.harvard.edu/~sjg/apr/ (visited on 12/08/2020) (cit. on p. 80).
Elgammal, Ahmed, Marian Mazzone, Bingchen Liu, Diana Kim, and Mohamed Elhoseiny (2018). "The Shape of Art History in the Eyes of the Machine". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. (Cit. on pp. 46, 49, 52, 53).
Engstrom, Logan, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry (2019). *Adversarial Robustness as a Prior for Learned Representations*. Version 2. arXiv: 1906.00945 [cs, stat] (cit. on p. 37).
Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio (2010). "Why Does Unsupervised Pre-Training Help Deep Learning?" In: *Journal of Machine Learning Research* 11, pp. 625–660 (cit. on pp. 28, 32).
Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pascal Vincent, and P O Box (2009). "Visualizing Higher-Layer Features of a Deep Network". In: *University of Montreal* 1341.3, p. 14 (cit. on pp. 36–38).
Europeana (2018). *Collections Europeana*. Collections Europeana. URL: https://www.europeana.eu/portal/?locale=fr (visited on 07/07/2018) (cit. on p. 126).
Everingham, Mark, Luc Van Gool, Christopher K. I. Williams, and Andrew Zisserman (2010). "The PASCAL Visual Object Classes Challenge". In: *International Journal of Computer Vision* 88, pp. 303–338 (cit. on pp. 19, 126, 128, 163).
Felzenszwalb, Pedro, Ross B. Girshick, David McAllester, and Deva Ramanan (2010). "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9, pp. 1627–1645 (cit. on pp. 18, 44, 57, 58, 65, 153, 154, 190).
Felzenszwalb, Pedro, David McAllester, and Deva Ramanan (2008). "A Discriminatively Trained, Multiscale, Deformable Part Model". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR. Anchorage, AK, USA: IEEE, pp. 1–8 (cit. on p. 58).
Fiorucci, Marco, Marina Khoroshiltseva, Massimiliano Pontil, Arianna Traviglia, Alessio Del Bue, and Stuart James (2020). "Machine Learning for Cultural Heritage: A Survey". In: *Pattern Recognition Letters* (cit. on p. 54).
Florea, Corneliu, Mihai Badea, Laura Florea, and Constantin Vertan (2017). "Domain Transfer for Delving into Deep Networks Capacity to De-Abstract Art". In: *Scandinavian Con-*

ference on Image Analysis. Vol. 10269. Lecture Notes in Computer Science. Springer, Cham, pp. 337–349 (cit. on pp. 46, 50).

Freund, Yoav and Robert E. Schapire (1995). "A Desicion-Theoretic Generalization of on-Line Learning and an Application to Boosting". In: *Computational Learning Theory*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 23–37 (cit. on p. 155).

Frühstück, Anna, Ibraheem Alhashim, and Peter Wonka (2019). "TileGAN: Synthesis of Large-Scale Non-Homogeneous Textures". In: *ACM Transactions on Graphics (TOG)* 38.4 (cit. on p. 84).

Fukushima, Kunihiko (1980). "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". In: *Biological Cybernetics* 36.4, pp. 193–202 (cit. on pp. 17, 27, 28).

Galerne, Bruno, Yann Gousseau, and Jean-Michel Morel (2011). "Micro-Texture Synthesis by Phase Randomization". In: *Image Processing On Line* 1, pp. 213–237 (cit. on pp. 75, 89, 91, 92, 117).

Galerne, Bruno, Arthur Leclaire, and Julien Rabin (2018). "A Texture Synthesis Model Based on Semi-Discrete Optimal Transport in Patch Space". In: *SIAM Journal on Imaging Sciences* 11.4, pp. 2456–2493 (cit. on p. 85).

Garcia, Noa, Benjamin Renoust, and Yuta Nakashima (2019). "Context-Aware Embeddings for Automatic Art Analysis". In: *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pp. 25–33 (cit. on pp. 46, 48, 51, 54).

Garcia, Noa, Chentao Ye, Zihua Liu, Qingtao Hu, Mayu Otani, Chenhui Chu, Yuta Nakashima, and Teruko Mitamura (2020). "A Dataset and Baselines for Visual Question Answering on Art". In: *VISART Workshop at ECCV 2020* (cit. on p. 48).

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge (2015a). *A Neural Algorithm of Artistic Style*. Version 2. arXiv: `1508.06576 [cs, q-bio]` (cit. on pp. 50, 51, 80, 89, 106, 189).

— (2015b). "Texture Synthesis Using Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. NIPS, pp. 262–270 (cit. on pp. 18, 20, 37, 39, 41, 72, 73, 76, 78, 79, 81, 82, 84, 85, 88–91, 95, 97–100, 104, 106, 108–110, 112, 113, 115–121, 228–262, 273, 275–285).

Gatys, Leon A., Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman (2016). *Controlling Perceptual Factors in Neural Style Transfer*. arXiv: `1611.07865 [cs]` (cit. on pp. 81, 82, 91, 95).

Ge, Ce, Jingyu Wang, Qi Qi, Haifeng Sun, and Jianxin Liao (2018). "Fewer Is More: Image Segmentation Based Weakly Supervised Object Detection with Partial Aggregation." In: *BMVC*, p. 136 (cit. on p. 67).

Gehler, Peter V. and Olivier Chapelle (2007). "Deterministic Annealing for Multiple-Instance Learning". In: *Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics, pp. 123–130 (cit. on p. 58).

Geirhos, Robert, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann (2020). *Shortcut Learning in Deep Neural Networks*. arXiv: `2004.07780 [cs, q-bio]` (cit. on pp. 35, 39).

Geirhos, Robert, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel (2019). "ImageNet-Trained CNNs Are Biased towards Texture; Increasing Shape Bias Improves Accuracy and Robustness". In: *ICLR* (cit. on pp. 38, 39, 50, 81).

Gidas, B. (1989). "A Renormalization Group Approach to Image Processing Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.2, pp. 164–180 (cit. on p. 89).

Ginosar, Shiry, Daniel Haas, Timothy Brown, and Jitendra Malik (2014). "Detecting People in Cubist Art". In: *Workshop at the European Conference on Computer Vision*. Springer, pp. 101–116 (cit. on p. 47).

Girshick, Ross B. (2015). "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision*. ICCV, pp. 1440–1448 (cit. on pp. 35, 63, 65, 67, 68).

Girshick, Ross B., Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587 (cit. on pp. 32, 33, 35, 63, 131, 190).

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *PMLR*. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (cit. on p. 29).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press (cit. on pp. 17, 28).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., pp. 2672–2680 (cit. on p. 84).

Gordea, Sergiu, David Haskiya, and Ash Marriott (2017). *Europeana DSI 2– Access to Digital Resources of European Heritage Advanced Image Discovery Report*. Deliverable. Europeana Foundation (cit. on p. 19).

Gorgels, Peter (2013). "Rijksstudio: Make Your Own Masterpiece!" In: *Museums and the Web 2013*. Silver Spring (cit. on p. 18).

Graham, Daniel J and David J Field (2008). "Variations in Intensity Statistics for Representational and Abstract Art, and for Art from the Eastern and Western Hemispheres". In: *Perception* 37.9, pp. 1341–1352 (cit. on p. 42).

— (2007). "Statistical Regularities of Art Images and Natural Scenes: Spectra, Sparseness and Nonlinearities". In: *Spatial Vision* 21.1-2, pp. 149–164 (cit. on pp. 42, 43, 52).

Graham, Daniel J. and Christoph Redies (2010). "Statistical Regularities in Art: Relations with Visual Coding and Perception". In: *Vision Research* 50.16, pp. 1503–1509 (cit. on p. 43).

Grzeszick, Rene, Sebastian Sudholt, and Gernot A. Fink (2018). *Weakly Supervised Object Detection with Pointwise Mutual Information*. Version 1. arXiv: 1801.08747 [cs] (cit. on p. 66).

Gu, Qianqian and Ross King (2019). "Deep Learning Does Not Generalize Well to Recognizing Cats and Dogs in Chinese Paintings". In: *Discovery Science*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 166–175 (cit. on p. 47).

Güler, Rıza Alp, Natalia Neverova, and Iasonas Kokkinos (2018). "DensePose: Dense Human Pose Estimation In The Wild". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7297–7306 (cit. on p. 35).

Gupta, Akash, Niluthpol C Mithun, Conrad Rudolph, and Amit K Roy-Chowdhury (2018). "Deep Learning Based Identity Verification in Renaissance Portraits". In: *2018 IEEE International Conference on Multimedia and Expo (ICME)*. San Diego, CA, USA, pp. 1–6 (cit. on p. 50).

gwern (2018). *Eat Your VGGtables, or, Why Does Neural Style Transfer Work Best With Old VGG CNNs' Features?* reddit. URL: https://www.reddit.com/r/MachineLearning/comments/7rrrk3/d_eat_your_vggtables_or_why_does_neural_style/ (visited on 11/12/2020) (cit. on p. 79).

Hall, Peter, Hongping Cai, Qi Wu, and Tadeo Corradi (2015). "Cross-Depiction Problem: Recognition and Synthesis of Photographs and Artwork". In: *Computational Visual Media* 1.2, pp. 91–103 (cit. on pp. 44, 45, 128, 177).

Haralick, R.M. (1979). "Statistical and Structural Approaches to Texture". In: *Proceedings of the IEEE* 67.5, pp. 786–804 (cit. on p. 93).

He, Kaiming, Ross B. Girshick, and Piotr Dollar (2019). "Rethinking ImageNet Pre-Training". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, pp. 4917–4926 (cit. on pp. 34, 132).

He, Kaiming, Gkioxari Gkioxari, Piotr Dollár, and Ross B. Girshick (2017). "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988 (cit. on pp. 34, 35).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR, pp. 770–778 (cit. on pp. 29, 30, 128, 129, 131, 164, 302, 318).

He, Kun, Yan Wang, and John Hopcroft (2016). "A Powerful Generative Model Using Random Weights for the Deep Image Representation". In: *Advances in Neural Information Processing Systems*. NIPS, pp. 631–639 (cit. on pp. 29, 79).

He, Zhenwei and Lei Zhang (2020). "Domain Adaptive Object Detection via Asymmetric Tri-Way Faster-RCNN". In: *ECCV* (cit. on p. 35).

Heeger, David J. and James R. Bergen (1995). "Pyramid-Based Texture Analysis/Synthesis". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. ACM, pp. 229–238 (cit. on pp. 72–74, 76, 95).

Heitz, Eric, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour (2020). *Pitfalls of the Gram Loss for Neural Texture Synthesis in Light of Deep Feature Histograms*. Version 2. arXiv: `2006.07229 [cs]` (cit. on pp. 72, 82).

Heitz, Fabrice, Henri Maitre, and Charles de Couessin (1987). "Application of Autoregressive Models to Fine Arts Painting Analysis". In: *Signal Processing* 13.1, pp. 1–14 (cit. on p. 42).

Hentschel, Christian, Timur Pratama Wiradarma, and Harald Sack (2016). "Fine Tuning CNNS with Scarce Training Data — Adapting Imagenet to Art Epoch Classification". In: *2016 IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, USA: IEEE, pp. 3693–3697 (cit. on p. 46).

Hertzmann, Aaron, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin (2001). "Image Analogies". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, pp. 327–340 (cit. on p. 80).

Hinton, Geoffrey E. (2007). "Learning Multiple Layers of Representation". In: *Trends in Cognitive Sciences* 11.10, pp. 428–434 (cit. on p. 32).

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh (2006). "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18.7, pp. 1527–1554 (cit. on p. 37).

Hoffman, Judy, Sergio Guadarrama, Eric Tzeng, Ronghang Hu, Jeff Donahue, Ross Girshick, Trevor Darrell, and Kate Saenko (2014). "LSDA: Large Scale Detection Through Adaptation". In: *Advances in neural information processing systems,* 27, pp. 3536–3544 (cit. on p. 70).

Houdard, Antoine, Arthur Leclaire, Nicolas Papadakis, and Julien Rabin (2020). *Wasserstein Generative Models for Patch-Based Texture Synthesis*. Version 1. arXiv: `2007.03408 [cs, stat]` (cit. on p. 85).

Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam (2017). *MobileNets: Efficient Convolu-*

tional Neural Networks for Mobile Vision Applications. Version 1. arXiv: 1704.04861 [cs] (cit. on p. 29).

Hu, Jie, Li Shen, and Gang Sun (2018). "Squeeze-and-Excitation Networks". In: CVPR (cit. on p. 29).

Huang, Xiaolin, Siamak Mehrkanoon, and Johan A.K. Suykens (2013). "Support Vector Machines with Piecewise Linear Feature Mapping". In: *Neurocomputing* 117, pp. 118–127 (cit. on p. 156).

Huang, Xingsheng, Sheng-hua Zhong, and Zhijiao Xiao (2017a). "Fine-Art Painting Classification via Two-Channel Deep Residual Network". In: *Advances in Multimedia Information Processing – PCM 2017*. Pacific Rim Conference on Multimedia. Lecture Notes in Computer Science. Springer, Cham, pp. 79–88 (cit. on pp. 46, 51).

Huang, Xun and Serge Belongie (2017b). "Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1501–1510 (cit. on p. 50).

Hubel, David H and Torsten N Wiesel (1962). "Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex". In: *The Journal of Physiology* 160, pp. 106–154 (cit. on p. 27).

— (1968). "Receptive Fields and Functional Architecture of Monkey Striate Cortex". In: *The Journal of Physiology* 195.1, pp. 215–243 (cit. on p. 27).

Huckle, Nikolai, Noa Garcia, and Yuta Nakashima (2020). *Demographic Influences on Contemporary Art with Unsupervised Style Embeddings*. Version 1. arXiv: 2009.14545 [cs] (cit. on pp. 49, 53).

Huh, Minyoung, Pulkit Agrawal, and Alexei A. Efros (2016). *What Makes ImageNet Good for Transfer Learning?* Version 2. arXiv: 1608.08614 [cs] (cit. on p. 33).

Hurtut, Thomas, Yann Gousseau, Farida Cheriet, and Francis Schmitt (2011). "Artistic Line-Drawings Retrieval Based on the Pictorial Content". In: *Journal on Computing and Cultural Heritage* 4.1, pp. 1–23 (cit. on pp. 19, 44).

Hurtut, Thomas, Yann Gousseau, and Francis Schmitt (2008). "Adaptive Image Retrieval Based on the Spatial Organization of Colors". In: *Computer Vision and Image Understanding* 112.2, pp. 101–113 (cit. on pp. 19, 44).

Ilse, Maximilian, Jakub M. Tomczak, and Max Welling (2018). "Attention-Based Deep Multiple Instance Learning". In: *ICML*. International Conference on Machine Learning. Stockholm (cit. on p. 64).

Inoue, Naoto, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa (2018). "Cross-Domain Weakly-Supervised Object Detection through Progressive Domain Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018)*. IEEE (cit. on pp. 20, 35, 50, 162, 165–167, 191, 303–305, 307).

Ioffe, Sergey (2017). "Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. Pp. 1945–1953 (cit. on p. 315).

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. ICML. Vol. 37 (cit. on p. 29).

Jackson, Philip T, Amir Atapour-Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara (2019). "Style Augmentation: Data Augmentation via Style Randomization". In: *CVPR Workshops*, pp. 83–92 (cit. on p. 50).

Jafarpour, Sina, Gungor Polatkan, Eugene Brevdo, Shannon Hughes, Andrei Brasoveanu, and Ingrid Daubechies (2009). "Stylistic Analysis of Paintings Using Wavelets and Machine Learning". In: *Signal Processing Conference, 2009 17th European*. IEEE, pp. 1220–1224 (cit. on p. 43).

Jenicek, Tomas and Ondřej Chum (2019). "Linking Art through Human Poses". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1338–1345 (cit. on p. 53).

Jetchev, Nikolay, Urs Bergmann, and Roland Vollgraf (2016). *Texture Synthesis with Spatial Generative Adversarial Networks*. Version 4. arXiv: 1611.08207 [cs, stat] (cit. on p. 84).

Jie, Zequn, Yunchao Wei, Xiaojie Jin, Jiashi Feng, and Wei Liu (2017). "Deep Self-Taught Learning for Weakly Supervised Object Localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 4294–4302 (cit. on p. 64).

Jing, Yongcheng, Yezhou Yang, Zunlei Feng, Jingwen Ye, and Mingli Song (2019). "Neural Style Transfer: A Review". In: *IEEE transactions on visualization and computer graphics* 26.11 (cit. on p. 80).

Jo, Jason and Yoshua Bengio (2017). *Measuring the Tendency of CNNs to Learn Surface Statistical Regularities*. Version 1. arXiv: 1711.11561 [cs, stat] (cit. on pp. 38, 39, 118).

Johnson, C., Ella Hendriks, Igor Berezhnoy, Eugene Brevdo, Shannon Hughes, Ingrid Daubechies, Jia Li, Eric Postma, and James Wang (2008). "Image Processing for Artist Identification". In: *IEEE Signal Processing Magazine* 25.4, pp. 37–48 (cit. on p. 43).

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *European Conference on Computer Vision*. ECCV. Springer, pp. 694–711 (cit. on pp. 50, 81, 82, 93, 118).

Joulin, Armand and Francis Bach (2012). "A Convex Relaxation for Weakly Supervised Classifiers". In: *ICML*, p. 8 (cit. on pp. 58, 63, 65).

Joyeux-Prunel, Béatrice and Olivier Marcel (2015). "Exhibition Catalogues in the Globalization of Art. A Source for Social and Spatial Art History". In: *Artl@s Bulletin* 4.2 (cit. on p. 19).

Julesz, Béla (1962). "Visual Pattern Discrimination". In: *IRE Transactions on Information Theory* 8.2, pp. 84–92 (cit. on pp. 18, 73).

Julesz, Béla, Edgar N Gilbert, Larry A Shepp, and Harry L Frisch (1973). "Inability of Humans to Discriminate between Visual Textures That Agree in Second-Order Statistics—Revisited". In: *Perception*. SAGE Publications Sage UK 2.4, pp. 391–405 (cit. on pp. 73, 75).

Julesz, Béla, Edgar N Gilbert, and Jonathan D Victor (1978). "Visual Discrimination of Textures with Identical Third-Order Statistics". In: *Biological Cybernetics*. Springer 31, pp. 137–140 (cit. on p. 73).

Kaizer, Herb (1955). "A Quantification of Textures on Aerialphotographs". In: *Boston University Research Laboratories, Boston University, Tech. Note* (cit. on p. 93).

Kantorov, Vadim, Maxime Oquab, Minsu Cho, and Ivan Laptev (2016). "ContextLocNet: Context-Aware Deep Network Models for Weakly Supervised Localization". In: *European Conference on Computer Vision*. ECCV. Springer, Cham, pp. 350–365 (cit. on pp. 60, 67, 68).

Karatzoglidi, Maria, Georgios Felekis, and Eleni Charou (2020). "Neural Style Transfer for Remote Sensing". In: *2nd Greek Remote Sensing Workshop RSSAC2020* (cit. on p. 51).

Karayev, Sergey, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller (2014). "Recognizing Image Style". In: *Proceedings British Machine Vision Conference*. BMVA Press (cit. on p. 45).

Khan, Fahad Shahbaz, Shida Beigpour, Joost Van de Weijer, and Michael Felsberg (2014). "Painting-91: A Large Scale Database for Computational Painting Categorization". In: *Machine vision and applications* 25.6, pp. 1385–1397 (cit. on p. 47).

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. Version 9. arXiv: 1412.6980 [cs] (cit. on p. 28).

Kitchin, Rob (2014). "Big Data, New Epistemologies and Paradigm Shifts". In: *Big Data & Society* (cit. on pp. 19, 191).

Kornblith, Simon, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton (2019). "Similarity of Neural Network Representations Revisited". In: *International Conference on Machine Learning*. ICML. Vol. 97. Long Beach, California, USA, pp. 3519–3529 (cit. on pp. 36, 41, 137, 144).

Kornblith, Simon, Jonathon Shlens, and Quoc V. Le (2018). "Do Better ImageNet Models Transfer Better?" In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2661–2671 (cit. on pp. 33, 128, 162).

Kotovenko, Dmytro, Artsiom Sanakoyeu, Sabine Lang, and Bjorn Ommer (2019). "Content and Style Disentanglement for Artistic Style Transfer". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South): IEEE, pp. 4421–4430 (cit. on p. 81).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., pp. 1097–1105 (cit. on pp. 17, 25, 29, 30, 32).

Kumar, M. P., Benjamin Packer, and Daphne Koller (2010). "Self-Paced Learning for Latent Variable Models". In: *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., pp. 1189–1197 (cit. on p. 65).

Kuznetsova, Alina, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari (2020). "The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale". In: *International Journal of Computer Vision* (cit. on pp. 19, 34, 125).

Kwatra, Vivek, Irfan Essa, Aaron Bobick, and Nipun Kwatra (2005). "Texture Optimization for Example-Based Synthesis". In: *ACM Transactions on Graphics* 24.3, pp. 795–802 (cit. on pp. 75, 91, 95).

Lang, Sabine and Björn Ommer (2018). "Reflecting on How Artworks Are Processed and Analyzed by Computer Vision". In: *Computer Vision – ECCV 2018 Workshops*. Vol. 11130. Cham: Springer International Publishing, pp. 647–652 (cit. on pp. 54, 81).

Lang, Sabine, Nikolai Ufer, and Björn Ommer (2019). "Finding Visual Patterns in Artworks: An Interactive Search Engine to Detect Objects in Artistic Images". In: *DH* (cit. on p. 45).

Lecoutre, Adrian, Benjamin Negrevergne, and Florian Yger (2017). "Recognizing Art Style Automatically in Painting with Deep Learning". In: *Asian Conference on Machine Learning*. ACML. JMLR: Workshop and Conference Proceedings, pp. 327–342 (cit. on pp. 20, 33, 46, 47, 53, 125, 126, 132, 142, 146, 148, 149, 291–293, 297).

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural computation*. MIT Press 1.4, pp. 541–551 (cit. on pp. 17, 28).

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on p. 28).

Lee, Seungkwan, Suha Kwak, and Minsu Cho (2019). "Universal Bounding Box Regression and Its Applications". In: *Computer Vision – ACCV 2018*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 373–387 (cit. on p. 70).

Lefebvre, Sylvain and Hugues Hoppe (2005). "Parallel Controllable Texture Synthesis". In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. New York, NY, USA: ACM, pp. 777–786 (cit. on pp. 73, 75).

Li, Da, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales (2017a). "Deeper, Broader and Artier Domain Generalization". In: *ICCV*. International Conference on Computer Vision (cit. on pp. 35, 48).

Li, Dong, Jia-Bin Huang, Yali Li, Shengjin Wang, and Ming-Hsuan Yang (2016). "Weakly Supervised Object Localization with Progressive Domain Adaptation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR. IEEE, pp. 3512–3520 (cit. on pp. 63, 70).

Li, Yanghao, Naiyan Wang, Jiaying Liu, and Xiaodi Hou (2017b). "Demystifying Neural Style Transfer". In: *IJCAI* (cit. on p. 83).

Li, Yixuan, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft (2015). "Convergent Learning: Do Different Neural Networks Learn the Same Representations?" In: *The 1st International Workshop "Feature Extraction: Modern Questions and Challenges"*. JMLR: Workshop and Conference Proceeding. Vol. 44, pp. 196–212 (cit. on p. 40).

Lin, Tsung-Yi, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár (2017). "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988 (cit. on p. 35).

Lin, Tsung-Yi, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár (2014). "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision*. ECCV. Springer, pp. 740–755 (cit. on pp. 19, 29, 128, 162, 191).

Liu, Gang (2017). "Spatial Statistics of Local Descriptors for Texture Modeling and Change Detection". Paris: Télécom ParisTech (cit. on pp. 84, 91, 92).

Liu, Gang, Yann Gousseau, and Gui-Song Xia (2016a). "Texture Synthesis Through Convolutional Neural Networks and Spectrum Constraints". In: *23rd International Conference on Pattern Recognition*. ICPR. IEEE, pp. 3234–3239 (cit. on pp. 72, 82, 83, 92, 118).

Liu, Li, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen (2020). "Deep Learning for Generic Object Detection: A Survey". In: *International Journal of Computer Vision* 128, pp. 261–318 (cit. on p. 35).

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg (2016b). "SSD: Single Shot MultiBox Detector". In: *European Conference on Computer Vision*. Springer, Cham, pp. 21–37 (cit. on p. 70).

Lombardi, Thomas Edward (2005). "The Classification of Style in Fine-Art Painting". Doctor of Professional Studies in Computing. School of Computer Science and Information Systems Pace University (cit. on p. 43).

Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2, pp. 91–110 (cit. on pp. 18, 29).

Lu, Yang, Song-Chun Zhu, and Ying Nian Wu (2015). "Learning FRAME Models Using CNN Filters". In: *Thirtieth AAAI Conference on Artificial Intelligence* (cit. on pp. 82, 83, 103).

Madhu, Prathmesh, Ronak Kosti, Lara Mührenberg, Peter Bell, Andreas Maier, and Vincent Christlein (2019). "Recognizing Characters in Art History Using Deep Learning". In: *Proceedings of the 1st Workshop on Structuring and Understanding of Multimedia heritAge Contents* (Nice, France). SUMAC '19. New York, NY, USA: ACM, pp. 15–22 (cit. on pp. 46, 47, 50).

Madhu, Prathmesh, Tilman Marquart, Ronak Kosti, Peter Bell, Andreas Maier, and Vincent Christlein (2020). "Understanding Compositional Structures in Art Historical Images Using Pose and Gaze Priors". In: *ECCV 2020 Workshops (VISART V)* (cit. on p. 53).

Mahajan, Dhruv, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten (2018). "Exploring the Limits of Weakly Supervised Pretraining". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 181–196 (cit. on p. 34).

Mahendran, Aravindh and Andrea Vedaldi (2015). "Understanding Deep Image Representations by Inverting Them". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196 (cit. on pp. 36, 39, 40, 77, 80, 90).

Malisiewicz, Tomasz, Abhinav Gupta, and Alexei A. Efros (2011). "Ensemble of Exemplar-SVMs for Object Detection and Beyond". In: *International Conference on Computer Vision*. Barcelona, Spain: IEEE, pp. 89–96 (cit. on p. 44).

Mallat, Stéphane (2016). "Understanding Deep Convolutional Networks". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065, p. 20150203 (cit. on pp. 26, 36, 76).

Mao, Hui, Ming Cheung, and James She (2017). "DeepArt : Learning Joint Representations of Visual Arts". In: *Proceedings of the 2017 ACM on Multimedia Conference*. ACM Press, pp. 1183–1191 (cit. on pp. 46, 47, 51, 126).

McCormick, Bruce H. and Sadali N. Jayaramamurthy (1974). "Time Series Model for Texture Synthesis". In: *International Journal of Computer & Information Sciences* 3.4, pp. 329–343 (cit. on pp. 72, 73).

Mensink, Thomas and Jan van Gemert (2014). "The Rijksmuseum Challenge: Museum-Centered Visual Recognition". In: *Proceedings of International Conference on Multimedia Retrieval*. ACM Press, pp. 451–454 (cit. on pp. 43, 45, 47, 126).

Morcos, Ari, Maithra Raghu, and Samy Bengio (2018). "Insights on Representational Similarity in Neural Networks with Canonical Correlation". In: *Neural Information Processing System*. NIPS, p. 10 (cit. on p. 41).

Mordvintsev, Alexander, Christopher Olah, and Mike Tyka (2015). *Inceptionism: Going Deeper into Neural Networks*. Google Research Blog. URL: `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html` (visited on 03/12/2018) (cit. on pp. 37, 38).

Moretti, Franco (2000). "Conjectures on World Literature". In: *New left review*, pp. 54–68 (cit. on p. 19).

Mueller, Manfred and Andreas Burmester (1993). "Registration of Transportation Damages Using a High-Resolution CCD Camera". In: *Recording Systems: High-Resolution Cameras and Recording Devices and Laser Scanning and Recording Systems*. Vol. 1987. International Society for Optics and Photonics, pp. 111–117 (cit. on p. 42).

Nakano, Reiichiro (2019). "A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features': Adversarially Robust Neural Style Transfer". In: *Distill* 4.8, e00019.4 (cit. on p. 79).

Neyshabur, Behnam, Hanie Sedghi, and Chiyuan Zhang (2020). "What Is Being Transferred in Transfer Learning?" In: *Advances in Neural Information Processing Systems* 33 (cit. on pp. 40, 41, 144).

Nguyen, Anh, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune (2016). "Synthesizing the Preferred Inputs for Neurons in Neural Networks via Deep Generator Networks". In: *30th Conference on Neural Information Processing Systems (NIPS)*. Barcelona, Spain (cit. on p. 37).

Nguyen, Anh, Jason Yosinski, and Jeff Clune (2015). "Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images". In: *EEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, pp. 427–436 (cit. on pp. 37, 39).

Nguyen, Minh Hoai, Lorenzo Torresani, Fernando de la Torre, and Carsten Rother (2009). "Weakly Supervised Discriminative Localization and Classification: A Joint Learning Process". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009 IEEE 12th International Conference on Computer Vision, pp. 1925–1932 (cit. on pp. 59, 63–65, 154).

Novak, Roman and Yaroslav Nikulin (2016). *Improving the Neural Algorithm of Artistic Style*. Version 1. arXiv: 1605.04603 [cs] (cit. on pp. 83, 93).

Odena, Augustus, Vincent Dumoulin, and Chris Olah (2016). "Deconvolution and Checkerboard Artifacts". In: *Distill* 1.10, e3 (cit. on p. 118).

Offert, Fabian (2018). "Images of Image Machines. Visual Interpretability in Computer Vision for Art". In: *Computer Vision – ECCV 2018 Workshops*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 710–715 (cit. on pp. 52, 190).

Offert, Fabian and Peter Bell (2020). "Perceptual Bias and Technical Metapictures: Critical Machine Vision as a Humanities Challenge". In: *AI & SOCIETY* (cit. on pp. 190, 191).

Olah, Chris, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter (2020a). "An Overview of Early Vision in InceptionV1". In: *Distill* 5.4, e00024–002 (cit. on pp. 38, 190).

— (2020b). "Zoom In: An Introduction to Circuits". In: *Distill* 5, e00024–001 (cit. on pp. 37, 38).

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature Visualization". In: *Distill* 2.11, e7 (cit. on pp. 18, 36–38, 136, 137).

Olah, Chris, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev (2018). "The Building Blocks of Interpretability". In: *Distill* 3.3, e10 (cit. on pp. 37, 38, 190).

Oquab, Maxime, Léon Bottou, Ivan Laptev, and Josef Sivic (2014). "Learning and Transferring Mid-Level Image Representations Using Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1717–1724 (cit. on p. 32).

— (2015). "Is Object Localization for Free? - Weakly-Supervised Learning with Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 685–694 (cit. on pp. 18, 65, 66, 154).

Orsenigo, Carlotta and Carlo Vercellis (2007). "Accurately Learning from Few Examples with a Polyhedral Classifier". In: *Computational Optimization and Applications* 38.2, pp. 235–247 (cit. on p. 156).

Pan, Sinno Jialin and Qiang Yang (2010). "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359 (cit. on pp. 31, 35).

Panofsky, Erwin (1939). *Studies in Iconology: Humanistic Themes in the Art of the Renaissance*. New York: Harper & Row. 262 pp. (cit. on p. 54).

Park, Joo Hyun, Song Park, and Hyunjung Shim (2019). "Semantic-Aware Neural Style Transfer". In: *Image and Vision Computing* 87, pp. 13–23 (cit. on p. 81).

Paul, Arnab and Suresh Venkatasubramanian (2015). *Why Does Deep Learning Work? - A Perspective from Group Theory*. Version 3. arXiv: 1412.6621 [cs, stat] (cit. on p. 36).

Paumard, Marie-Morgane, David Picard, and Hedi Tabia (2018). "Jigsaw Puzzle Solving Using Local Feature Co-Occurrences in Deep Neural Networks". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. Athens: IEEE, pp. 1018–1022 (cit. on p. 48).

Perez, Luis and Jason Wang (2017). *The Effectiveness of Data Augmentation in Image Classification Using Deep Learning*. Version 1. arXiv: 1712.04621 [cs] (cit. on p. 50).

Peyré, Gabriel (2009). "Sparse Modeling of Textures". In: *Journal of Mathematical Imaging and Vision* 34.1, pp. 17–31 (cit. on p. 76).

Pirovano, Antoine, Hippolyte Heuberger, Sylvain Berlemont, Saïd Ladjal, and Isabelle Bloch (2020). "Improving Interpretability for Computer-Aided Diagnosis Tools on Whole Slide Imaging with Multiple Instance Learning and Gradient-Based Explanations". In: *Interpretable and Annotation-Efficient Learning for Medical Image Computing*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 43–53 (cit. on p. 66).

Polatkan, Güngör, Sina Jafarpour, Andrei Brasoveanu, Shannon Hughes, and Ingrid Daubechies (2009). "Detection of Forgery in Paintings Using Supervised Learning". In: *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 2921–2924 (cit. on p. 43).

Pollock, Griselda (2014). *Computers Can Find Similarities between Paintings – but Art History Is about so Much More*. The Conversation. URL: http://theconversation.com/computers-can-find-similarities-between-paintings-but-art-history-is-about-so-much-more-30752 (visited on 11/13/2020) (cit. on p. 53).

Portilla, Javier and Eero P. Simoncelli (2000). "A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients". In: *International journal of computer vision* 40.1, pp. 49–70 (cit. on pp. 20, 72–74, 76, 78, 83, 89, 91, 120).

Posthumus, Etienne (2020). *Brill Iconclass AI Test Set*. URL: https://labs.brill.com/ictestset/ (visited on 03/04/2020) (cit. on pp. 126, 189).

Pratt, Lorien Y, Jack Mostow, and Candace A Kamm (1991). "Direct Transfer of Learned Information Among Neural Networks". In: *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*. Vol. 91. Anaheim, CA, pp. 584–589 (cit. on p. 31).

Raad, Lara and Bruno Galerne (2017a). "Efros and Freeman Image Quilting Algorithm for Texture Synthesis". In: *Image Processing On Line* 7, pp. 1–22 (cit. on p. 75).

— (2017b). "Efros and Freeman Image Quilting Algorithm for Texture Synthesis". In: *Image Processing On Line* 7, pp. 1–22 (cit. on pp. 95, 103).

Raad, Lara Cisa, AXEL DAVY, Agnès Desolneux, and Jean-Michel Morel (2018). "A Survey of Exemplar-Based Texture Synthesis". In: *Annals of Mathematical Sciences and Applications* 3.1, pp. 89–148 (cit. on p. 120).

Raghu, Maithra, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein (2017). "SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 6076–6085 (cit. on pp. 36, 40, 41).

Raghu, Maithra, Jon Kleinberg, Chiyuan Zhang, and Samy Bengio (2019). "Transfusion: Understanding Transfer Learning for Medical Imaging". In: *Advances in Neural Information Processing Systems*. NeurIPS, pp. 3342–3352 (cit. on pp. 33, 34, 41).

Rahmani, Rouhollah, Sally A. Goldman, Hui Zhang, John Krettek, and Jason E. Fritts (2005). "Localized Content Based Image Retrieval". In: *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval* (Hilton, Singapore). MIR'05. New York, NY, USA: ACM, pp. 227–236 (cit. on pp. 185, 311, 313).

Ramon, Jan and Luc De Raedt (2000). "Multi Instance Neural Networks". In: *Proceedings of the ICML-2000 Workshop on Attribute-Value and Relational Learning*. Pp. 53–60 (cit. on pp. 57, 59, 163).

Ranzato, Marc'Aurelio, Christopher Poultney, Sumit Chopra, and Yann LeCun (2006). "Efficient Learning of Sparse Representations with an Energy-Based Model". In: *Advances in Neural Information Processing Systems*. NIPS. Vol. 19, pp. 1137–1144 (cit. on p. 28).

Redies, Christoph and Anselm Brachmann (2017a). "Statistical Image Properties in Large Subsets of Traditional Art, Bad Art, and Abstract Art". In: *Frontiers in Neuroscience* 11 (cit. on p. 42).

Redies, Christoph, Anselm Brachmann, and Johan Wagemans (2017b). "High Entropy of Edge Orientations Characterizes Visual Artworks from Diverse Cultural Backgrounds". In: *Vision Research* 133, pp. 130–144 (cit. on pp. 42, 43).

Redmon, Joseph, Santosh Divvala, Ross B. Girshick, and Ali Farhadi (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788 (cit. on pp. 35, 47, 128).

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: Better, Faster, Stronger". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR, pp. 7263–7271 (cit. on p. 128).

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in neural information processing systems*, pp. 91–99 (cit. on pp. 35, 63, 128, 161, 163, 174).

Ren, Zhongzheng, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Yong Jae Lee, Alexander G. Schwing, and Jan Kautz (2020). "Instance-Aware, Context-Focused, and Memory-Efficient Weakly Supervised Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10598–10607 (cit. on pp. 64, 69).

Resig, John (2014). "Using Computer Vision to Increase the Research Potential of Photo Archives". In: *Journal of Digital Humanities* (cit. on p. 19).

Rijksmuseum (2018). *Online Collection Catalogue - Research*. Rijksmuseum. URL: https://www.rijksmuseum.nl/en/research/online-collection-catalogue (visited on 07/07/2018) (cit. on p. 126).

Risser, Eric, Pierre Wilmot, and Connelly Barnes (2017). *Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses*. Version 2. arXiv: 1701.08893 [cs] (cit. on pp. 81, 82, 84, 91, 112, 117).

Romero, Miguel, Yannet Interian, Timothy Solberg, and Gilmer Valdes (2020). "Targeted Transfer Learning to Improve Performance in Small Medical Physics Datasets". In: *Medical Physics* (cit. on pp. 33, 34).

Rosasco, Lorenzo, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri (2004). "Are Loss Functions All the Same?" In: *Neural Computation* 16.5, pp. 1063–1076 (cit. on p. 159).

Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization". In: *Psychological Review* 65.6, pp. 386–408 (cit. on pp. 27, 153).

Rosenfeld, Amir, Richard Zemel, and John K. Tsotsos (2018). *The Elephant in the Room*. Version 1. arXiv: 1808.03305 [cs] (cit. on p. 170).

Rosset, Saharon, Ji Zhu, and Trevor J Hastie (2004). "Margin Maximizing Loss Functions". In: *Advances in Neural Information Processing Systems*, pp. 1237–1244 (cit. on p. 159).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088, pp. 533–536 (cit. on pp. 27, 28).

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision*. Springer 115.3, pp. 211–252 (cit. on pp. 17, 25, 32, 125).

Sabatelli, Matthia, Mike Kestemont, Walter Daelemans, and Pierre Geurts (2018). "Deep Transfer Learning for Art Classification Problems". In: *Workshop on Computer Vision for Art Analysis ECCV*. Munich, pp. 1–17 (cit. on pp. 46, 52, 132).

Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton (2017). "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems*, pp. 3856–3866 (cit. on pp. 35, 64).

Saleh, Babak, Kanako Abe, Ravneet Singh Arora, and Ahmed Elgammal (2014). "Toward Automated Discovery of Artistic Influence". In: *Multimedia Tools and Applications* 75.7, pp. 3565–3591 (cit. on pp. 43, 53).

Saleh, Babak and Ahmed Elgammal (2016). "Large-Scale Classification of Fine-Art Paintings: Learning The Right Metric on The Right Feature". In: *International Journal for Digital Art History* 2 (cit. on p. 45).

Sanakoyeu, Artsiom, Dmytro Kotovenko, Sabine Lang, and Bjorn Ommer (2018). "A Style-Aware Content Loss for Real-Time HD Style Transfer". In: *ECCV*, p. 17 (cit. on p. 81).

Sarı, Cihan, Albert Ali Salah, and Alkım Almıla Akdag Salah (2019). "Automatic Detection and Visualization of Garment Color in Western Portrait Paintings". In: *Digital Scholarship in the Humanities* (cit. on pp. 50, 53).

Schmidhuber, Jürgen (2015). "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61, pp. 85–117 (cit. on p. 27).

Schreiber, Shaun, Jaco Geldenhuys, and Hendrik de Villiers (2016). "Texture Synthesis Using Convolutional Neural Networks with Long-Range Consistency and Spectral Constraints". In: *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, pp. 1–6 (cit. on p. 83).

Schreibman, Susan, Ray Siemens, and John Unsworth (2004). *Companion to Digital Humanities*. Hardcover. Blackwell Companions to Literature and Culture. Oxford: Blackwell Publishing Professional (cit. on p. 18).

Seguin, Benoit (2018). "The Replica Project: Building a Visual Search Engine for Art Historians". In: *XRDS* 24.3, pp. 24–29 (cit. on pp. 19, 54).

Seguin, Benoit, Lisandra Costiner, Isabella di Lenardo, and Frédéric Kaplan (2018). "New Techniques for the Digitization of Art Historical Photographic Archives - the Case of the Cini Foundation in Venice". In: *Archiving Conference* 2018.1, pp. 1–5 (cit. on p. 19).

Seguin, Benoit, Striolo Carlotta, Isabella diLenardo, and Kaplan Frederic (2016). "Visual Link Retrieval in a Database of Paintings". In: *Computer Vision – ECCV 2016 Workshops* (cit. on pp. 45, 46, 48, 49, 54).

Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra (2017). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *ICCV* (cit. on p. 36).

Sendik, Omry and Daniel Cohen-Or (2017). "Deep Correlations for Texture Synthesis". In: *ACM Transactions on Graphics (TOG)* 36.5, pp. 1–15 (cit. on pp. 82, 83, 89, 93–100, 104, 106, 108–110, 228–262).

Sermanet, Pierre, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun (2013a). "OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks". In: *International Conference on Learning Representations (ICLR2014), CBLS* (cit. on pp. 32, 67).

Sermanet, Pierre, Koray Kavukcuoglu, Soumith Chintala, and Yann Lecun (2013b). "Pedestrian detection with unsupervised multi-stage feature learning". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 3626–3633 (cit. on p. 32).

Shaham, Tamar Rott, Tali Dekel, and Tomer Michaeli (2019). "SinGAN: Learning a Generative Model From a Single Natural Image". In: *ICCV*, p. 11 (cit. on pp. 84, 85).

Shen, Xi, Alexei A. Efros, and Mathieu Aubry (2019). "Discovering Visual Patterns in Art Collections with Spatially-Consistent Feature Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR (cit. on p. 48).

Shi, Rui, Tianxing Li, and Yasushi Yamaguchi (2020a). "Group Visualization of Class-Discriminative Features". In: *Neural Networks* 129, pp. 75–90 (cit. on p. 38).

Shi, Wu and Yu Qiao (2020b). "Fast Texture Synthesis via Pseudo Optimizer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 81–83).

Shrivastava, Abhinav, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros (2011). "Data-Driven Visual Similarity for Cross-Domain Image Matching". In: *Proceedings of the 2011 SIGGRAPH Asia Conference*. ACM Press, pp. 1–10 (cit. on p. 44).

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman (2014). "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *ICLR*, pp. 1–8 (cit. on pp. 29, 36–38, 65, 78, 79).

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations* (cit. on pp. 30, 73, 125, 129, 131, 321).

Siva, Parthipan and Tao Xiang (2011). "Weakly Supervised Object Detector Learning with Model Drift Detection". In: *2011 International Conference on Computer Vision*. (ICCV). Barcelona, Spain: IEEE, pp. 343–350 (cit. on pp. 62, 64).

Smilkov, Daniel, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg (2017). "SmoothGrad: Removing Noise by Adding Noise". In: *Workshop on Visualization for Deep Learning*. ICML (cit. on p. 36).

Smirnov, Stanislav and Alma Eguizabal (2019). "Deep Learning for Object Detection in Fine-Art Paintings". In: *2018 Metrology for Archaeology and Cultural Heritage (MetroArchaeo)*. IEEE, pp. 45–49 (cit. on p. 50).

Snelgrove, Xavier (2017). "High-Resolution Multi-Scale Neural Texture Synthesis". In: *SIGGRAPH Asia*. ACM Press, pp. 1–4 (cit. on pp. 72, 84, 91, 95, 97–101, 103, 104, 106, 108–110, 114, 115, 120, 228–262, 274–280).

Song, Hyun Oh, Ross Girshick, Stefanie Jegelka, Julien Mairal, Zaid Harchaoui, and Trevor Darrell (2014). "On Learning to Localize Objects with Minimal Supervision". In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. Beijing, China, p. 9 (cit. on pp. 63, 65, 70).

Spratt, Emily L. (2017). "Dream Formulations and Deep Neural Networks: Humanistic Themes in the Iconology of the Machine-Learned Image". In: *kunsttexte.de* (cit. on p. 53).

Stoet, Gijsbert (2010). "PsyToolkit: A Software Package for Programming Psychological Experiments Using Linux". In: *Behavior Research Methods* 42.4, pp. 1096–1104 (cit. on p. 106).

— (2017). "PsyToolkit: A Novel Web-Based Method for Running Online Questionnaires and Reaction-Time Experiments". In: *Teaching of Psychology* 44.1, pp. 24–31 (cit. on p. 106).

Strezoski, Gjorgji, Rogier Knoester, Nanne van Noord, and Marcel Worring (2020). "OmniEyes: Analysis and Synthesis of Artistically Painted Eyes". In: *MultiMedia Modeling*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 628–641 (cit. on p. 47).

Strezoski, Gjorgji and Marcel Worning (2018). "OmniArt: A Large-Scale Artistic Benchmark". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) - Special Section on Deep Learning for Intelligent Multimedia Analytics*. Transactions on Multimedia Computing, Communications, and Applications (TOMM) 14.4 (cit. on pp. 47, 126, 189, 191).

Strezoski, Gjorgji and Marcel Worring (2017a). *OmniArt: Multi-Task Deep Learning for Artistic Data Analysis*. Version 1. arXiv: `1708.00684 [cs]` (cit. on p. 45).

— (2017b). "Plug-and-Play Interactive Deep Network Visualization". In: *VADL: Visual Analytics for Deep Learning*, p. 7 (cit. on pp. 46, 52).

Strubell, Emma, Ananya Ganesh, and Andrew McCallum (2019). "Energy and Policy Considerations for Deep Learning in NLP". In: *ACL*, p. 6 (cit. on p. 34).

Su, Hao, Jia Deng, and Li Fei-Fei (2016). "Crowdsourcing Annotations for Visual Object Detection". In: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, p. 7 (cit. on p. 54).

Sun, Baochen, Jiashi Feng, and Kate Saenko (2016a). "Return of Frustratingly Easy Domain Adaptation". In: AAAI Conference on Artificial Intelligence (cit. on p. 35).
Sun, Baochen and Kate Saenko (2016b). "Deep Coral: Correlation Alignment for Deep Domain Adaptation". In: *Computer Vision–ECCV 2016 Workshops*. Springer, pp. 443–450 (cit. on p. 35).
Sun, Chen, Manohar Paluri, Ronan Collobert, Ram Nevatia, and Lubomir Bourdev (2016c). "ProNet: Learning to Propose Object-Specific Boxes for Cascaded Neural Networks". In: *CVPR* (cit. on p. 66).
Sun, Chen, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta (2017). "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *Proceedings of the IEEE International Conference on Computer Vision*. ICCV, pp. 843–852 (cit. on p. 34).
Sun, Weihan and K. Kise (2009). "Speeding up the Detection of Line Drawings Using a Hash Table". In: *2009 Chinese Conference on Pattern Recognition*. IEEE (cit. on p. 44).
Sundararajan, Mukund, Ankur Taly, and Qiqi Yan (2017). "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Sydney, Australia: PMLR (cit. on p. 36).
Szabó, Róbert, Dániel Katona, Márton Csillag, Adrián Csiszárik, and Dániel Varga (2020). "Visualizing Transfer Learning". In: *ICML Workshop on Human Interpretability in Machine Learning (WHI 2020)*. ICML (cit. on p. 38).
Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. (Cit. on pp. 29, 128, 129).
Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). "Going Deeper with Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9 (cit. on pp. 29, 30, 131, 137, 315, 316).
Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus (2014). *Intriguing Properties of Neural Networks*. Version 4. arXiv: `1312.6199 [cs]` (cit. on pp. 37, 39).
Tajbakhsh, Nima, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang (2016). "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?" In: *IEEE Transactions on Medical Imaging* 35.5, pp. 1299–1312 (cit. on p. 18).
Tan, Mingxing and Quoc V. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *ICML 2019* (cit. on pp. 29, 31, 129, 189).
Tan, W. R., C. S. Chan, H. E. Aguirre, and K. Tanaka (2016). "Ceci n'est Pas Une Pipe: A Deep Convolutional Network for Fine-Art Paintings Classification". In: *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3703–3707 (cit. on pp. 46, 47, 52, 53, 125, 133).
Tang, Peng, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Yuille (2018a). "PCL: Proposal Cluster Learning for Weakly Supervised Object Detection". In: *IEEE transactions on pattern analysis and machine intelligence* (cit. on pp. 62, 69, 162, 165, 167, 168, 303–306).
Tang, Peng, Xinggang Wang, Xiang Bai, and Wenyu Liu (2017a). "Multiple Instance Detection Network with Online Instance Classifier Refinement". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3059–3067 (cit. on pp. 60, 68, 69).
Tang, Peng, Xinggang Wang, Angtian Wang, Yongluan Yan, Wenyu Liu, Junzhou Huang, and Alan Yuille (2018b). "Weakly Supervised Region Proposal Network and Object Detec-

tion". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 352–368 (cit. on p. 69).

Tang, Y., J. Wang, X. Wang, B. Gao, E. Dellandréa, R. Gaizauskas, and L. Chen (2017b). "Visual and Semantic Knowledge Transfer for Large Scale Semi-Supervised Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.12, pp. 3045–3058 (cit. on p. 70).

Tang, Yuxing, Xiaofang Wang, Emmanuel Dellandrea, and Liming Chen (2017c). "Weakly Supervised Learning of Deformable Part-Based Models for Object Detection via Region Proposals". In: *IEEE Transactions on Multimedia* 19.2, pp. 393–407 (cit. on pp. 64, 158).

Tartavel, Guillaume, Yann Gousseau, and Gabriel Peyré (2015). "Variational Texture Synthesis with Sparsity and Spectrum Constraints". In: *Journal of Mathematical Imaging and Vision* 52.1, pp. 124–144 (cit. on pp. 76, 91, 93).

Taylor, Richard (2002). "Order in Pollock's Chaos". In: *Scientific American* 287.6, pp. 116–121 (cit. on p. 42).

Taylor, Richard P, Adam P Micolich, and David Jonas (1999). "Fractal Analysis of Pollock's Drip Paintings". In: *Nature* 422.399, p. 2 (cit. on p. 42).

Teh, EuWern, Mrigank Rochan, and Yang Wang (2016). "Attention Networks for Weakly Supervised Object Localization". In: *Procedings of the British Machine Vision Conference 2016*. York, UK: British Machine Vision Association, pp. 52.1–52.11 (cit. on pp. 65, 66).

The Metropolitan Museum of Art (2017). *Press Conference to Announce New Open Access Policy*. The Metropolitan Museum of Art. URL: `https://www.metmuseum.org/metmedia/video/news/open-access-press-conference` (visited on 11/27/2020) (cit. on p. 18).

Theis, Lucas and Matthias Bethge (2015). "Generative Image Modeling Using Spatial LSTMs". In: *Advances in Neural Information Processing Systems 28* (cit. on p. 84).

Thomas, Christopher and Adriana Kovashka (2018). "Artistic Object Recognition by Unsupervised Style Adaptation". In: *Asian Conference on Computer Vision*. Cham: Springer, pp. 460–476 (cit. on pp. 20, 48, 50, 166, 303, 307).

Tian, Yingtao, Chikahiko Suzuki, Tarin Clanuwat, Mikel Bober-Irizar, Alex Lamb, and Asanobu Kitamoto (2020). *KaoKore: A Pre-Modern Japanese Art Facial Expression Dataset*. Version 1. arXiv: `2002.08595 [cs, stat]` (cit. on pp. 46, 47, 189).

Touvron, Hugo, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou (2020). "Fixing the Train-Test Resolution Discrepancy". In: *Advances in Neural Information Processing Systems*. NeurIPS 2019. Vancouver, Canada (cit. on p. 29).

Tubaro, Paola and Antonio A. Casilli (2019). "Micro-Work, Artificial Intelligence and the Automotive Industry". In: *Journal of Industrial and Business Economics* 46.3, pp. 333–345 (cit. on pp. 34, 54).

Uijlings, Jasper, Stefan Popov, and Vittorio Ferrari (2018). "Revisiting Knowledge Transfer for Training Object Class Detectors". In: *CVPR* (cit. on pp. 70, 158, 165, 290).

Uijlings, Jasper, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders (2013). "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104.2, pp. 154–171 (cit. on pp. 63, 166, 174).

Ulyanov, Dmitry, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky (2016). "Texture Networks: Feed-Forward Synthesis of Textures and Stylized Images". In: *ICML* 1.2, p. 4 (cit. on pp. 72, 81–83, 85, 95, 97–100, 104, 228–262).

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky (2017). "Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR, pp. 6924–6932 (cit. on pp. 81–83).

Um, Kiwon, Xiangyu Hu, Bing Wang, and Nils Thuerey (2019). "Spot the Difference: Accuracy of Numerical Simulations via the Human Visual System". In: *Journal of Computational Physics* (cit. on pp. 106, 111, 269, 270).

Ustyuzhaninov, Ivan, Wieland Brendel, Leon A. Gatys, and Matthias Bethge (2016). *Texture Synthesis Using Shallow Convolutional Networks with Random Filters*. Version 1. arXiv: 1606.00021 [cs] (cit. on p. 79).

Vakanski, Aleksandar, Min Xian, and Phoebe E. Freer (2020). "Attention-Enriched Deep Learning Model for Breast Tumor Segmentation in Ultrasound Images". In: *Ultrasound in Medicine & Biology* 46.10, pp. 2819–2833 (cit. on p. 35).

Van Noord, Nanne, E. Hendriks, and E. Postma (2015). "Toward Discovery of the Artist's Style: Learning to Recognize Artists by Their Artworks". In: *IEEE Signal Processing Magazine* 32.4, pp. 46–54 (cit. on pp. 47, 52).

Van Noord, Nanne and Eric Postma (2017). "Learning Scale-Variant and Scale-Invariant Features for Deep Image Classification". In: *Pattern Recognition* 61, pp. 583–592 (cit. on pp. 46, 50, 52, 188, 189).

Vanwinckelen, Gitte, Vinicius Tragante do O, Daan Fierens, and Hendrik Blockeel (2016). "Instance-Level Accuracy versus Bag-Level Accuracy in Multi-Instance Learning". In: *Data Mining and Knowledge Discovery* 30.2, pp. 313–341 (cit. on pp. 57, 170).

Viazovetskyi, Yuri, Vladimir Ivashkin, and Evgeny Kashin (2020). "StyleGAN2 Distillation for Feed-Forward Image Manipulation". In: *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, pp. 170–186 (cit. on p. 35).

Viola, Paul, John C Platt, and Cha Zhang (2005). "Multiple Instance Boosting for Object Detection". In: *Advances in Neural Information Processing Systems 18*. NIPS 2005, p. 8 (cit. on pp. 59, 311–313).

Wan, Fang, Pengxu Wei, Jianbin Jiao, Zhenjun Han, and Qixiang Ye (2018). "Min-Entropy Latent Model for Weakly Supervised Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1297–1306 (cit. on pp. 67, 69, 158, 190).

Wang, Chong, Weiqiang Ren, Kaiqi Huang, and Tieniu Tan (2014). "Weakly Supervised Object Localization with Latent Category Learning". In: *Computer Vision – ECCV 2014*. Vol. 8694. Cham: Springer International Publishing, pp. 431–445 (cit. on p. 65).

Wang, Liwei, Lunjia Hu, Jiayuan Gu, Yue Wu, Zhiqiang Hu, Kun He, and John Hopcroft (2018a). "Towards Understanding Learning Representations: To What Extent Do Different Neural Networks Learn the Same Representation". In: *Advances in Neural Information Processing Systems*. NeurIPS. Montréal, Canada, pp. 9584–9593 (cit. on p. 40).

Wang, Rui, Dhruv Mahajan, and Vignesh Ramanathan (2020). "What Leads to Generalization of Object Proposals?" In: *1st Visual Inductive Priors for Data-Efficient Deep Learning Workshop VIPriors*. ECCV 2020 (cit. on p. 170).

Wang, Shuning and Xusheng Sun (2005). "Generalization of Hinging Hyperplanes". In: *IEEE Transactions on Information Theory* 51.12, pp. 4425–4431 (cit. on p. 156).

Wang, Xinggang, Yongluan Yan, Peng Tang, Xiang Bai, and Wenyu Liu (2018b). "Revisiting Multiple Instance Neural Networks". In: *Pattern Recognition* 74, pp. 15–24 (cit. on pp. 59, 165, 167, 168, 303–310).

Warburg, Aby (2012). *L'atlas Mnémosyne (Avec Un Essai de Roland Recht)*. Trans. by Sacha Zilberfarb. L'Écarquillé/Institut national d'histoire de l'art. Paris (cit. on p. 54).

Wei, Longhui, An Xiao, Lingxi Xie, Xiaopeng Zhang, Xin Chen, and Qi Tian (2020). "Circumventing Outliers of AutoAugment with Knowledge Distillation". In: *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, pp. 608–625 (cit. on p. 31).

Wei, Li-Yi and Marc Levoy (2000). "Fast Texture Synthesis Using Tree-Structured Vector Quantization". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH'00*. ACM Press, pp. 479–488 (cit. on pp. 73, 75).

Wei, Zhenao, Lilang Xiong, Kazuki Mori, Tung Duc Nguyen, Ruck Thawonmas, Keiko Suzuki, and Masaaki Kidachi (2017). "Deep Features for Image Classification and Image Similarity Perception". In: *JADH*, pp. 60–62 (cit. on p. 51).

Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang (2016). "A Survey of Transfer Learning". In: *Journal of Big Data* 3.1 (cit. on pp. 31, 35).

Westlake, Nicholas, Hongping Cai, and Peter Hall (2016). "Detecting People in Artwork with CNNs". In: *Computer Vision – ECCV 2016 Workshops*. Cham: Springer International Publishing, pp. 825–841 (cit. on pp. 20, 47, 166, 167, 307).

White, Tom (2019). "Shared Visual Abstractions". In: *Workshop: Shared Visual Representations in Human and Machine Intelligence*. NeurIPS (cit. on p. 39).

Wilber, Michael J., Chen Fang, Hailin Jin, Aaron Hertzmann, John Collomosse, and Serge Belongie (2017). "BAM! The Behance Artistic Media Dataset for Recognition Beyond Photography". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1211–1220 (cit. on pp. 46, 47, 52, 189, 191).

Wölfflin, Heinrich (1915). *Principles of Art History*. Courier Corporation (cit. on p. 53).

Wu, Qi, Hongping Cai, and Peter Hall (2014). "Learning Graphs to Model Visual Objects across Different Depictive Styles". In: *Computer Vision – ECCV 2014*. Vol. 8695. Cham: Springer International Publishing, pp. 313–328 (cit. on p. 44).

Wu, Qi, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel (2017). "Visual Question Answering: A Survey of Methods and Datasets". In: *Computer Vision and Image Understanding*. Language in Vision 163, pp. 21–40 (cit. on p. 35).

Wu, Ying Nian, Song Chun Zhu, and Xiuwen Liu (2000). "Equivalence of Julesz Ensembles and FRAME Models". In: *International Journal of Computer Vision* 38.3, pp. 247–265 (cit. on p. 74).

Wynen, Daan, Cordelia Schmid, and Julien Mairal (2018). "Unsupervised Learning of Artistic Styles with Archetypal Style Analysis". In: *Advances in Neural Information Processing Systems*. NeurIPS, pp. 6584–6593 (cit. on pp. 49, 81).

Yan, Yongluan, Xinggang Wang, Xiaojie Guo, Jiemin Fang, Wenyu Liu, and Junzhou Huang (2018). "Deep Multi-Instance Learning with Dynamic Pooling". In: *Proceedings of The 10th Asian Conference on Machine Learning* PMLR.95, pp. 662–677 (cit. on p. 64).

Yeh, Mao-Chuang and Shuai Tang (2018). *Improved Style Transfer by Respecting Inter-Layer Correlations*. Version 1. arXiv: `1801.01933 [cs.CV]` (cit. on p. 83).

Yin, Rujie, Eric Monson, Elizabeth Honig, Ingrid Daubechies, and Mauro Maggioni (2016). "Object Recognition in Art Drawings: Transfer of a Neural Network". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai: IEEE, pp. 2299–2303 (cit. on pp. 45, 46, 51, 52, 137).

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson (2014). "How Transferable Are Features in Deep Neural Networks?" In: *Advances in Neural Information Processing Systems*. NIPS (cit. on p. 33).

Yu, Chun-Nam John and Thorsten Joachims (2009). "Learning Structural SVMs with Latent Variables". In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML'09*. Montreal, Quebec, Canada: ACM Press, pp. 1–8 (cit. on pp. 58, 65).

Zeiler, Matthew D. and Rob Fergus (2014). "Visualizing and Understanding Convolutional Networks". In: *European Conference on Computer Vision*. ECCV. Springer, Cham, pp. 818–833 (cit. on pp. 18, 32, 36, 40, 52).

Zhang, Chenyang, Christine Kaeser-Chen, Grace Vesom, Jennie Choi, Maria Kessler, and Serge Belongie (2019a). *The iMet Collection 2019 Challenge Dataset*. arXiv: 1906.00901 [cs] (cit. on pp. 47, 126).

Zhang, Qi and Sally A Goldman (2002). "EM-DD: An Improved Multiple-Instance Learning Technique". In: *Advances in Neural Information Processing Systems* 14, pp. 1073–1080 (cit. on pp. 59, 311–313).

Zhang, Richard, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang (2018a). "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR, pp. 586–595 (cit. on p. 28).

Zhang, Xiaolin, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang (2018b). "Adversarial Complementary Learning for Weakly Supervised Object Localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1325–1334 (cit. on p. 67).

Zhang, Xiaopeng, Jiashi Feng, Hongkai Xiong, and Qi Tian (2018c). "Zigzag Learning for Weakly Supervised Object Detection". In: *CVPR*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4262–4270 (cit. on p. 68).

Zhang, Xu, Svebor Karaman, and Shih-Fu Chang (2019b). "Detecting and Simulating Artifacts in GAN Fake Images". In: *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*. Delft, Netherlands, pp. 1–6 (cit. on p. 118).

Zhang, Yongqiang, Yancheng Bai, Mingli Ding, Yongqiang Li, and Bernard Ghanem (2018d). "W2F: A Weakly-Supervised to Fully-Supervised Framework for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR, pp. 928–936 (cit. on p. 69).

Zhong, Yuanyi, Jianfeng Wang, Jian Peng, and Lei Zhang (2020). "Boosting Weakly Supervised Object Detection with Progressive Knowledge Transfer". In: *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, pp. 615–631 (cit. on pp. 70, 158).

Zhou, Bolei, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba (2016). "Learning Deep Features for Discriminative Localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR. IEEE, pp. 2921–2929 (cit. on pp. 66–68).

Zhou, Yang, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang (2018). "Non-Stationary Texture Synthesis by Adversarial Expansion". In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 37.4, 49:1–49:13 (cit. on p. 84).

Zhou, Zhi-Hua and Min-Ling Zhang (2002). "Neural Networks for Multi-Instance Learning". In: *Proceedings of the International Conference on Intelligent Information Technology*. Beijing, China, pp. 455–459 (cit. on pp. 57, 59, 153, 154, 163).

Zhu, Ciyou, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal (1997). "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization". In: *ACM Trans. Math. Softw.* 23.4, pp. 550–560 (cit. on pp. 90, 103).

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros (2017a). "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks Web Page". In: *Computer Vision (ICCV), 2017 IEEE International Conference On* (cit. on p. 50).

Zhu, Song Chun, Xiu Wen Liu, and Ying Nian Wu (2000). "Exploring Texture Ensembles by Efficient Markov Chain Monte Carlo-Toward a "Trichromacy" Theory of Texture". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.6, pp. 554–569 (cit. on pp. 74, 82).

Zhu, Song Chun, Yingnian Wu, and David Mumford (1998). "Filters, Random Fields and Maximum Entropy (FRAME): Towards a Unified Theory for Texture Modeling". In:

International Journal of Computer Vision. Springer 27.2, pp. 107–126 (cit. on pp. 74, 82).

Zhu, Xiaojin (2005). "Semi-Supervised Learning Literature Survey". In: *University of Wisconsin-Madison Department of Computer Sciences*, p. 60 (cit. on p. 55).

Zhu, Yi, Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao (2017b). "Soft Proposal Networks for Weakly Supervised Object Localization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1859–1868 (cit. on pp. 60, 62, 63, 66, 162, 165, 167, 168, 303–306).

Zitnick, C. Lawrence and Piotr Dollár (2014). "Edge Boxes: Locating Object Proposals from Edges". In: *Computer Vision – ECCV 2014*. Vol. 8693. Cham: Springer International Publishing, pp. 391–405 (cit. on pp. 63, 64, 69, 172, 174).

Zoph, Barret, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, and Quoc V. Le (2020). "Rethinking Pre-Training and Self-Training". In: *NeurIPS* (cit. on p. 28).

Zoph, Barret and Quoc V. Le (2017a). "Neural Architecture Search with Reinforcement Learning". In: *ICLR* (cit. on p. 31).

Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le (2017b). "Learning Transferable Architectures for Scalable Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710 (cit. on p. 31).

Zujovic, Jana, Lisa Gandy, Scott Friedman, Bryan Pardo, and Thrasyvoulos N. Pappas (2009). "Classifying Paintings by Artistic Genre: An Analysis of Features Classifiers". In: *2009 IEEE International Workshop on Multimedia Signal Processing*. Rio De Janeiro, pp. 1–5 (cit. on p. 43).

List of Acronyms and Abbreviations

(SV)CCA Singular Vector Canonical Correlation Analysis
AP Average Precision
AUC Area Under receiver operating characteristic Curve
BoW Bag-of-Words
CAM Class Activation Mapping
CCA Canonical Correlation Analysis
CKA Centred Kernel Alignement
CNN Convolutional Neural Network
CPU Central Processing Unit
DFT Discrete Fourier Transform
DPM Deformable Part Model
DSLR Digital Single-Lens Reflex
FRAME Filters Random Fields and Maximum Entropy
FSD Fully Supervised Detection
FT Fine-Tuning
GAN Generative Adversarial Network
GAP Global Average Pooling
GPU Graphics Processing Unit
HOG Histogram of Oriented Gradient
ILSVRC ImageNet Large Scale Visual Recognition Challenge
IoU Intersection over Union
KL Kullback–Leibler
LSE Log-Sum-Exponential
LSTM Long Short-Term Memory
LSVM Latent SVM
MI-SVM Maximum Bag Margin SVM
mi-SVM Maximum Pattern Margin SVM
MIL Multiple Instance Learning
MLP Multi-Layer Perceptron
NMS Non Maximum Suppression
OICR Online Instance Classifier Refinement
PCA Principal Component Analysis
PCL Proposal Cluster Learning
R-CNN Region Convolutional Neural Network
RANSAC RANdom SAmple Consensus
ReLU Rectified Linear Unit
ResNet Residual Network
RGB Red Green Blue
RNN Recurrent Neural Network

ROI Region Of Interest
RPN Region Proposal Network
SGD Stochastic Gradient Descent
SIFT Scale-Invariant Feature Transform
SOTA state-of-the-art
SPN Soft Proposal Network
SVM Support Vector Machine
UAR Unweighted Average Recall
w/o without
WR Witness Rate
WSDDN Weakly Supervised Deep Detection Network
WSOD Weakly Supervised Object Detection

List of Tables

List of Figures

Annexes

Transfer Learning of Convolutional Neural Networks for Texture Synthesis and Visual Recognition in Artistic Images

---

Nicolas GONTHIER

Supervised by:

Yann Gousseau, Télécom Paris - Institut Polytechnique de Paris
Saïd Ladjal, Télécom Paris - Institut Polytechnique de Paris
Olivier Bonfait, Université de Bourgogne

A

Additional Results for Texture Synthesis with Convolutional Neural Networks

Contents

A.1 Additional Comparisons with Alternative Methods

In Figures A.1 to A.14, one can see the results of the different texture synthesize methods mentioned in Section 3.3.

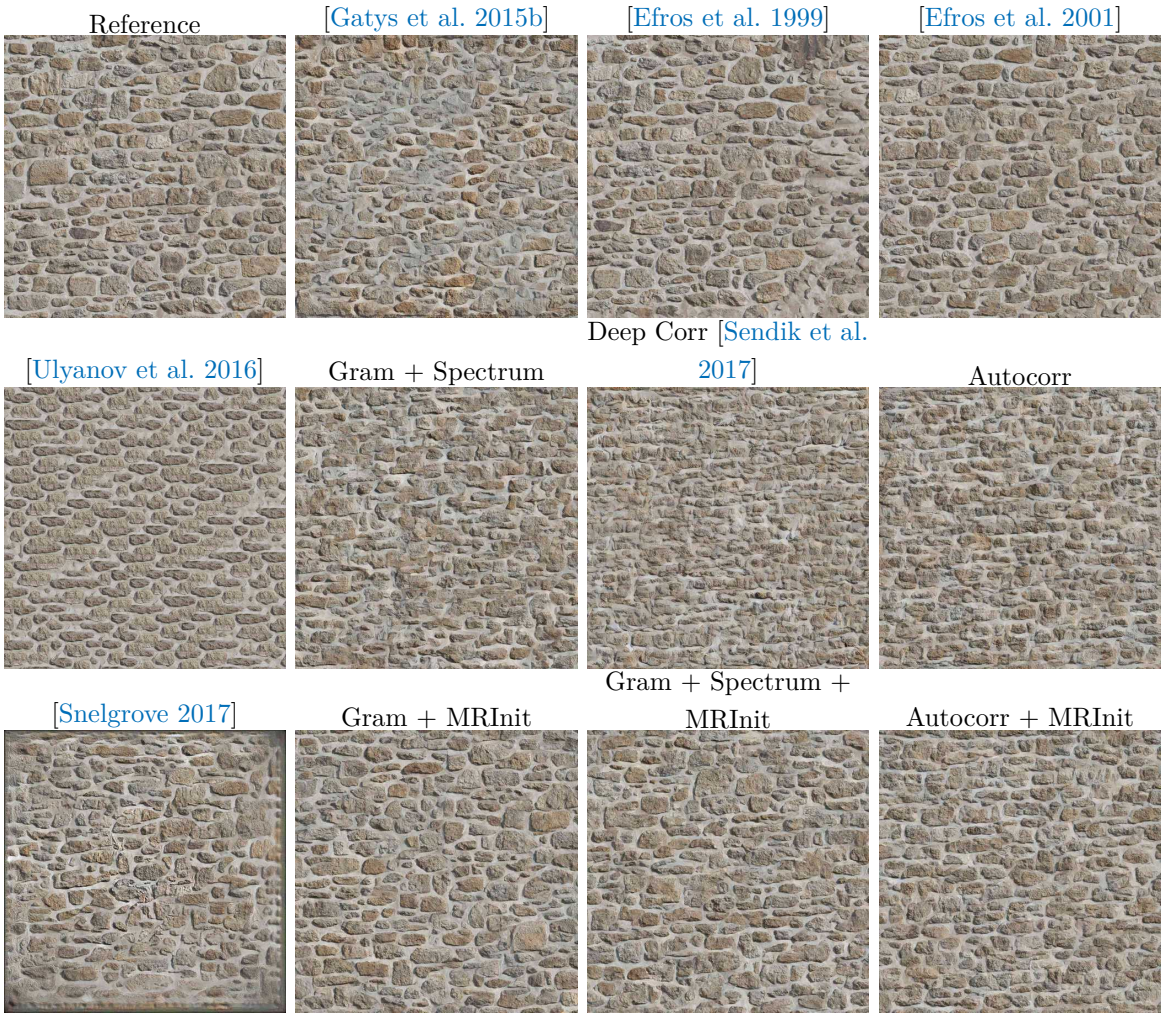


Figure A.1: Synthesis results using different methods for a given reference of size 1048×1048.

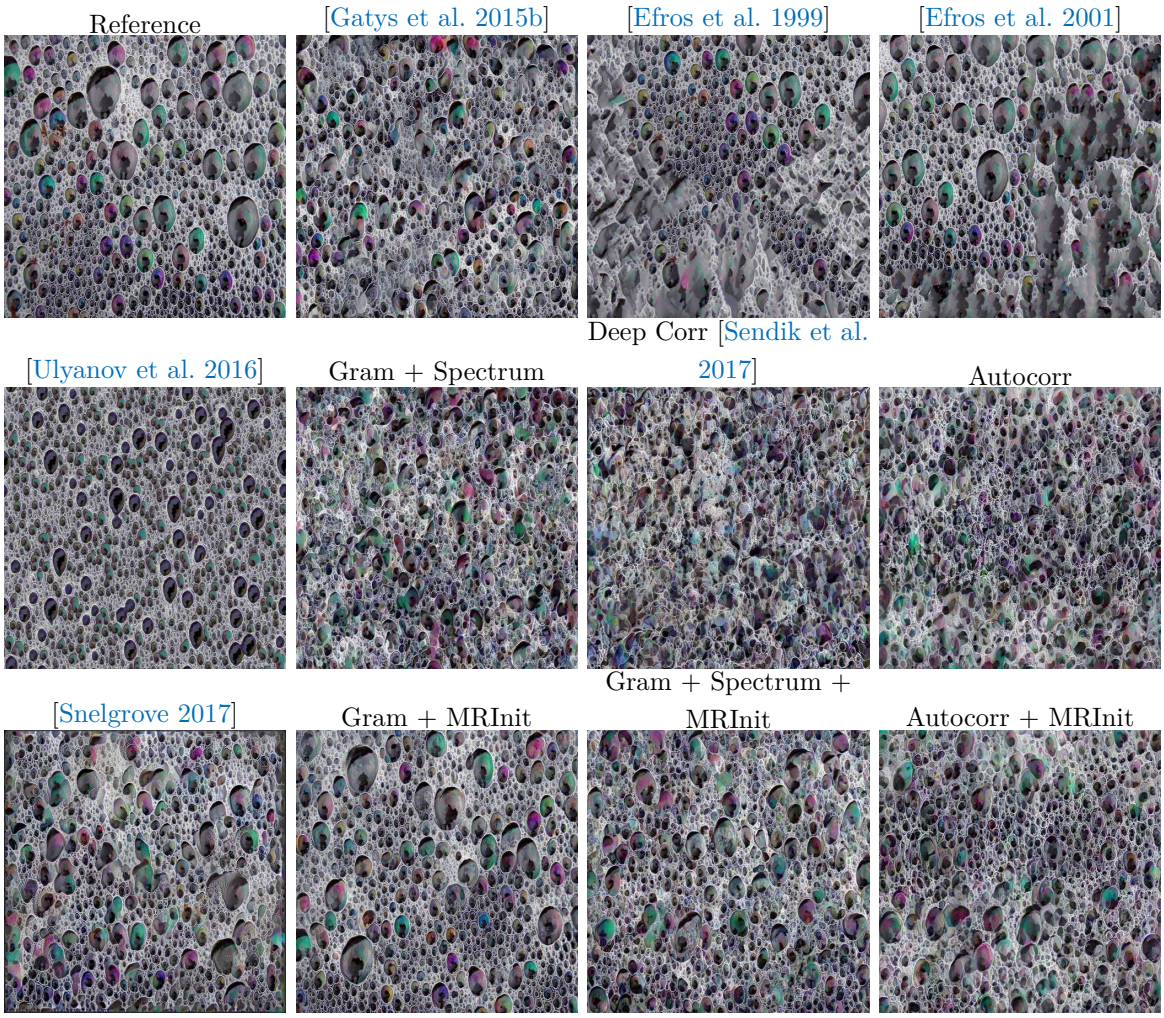


Figure A.2: Synthesis results using different methods for a given reference of size 1048×1048.

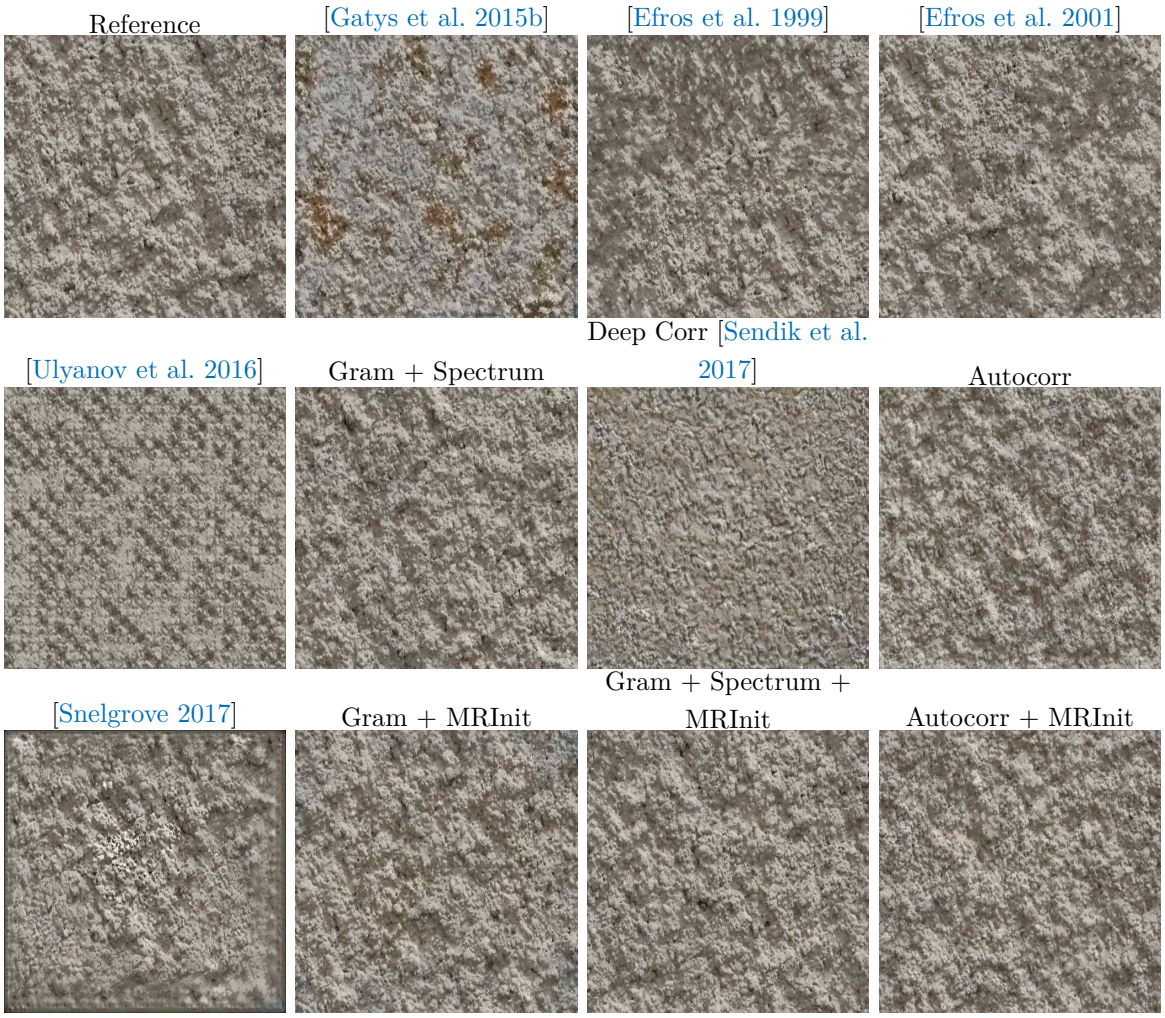


Figure A.3: Synthesis results using different methods for a given reference of size 1048×1048.

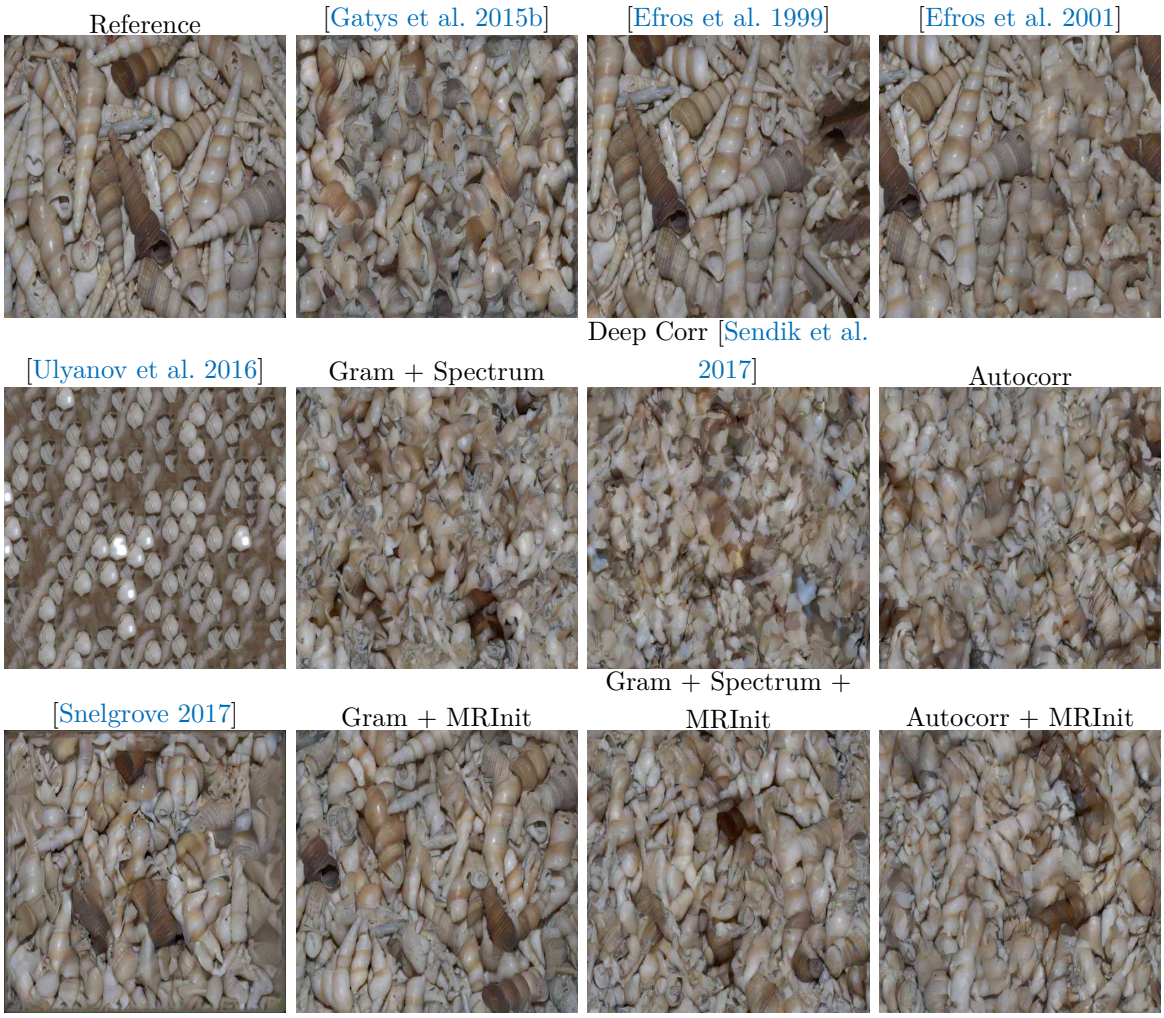


Figure A.4: Synthesis results using different methods for a given reference of size 1048×1048.

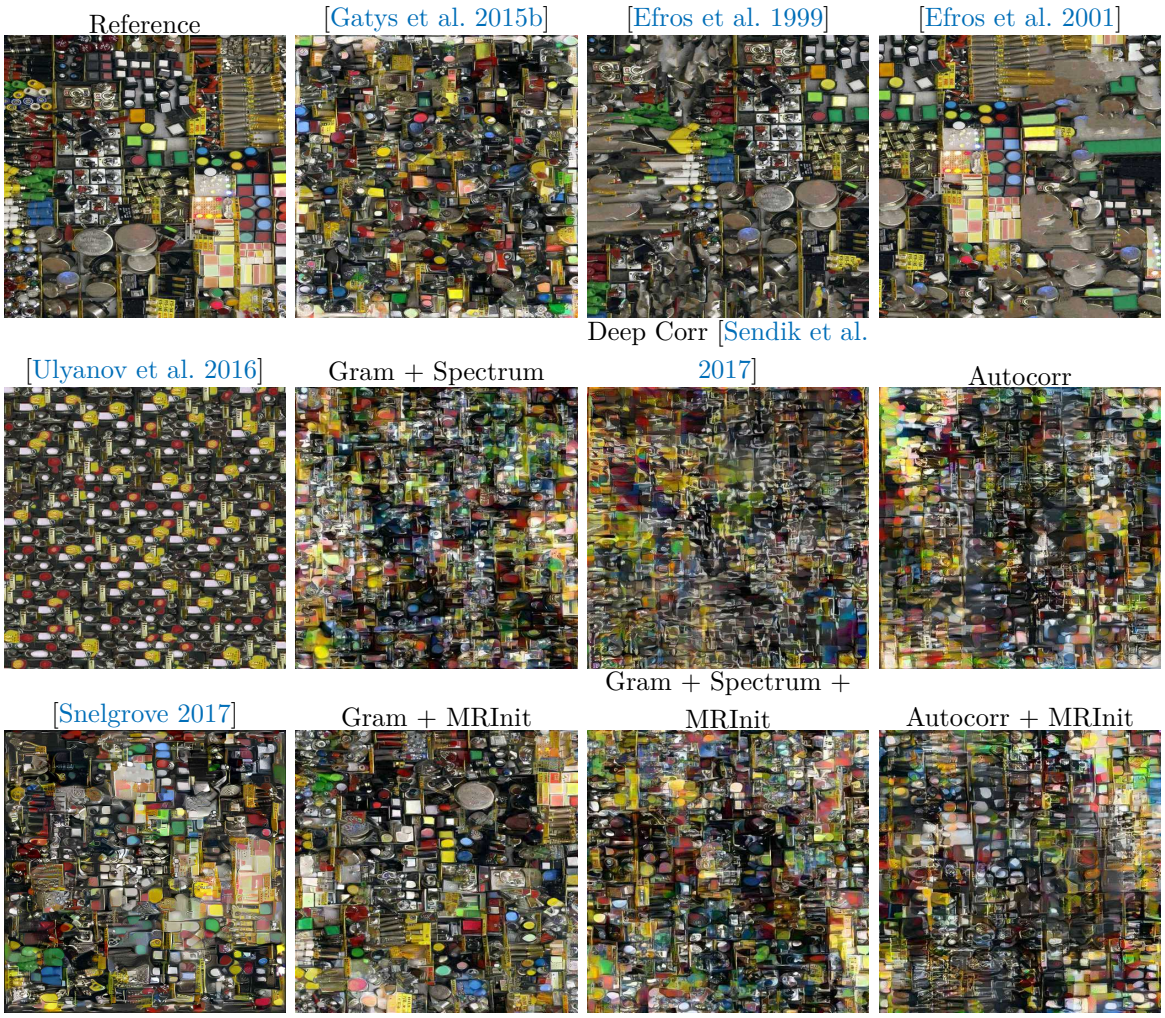


Figure A.5: Synthesis results using different methods for a given reference of size 1048×1048.

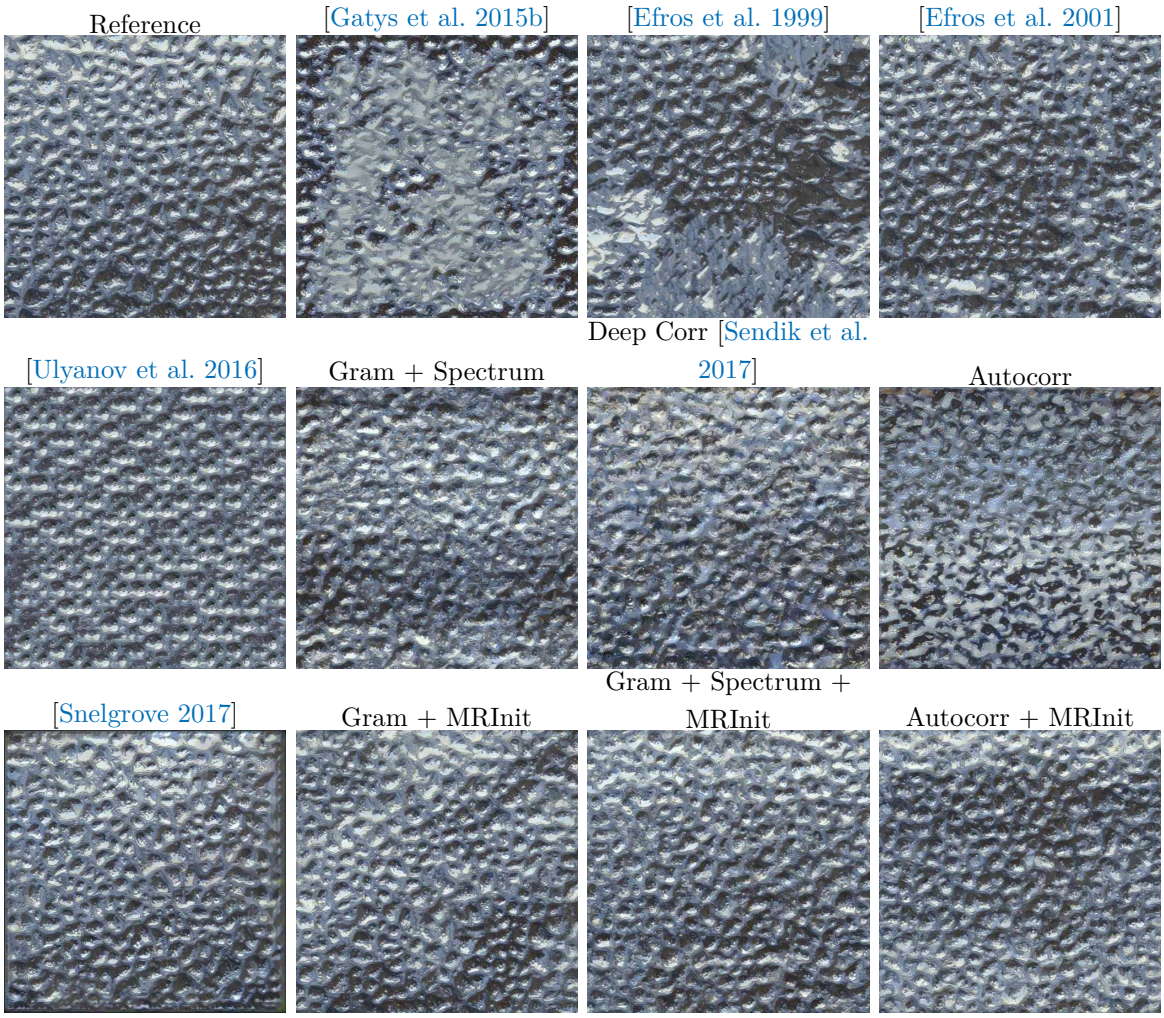


Figure A.6: Synthesis results using different methods for a given reference of size 1048×1048.

Figure A.7: Synthesis results using different methods for a given reference of size 1048×1048.

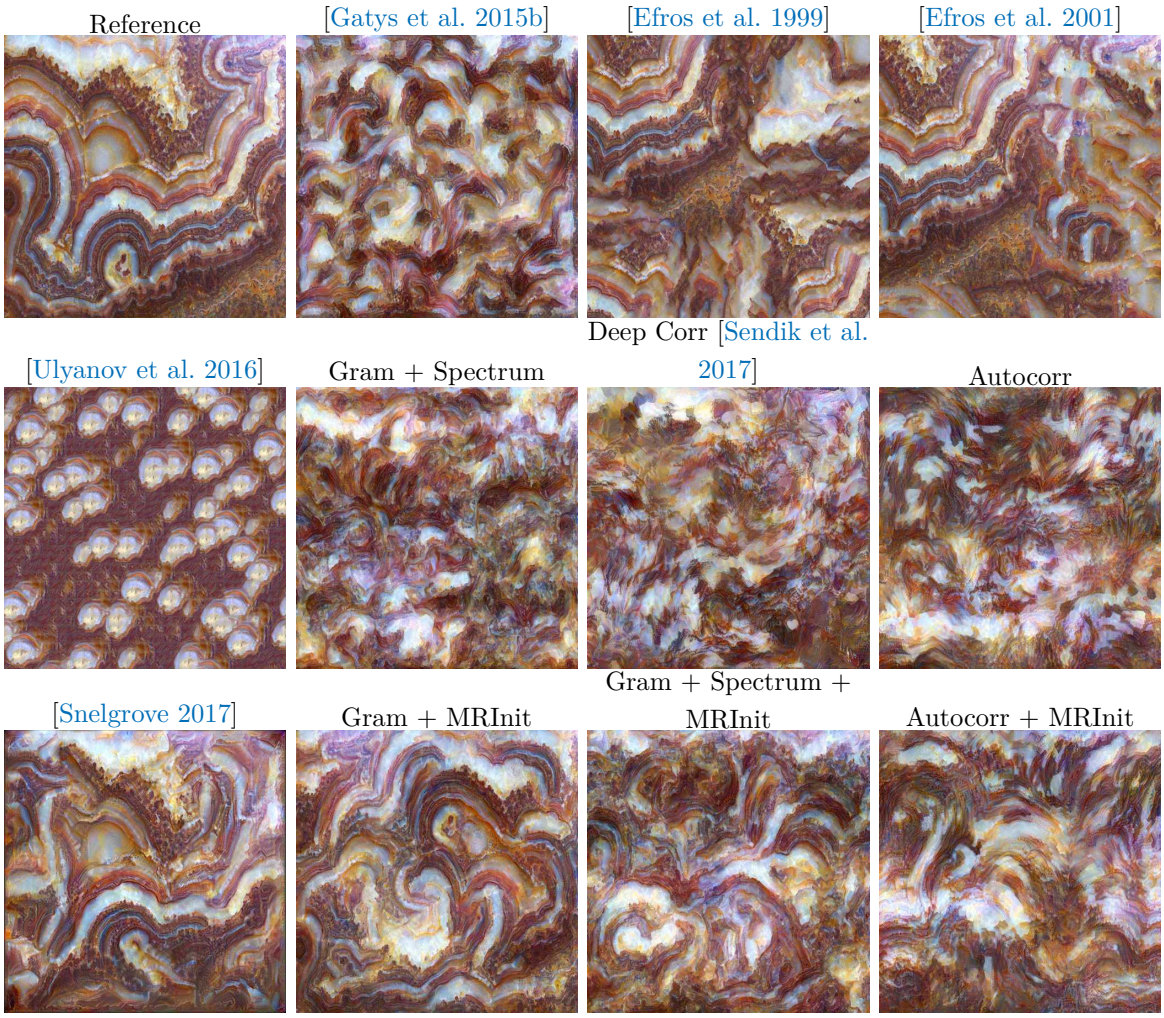


Figure A.8: Synthesis results using different methods for a given reference of size 1048×1048.

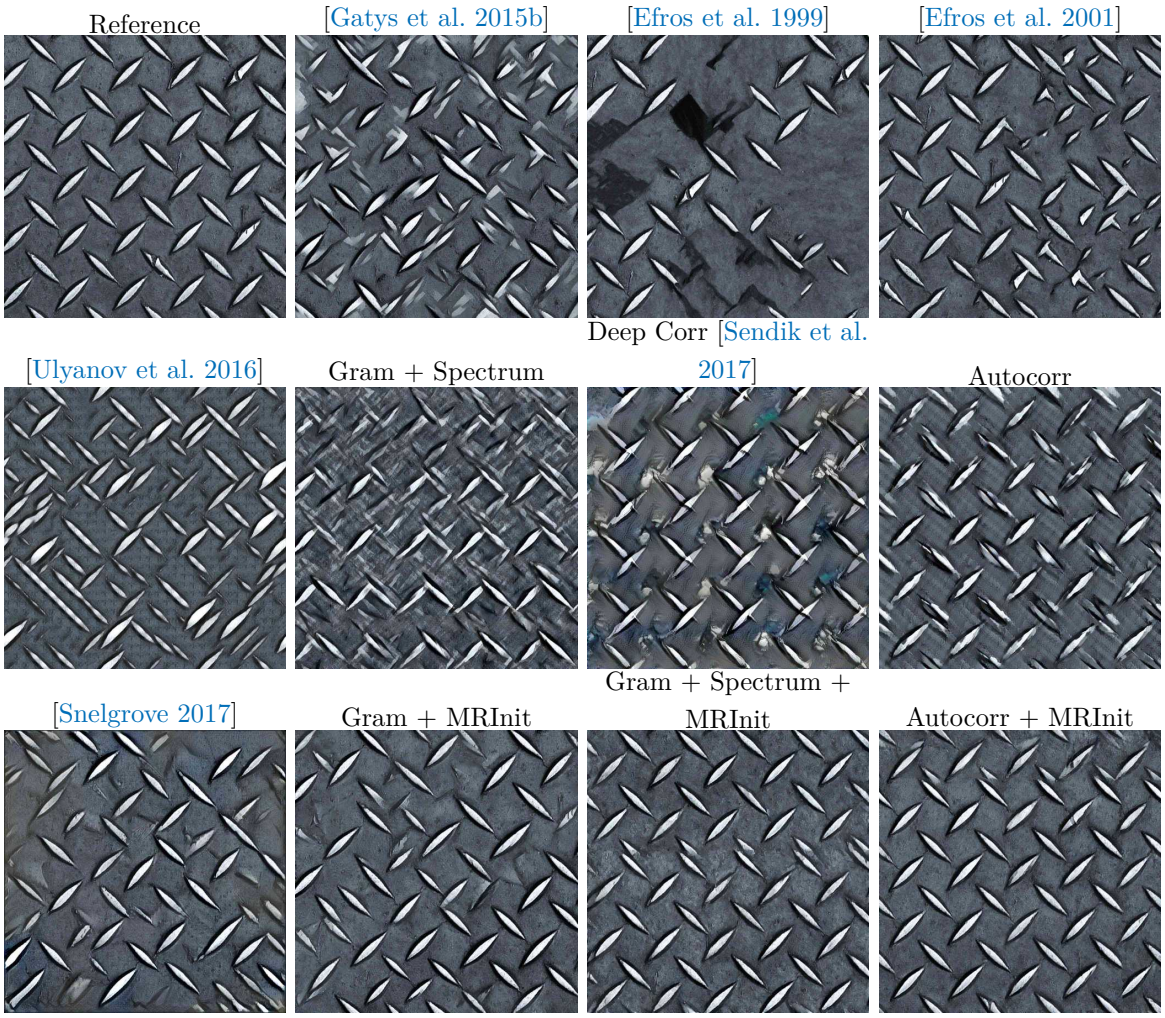


Figure A.9: Synthesis results using different methods for a given reference of size 1048×1048.

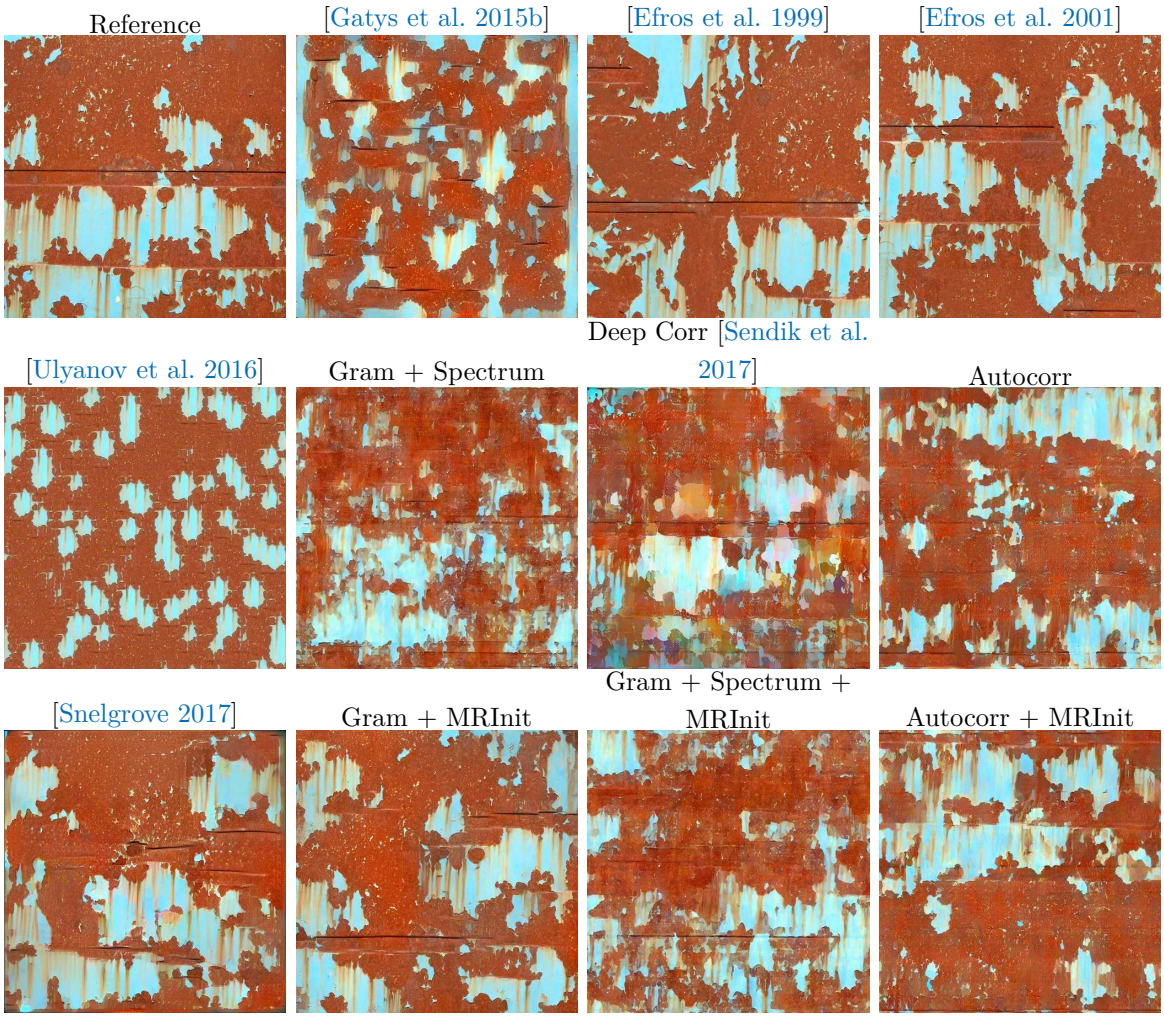


Figure A.10: Synthesis results using different methods for a given reference of size 1048×1048.

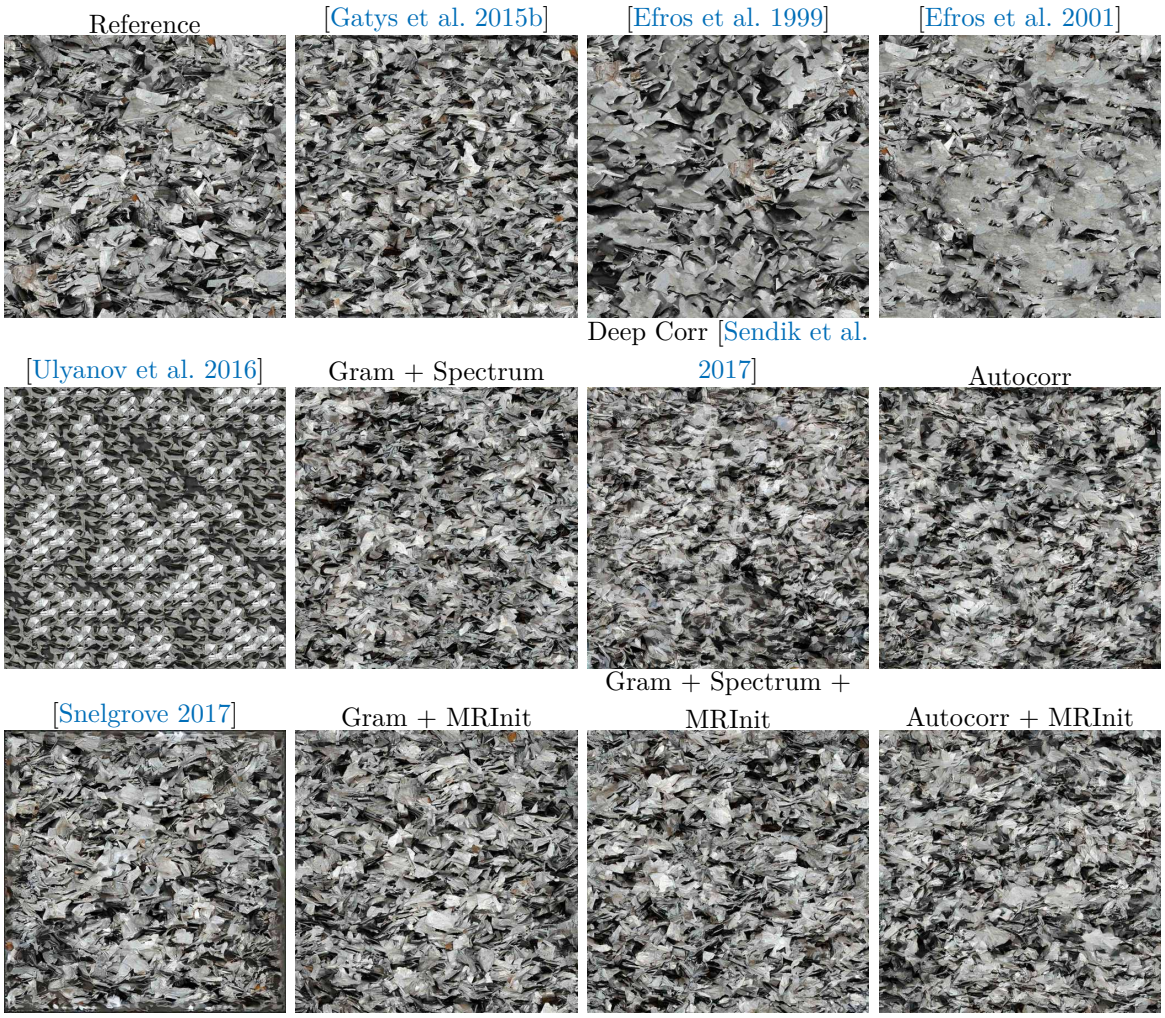


Figure A.11: Synthesis results using different methods for a given reference of size 1048×1048.

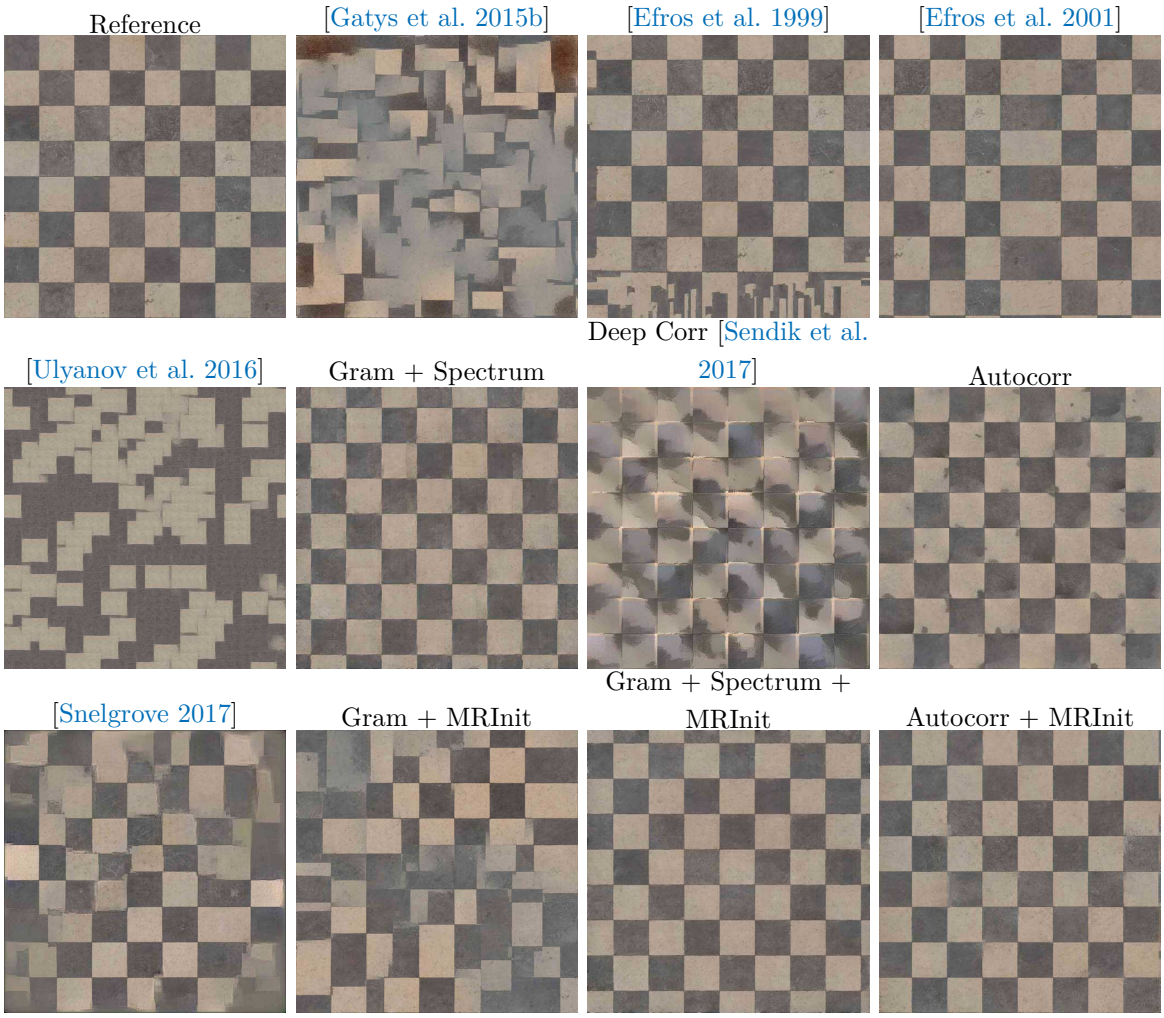


Figure A.12: Synthesis results using different methods for a given reference of size 1048×1048.

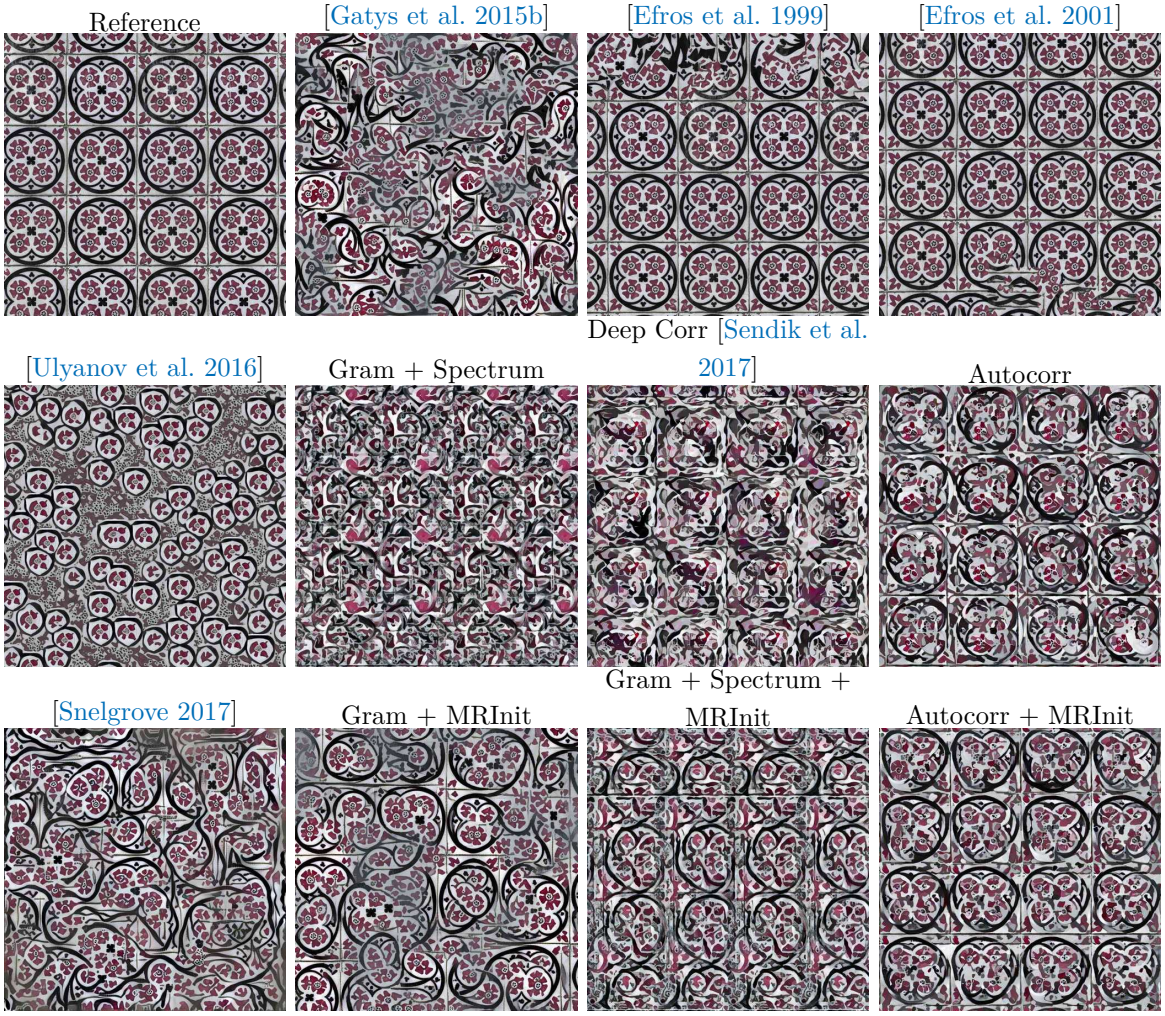


Figure A.13: Synthesis results using different methods for a given reference of size 1048×1048.

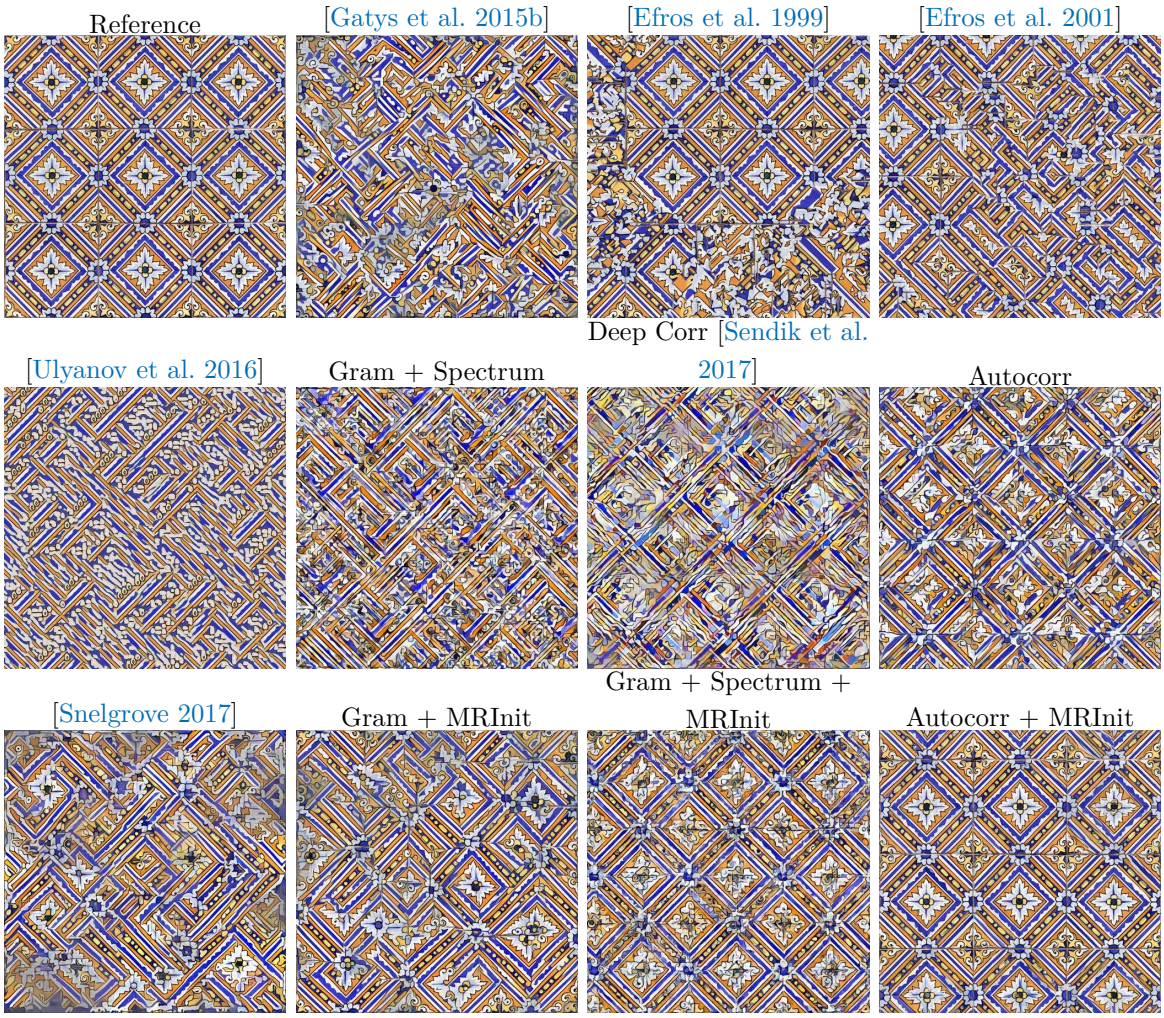


Figure A.14: Synthesis results using different methods for a given reference of size 1048×1048.

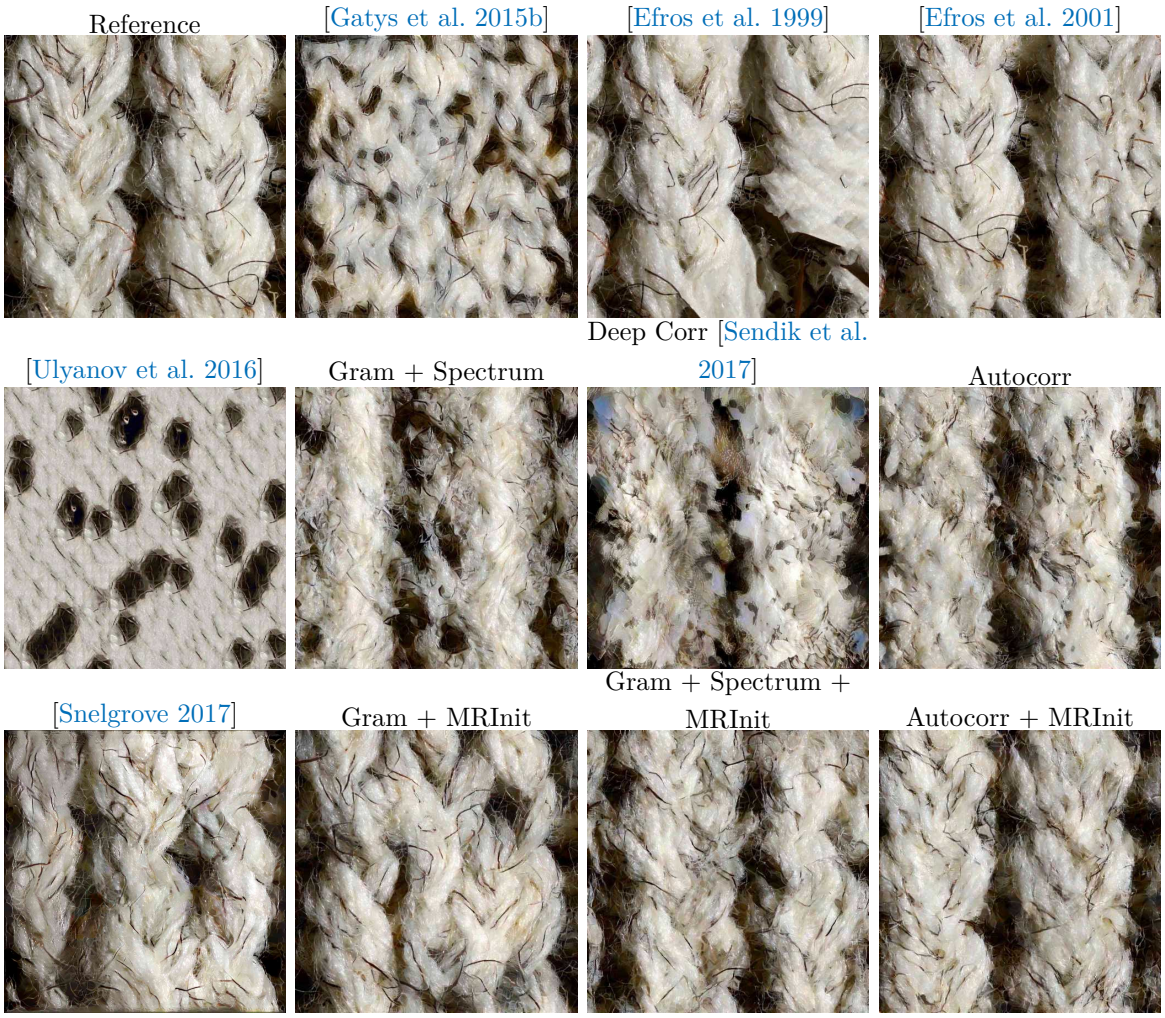


Figure A.15: Synthesis results using different methods for a given reference of size 1048×1048.

Figure A.16: Synthesis results using different methods for a given reference of size 1048×1048.

A.2 Additional Displacement Maps

In Figures A.17 to A.35, one can see the displacement for 19 of the reference images mentioned before, the latter can be seen in Figure 3.7.

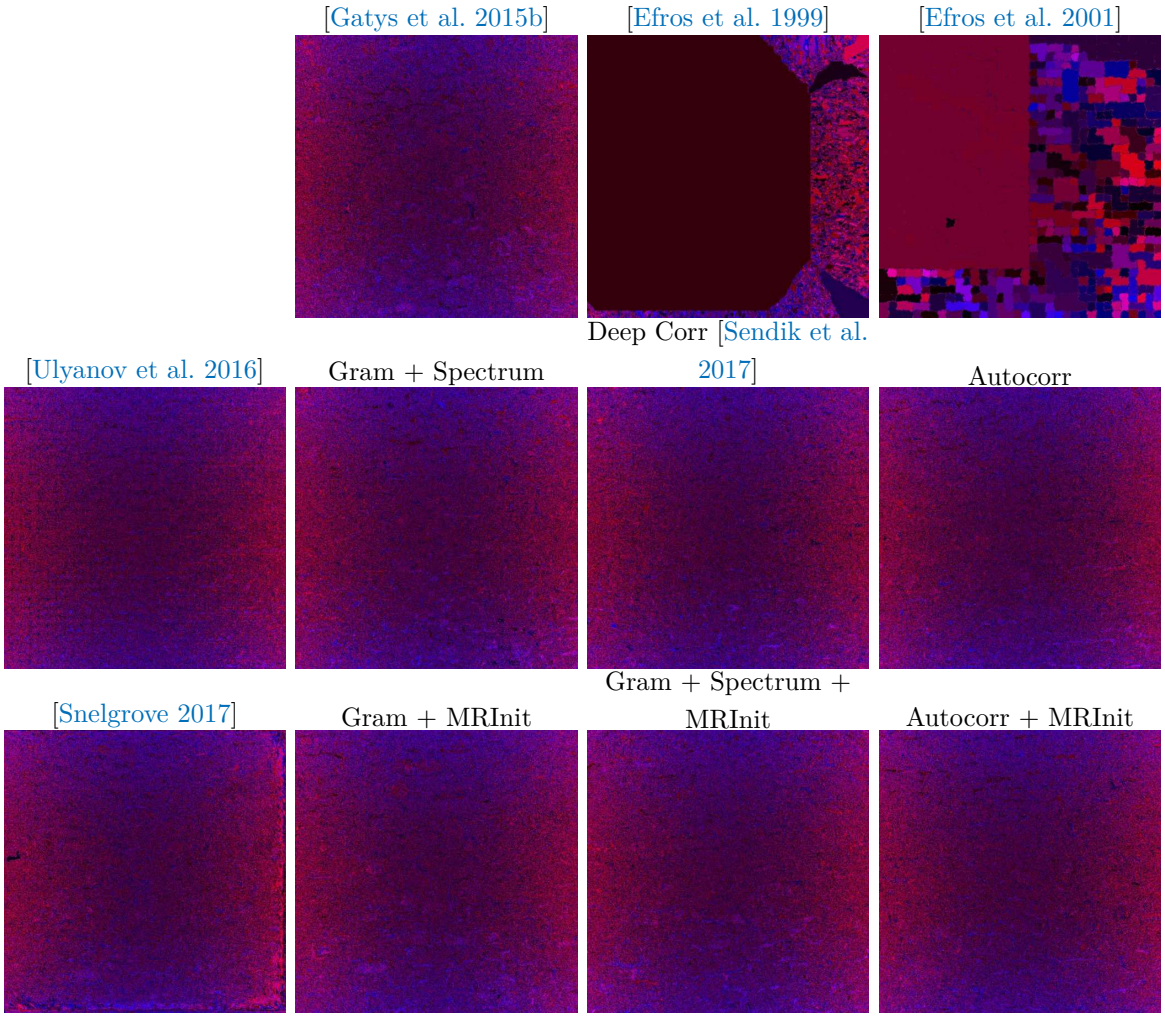


Figure A.17: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.1.

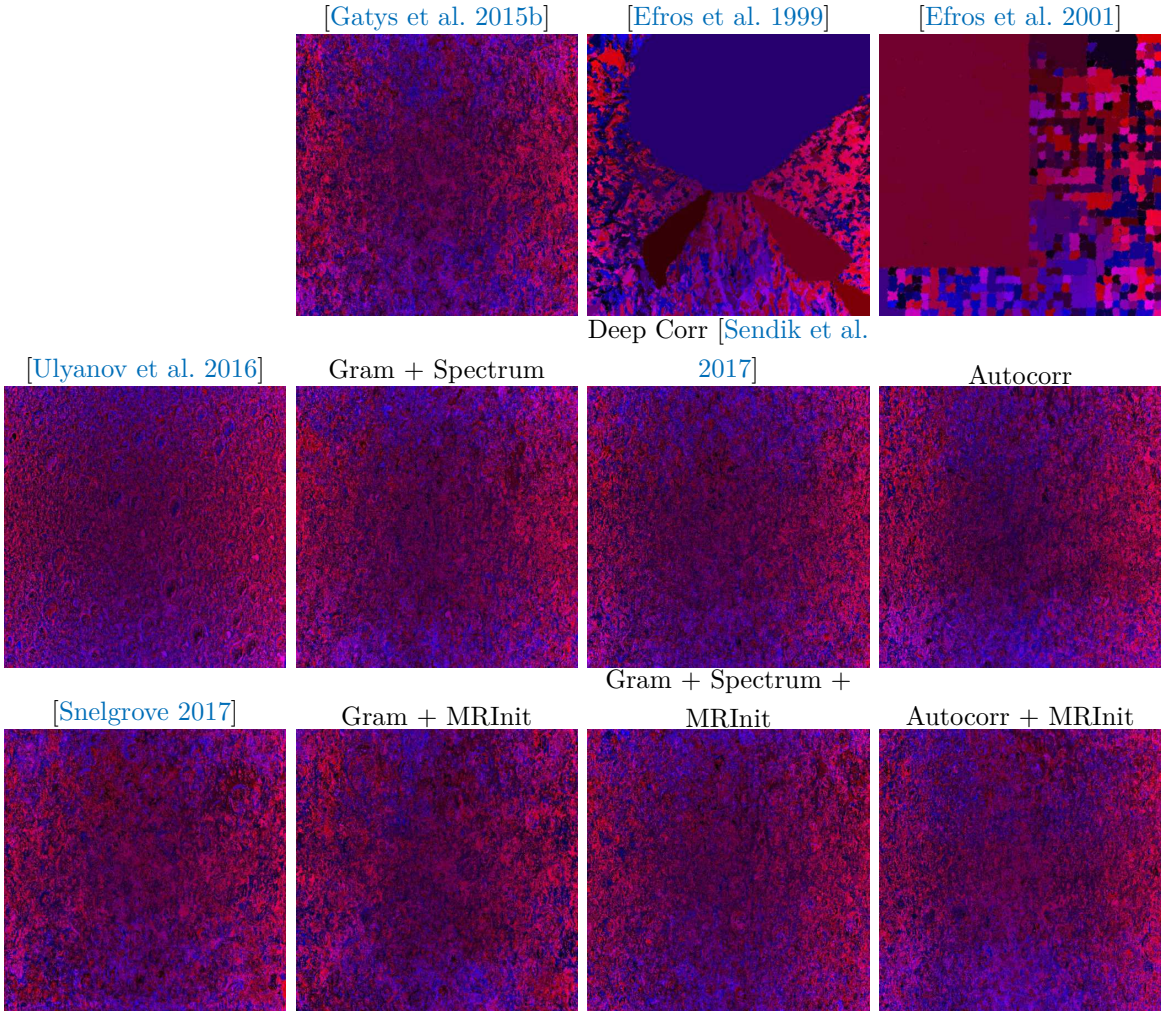


Figure A.18: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.2.

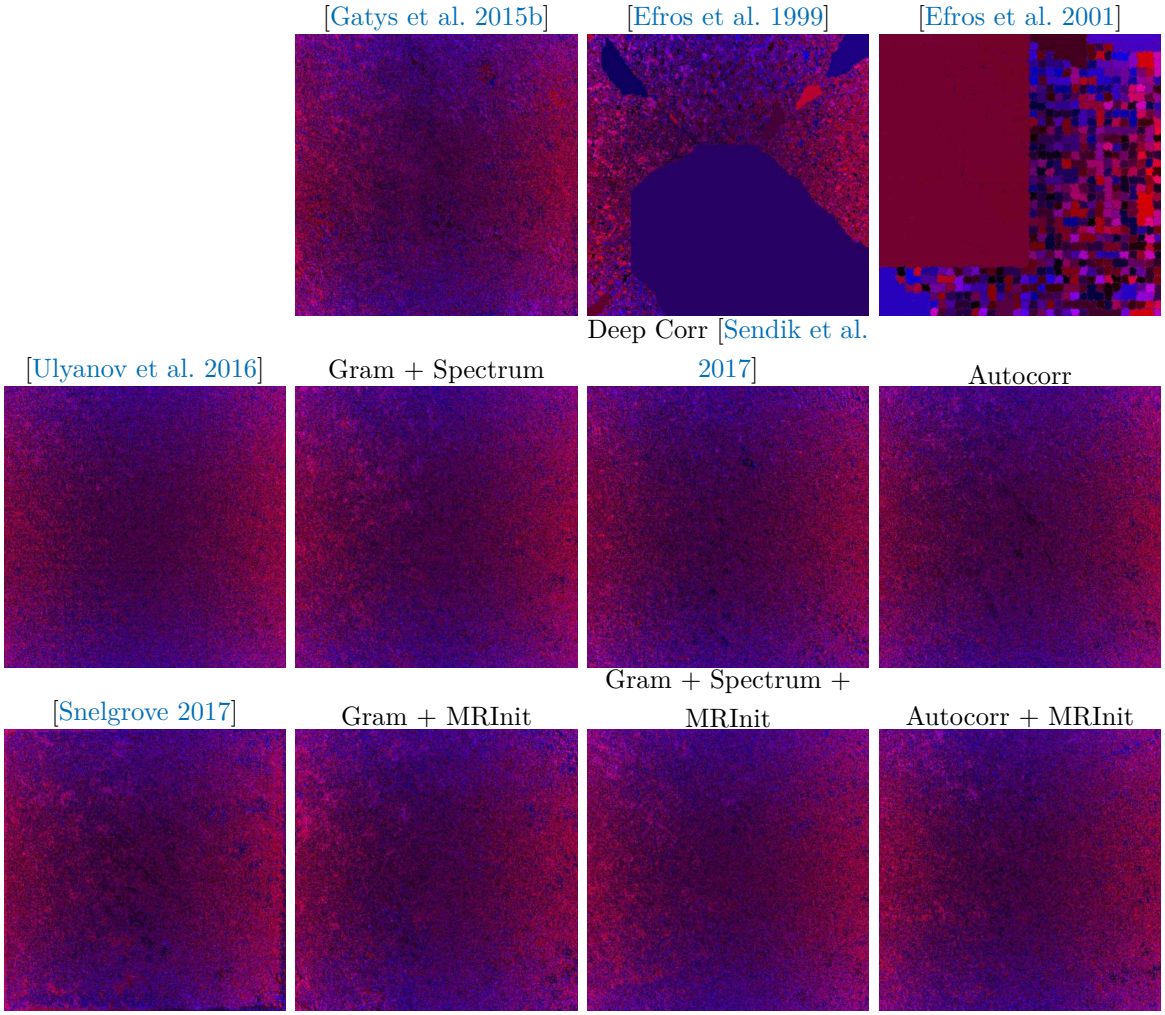


Figure A.19: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.3.

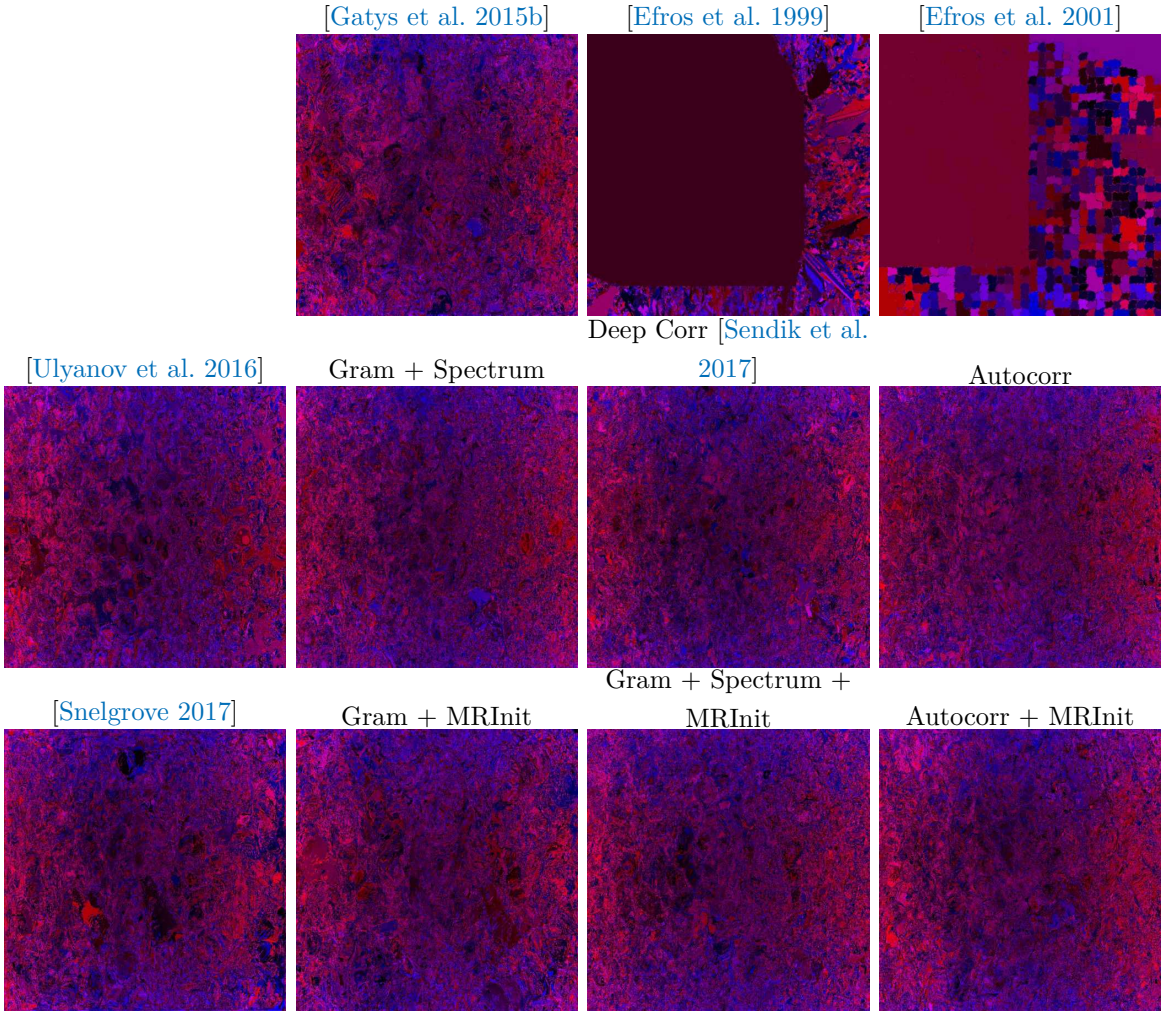


Figure A.20: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.4.

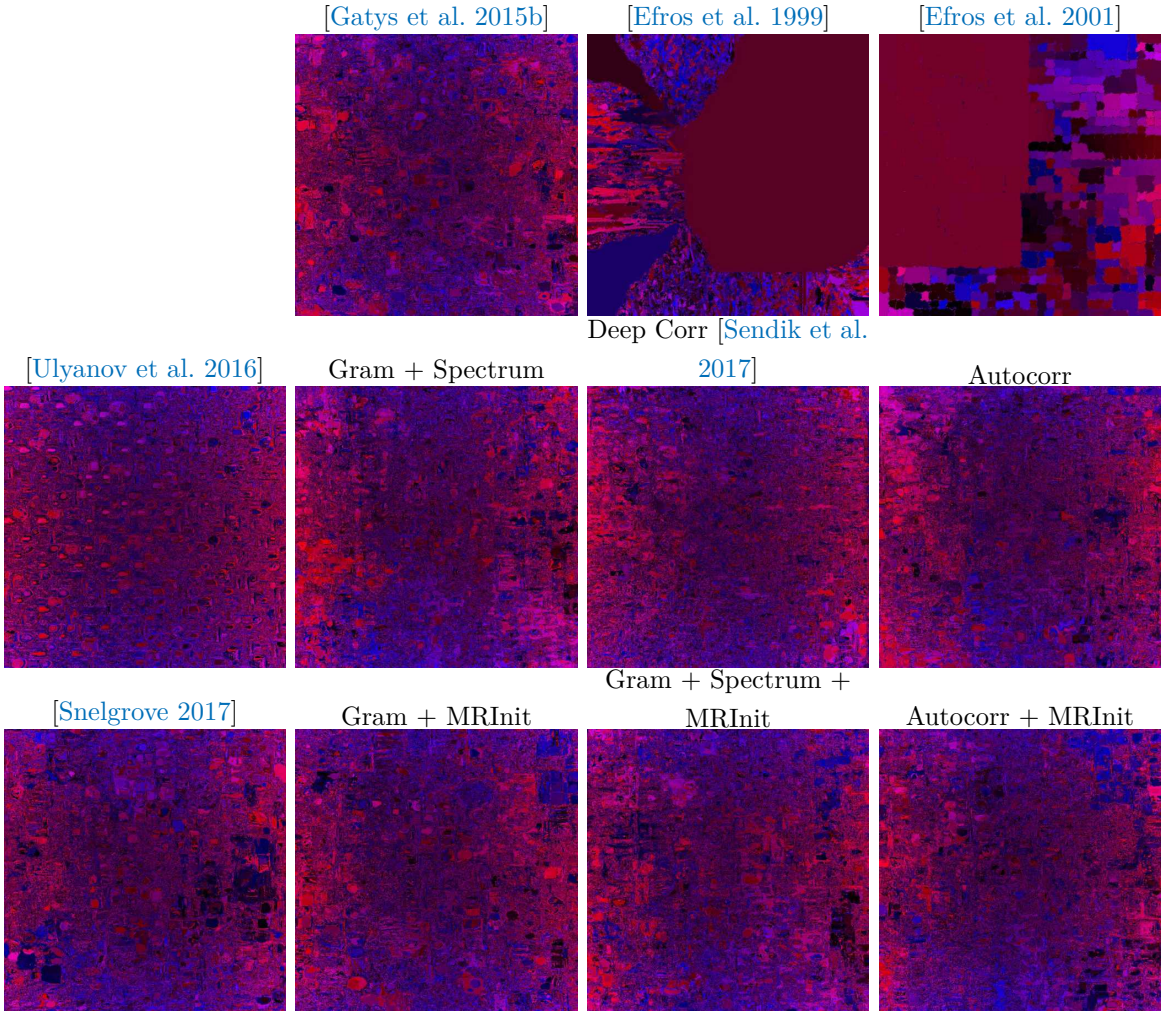


Figure A.21: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.5.

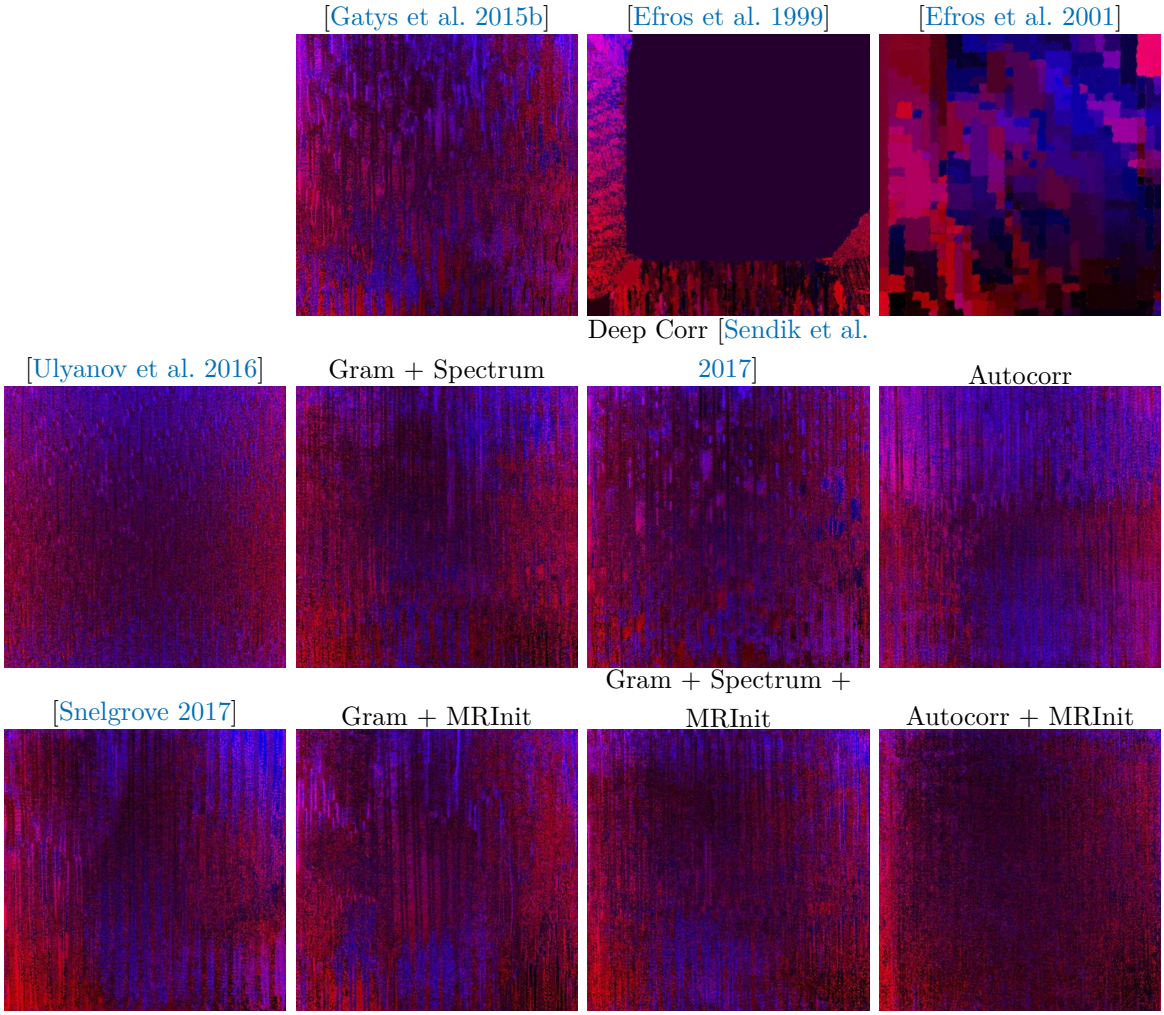


Figure A.22: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure 3.3.

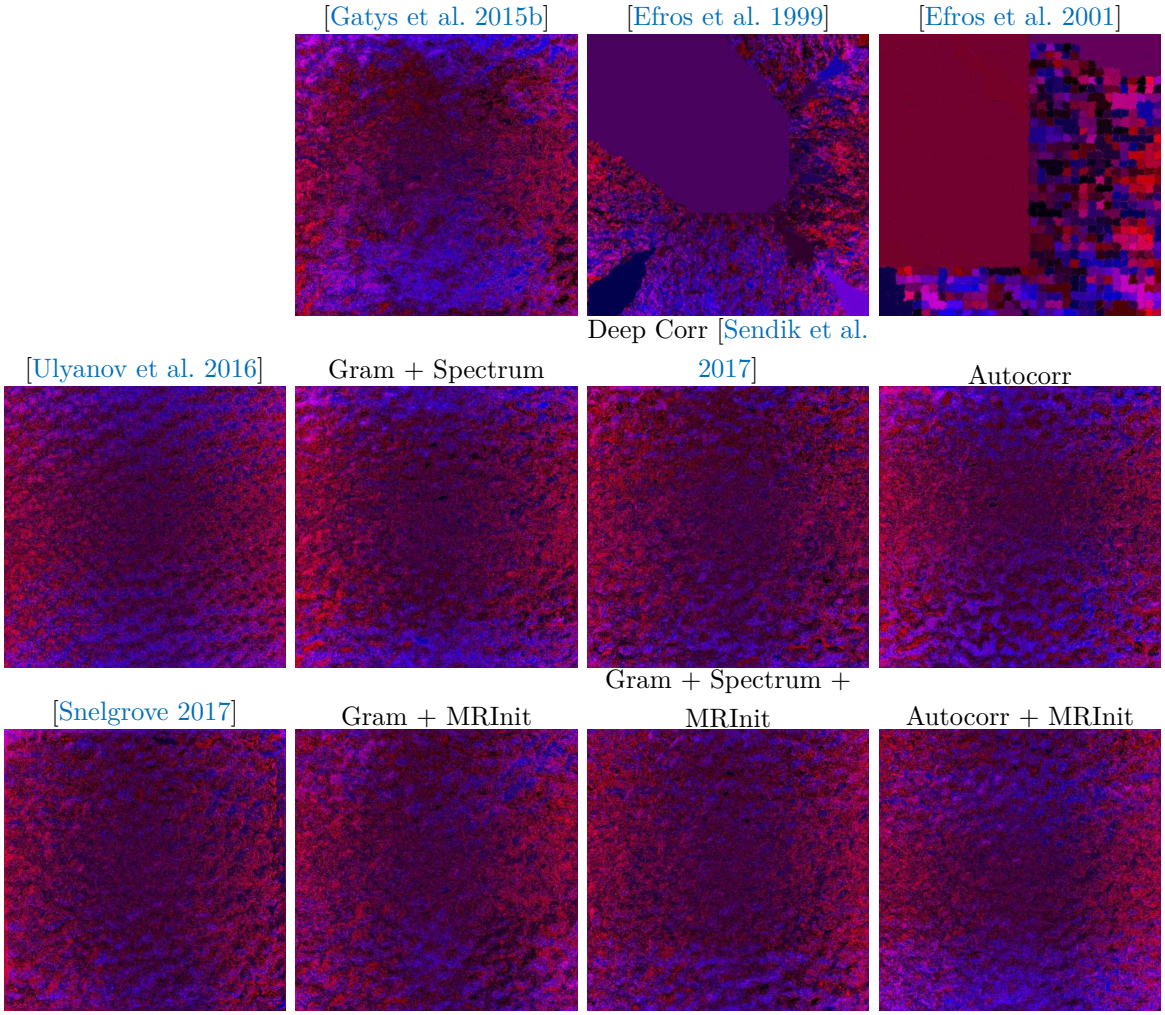


Figure A.23: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.6.

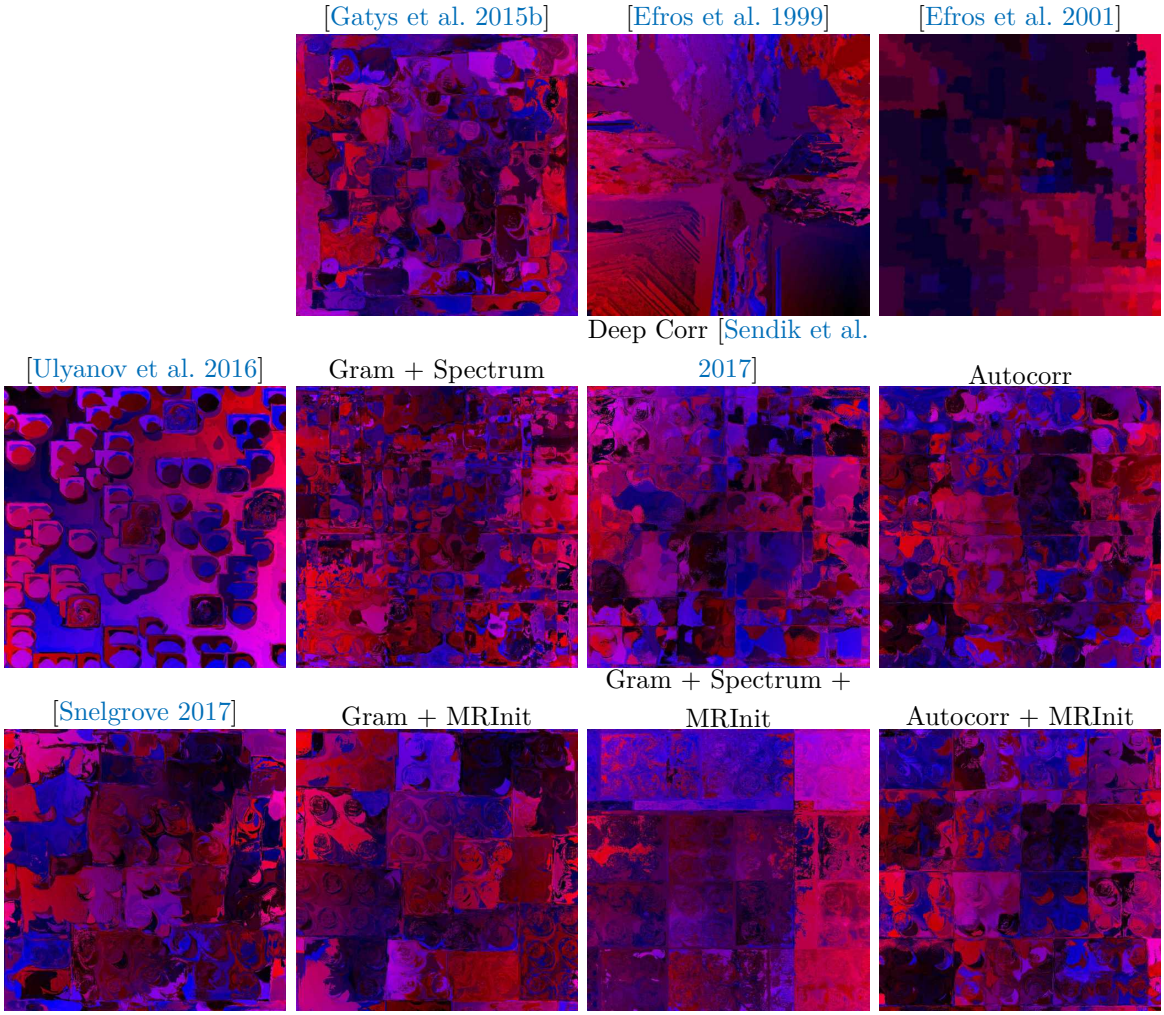


Figure A.24: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.7.

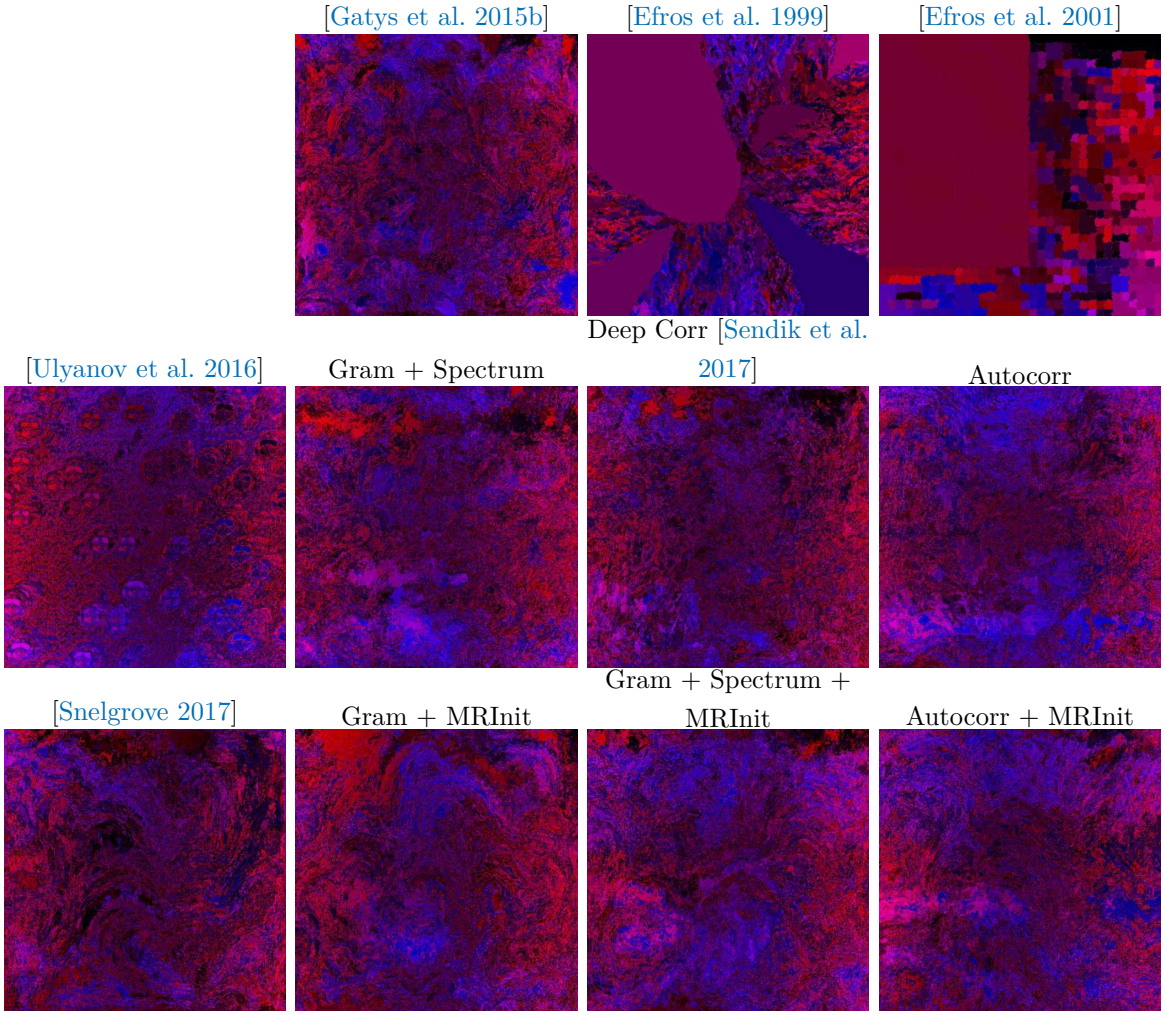


Figure A.25: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.8.

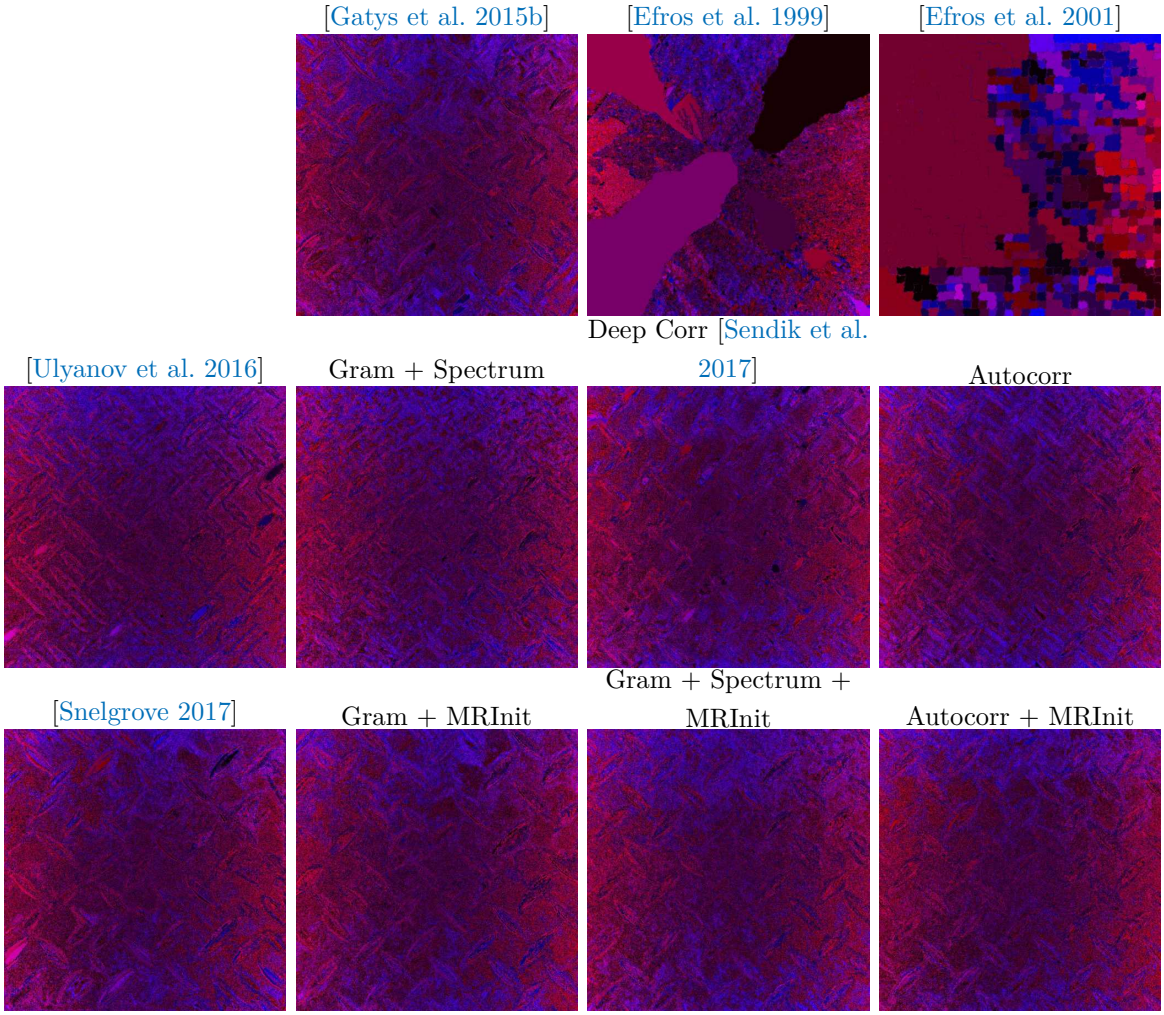


Figure A.26: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.9.

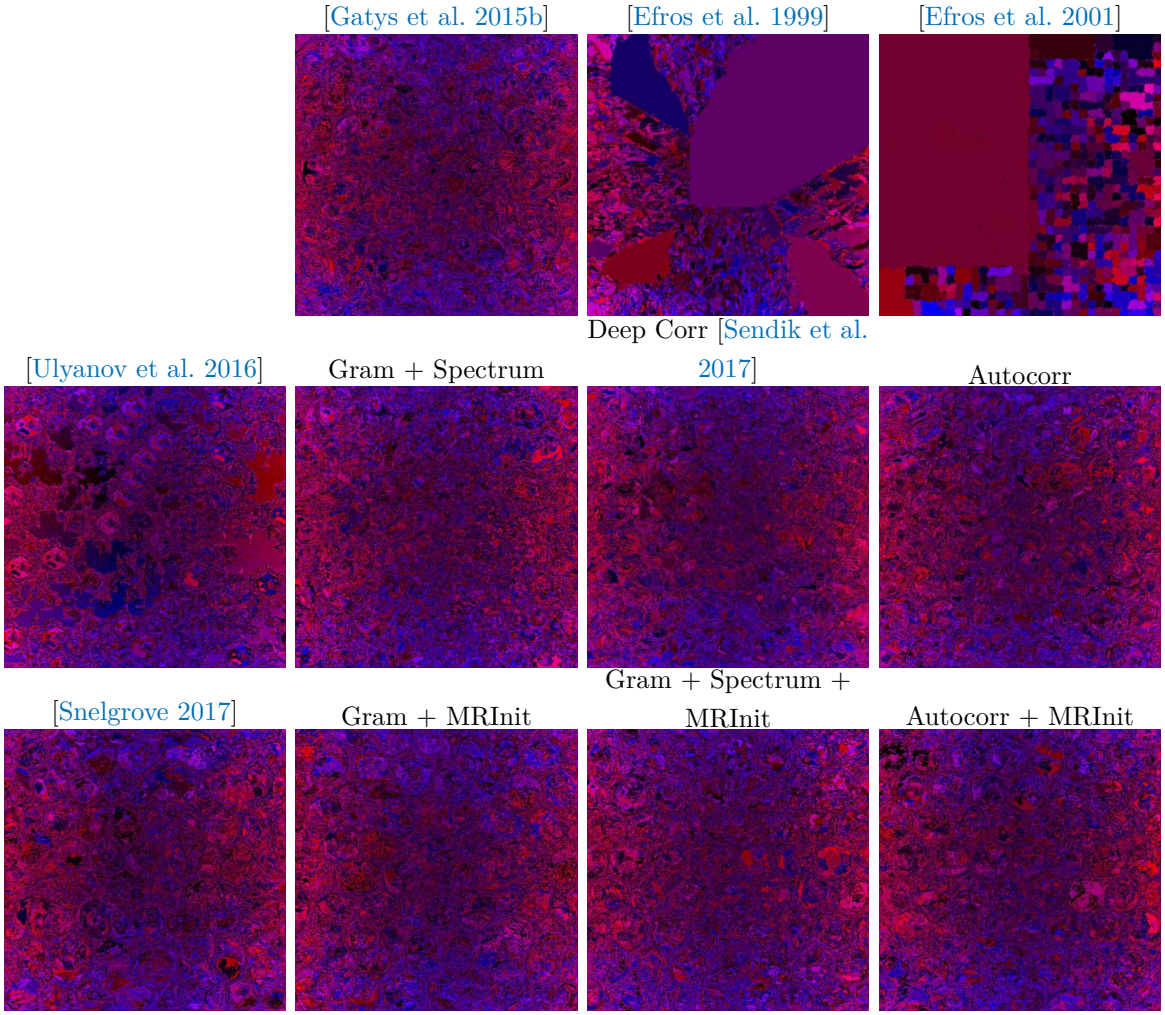


Figure A.27: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure 3.4.

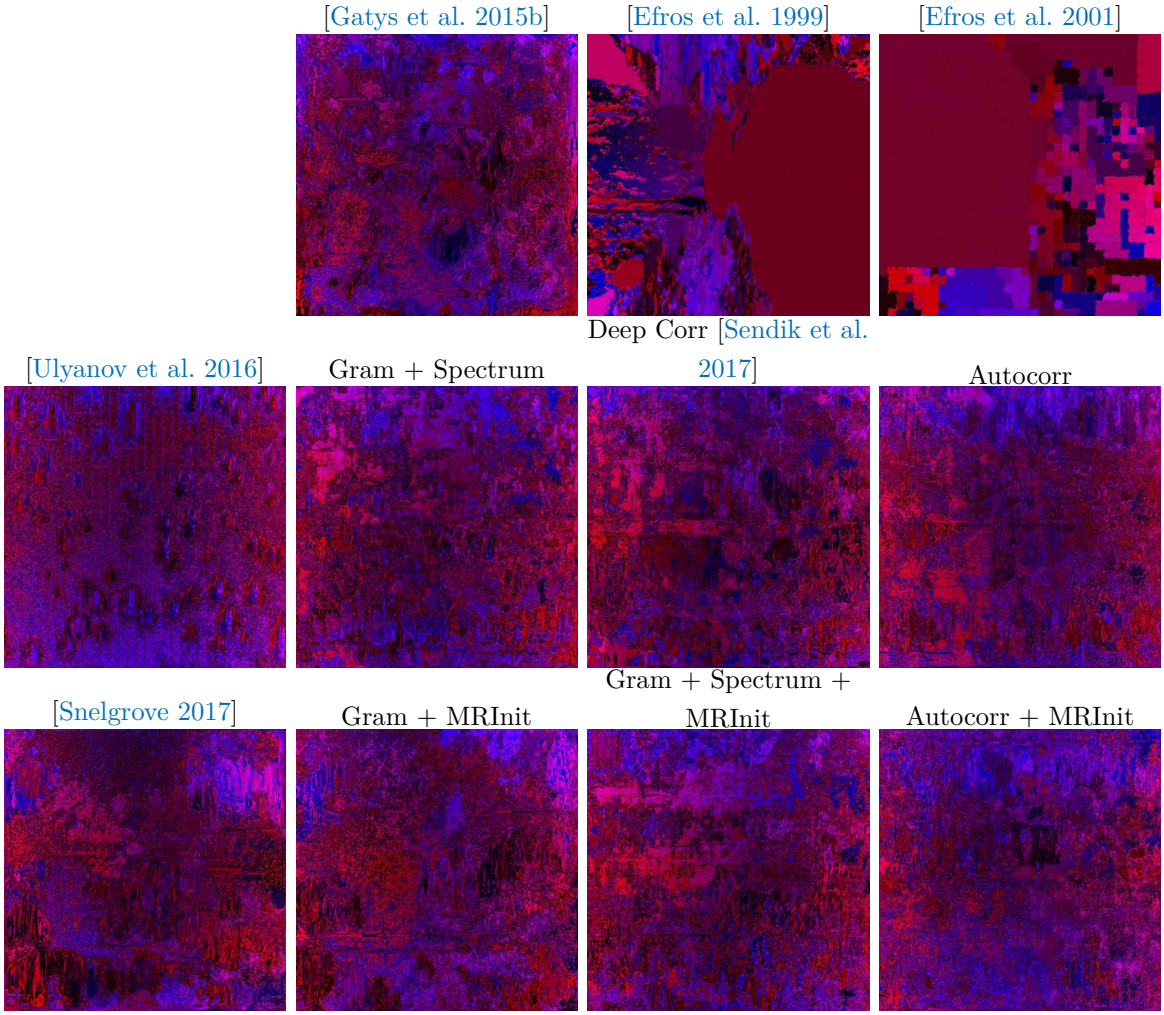


Figure A.28: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.10.

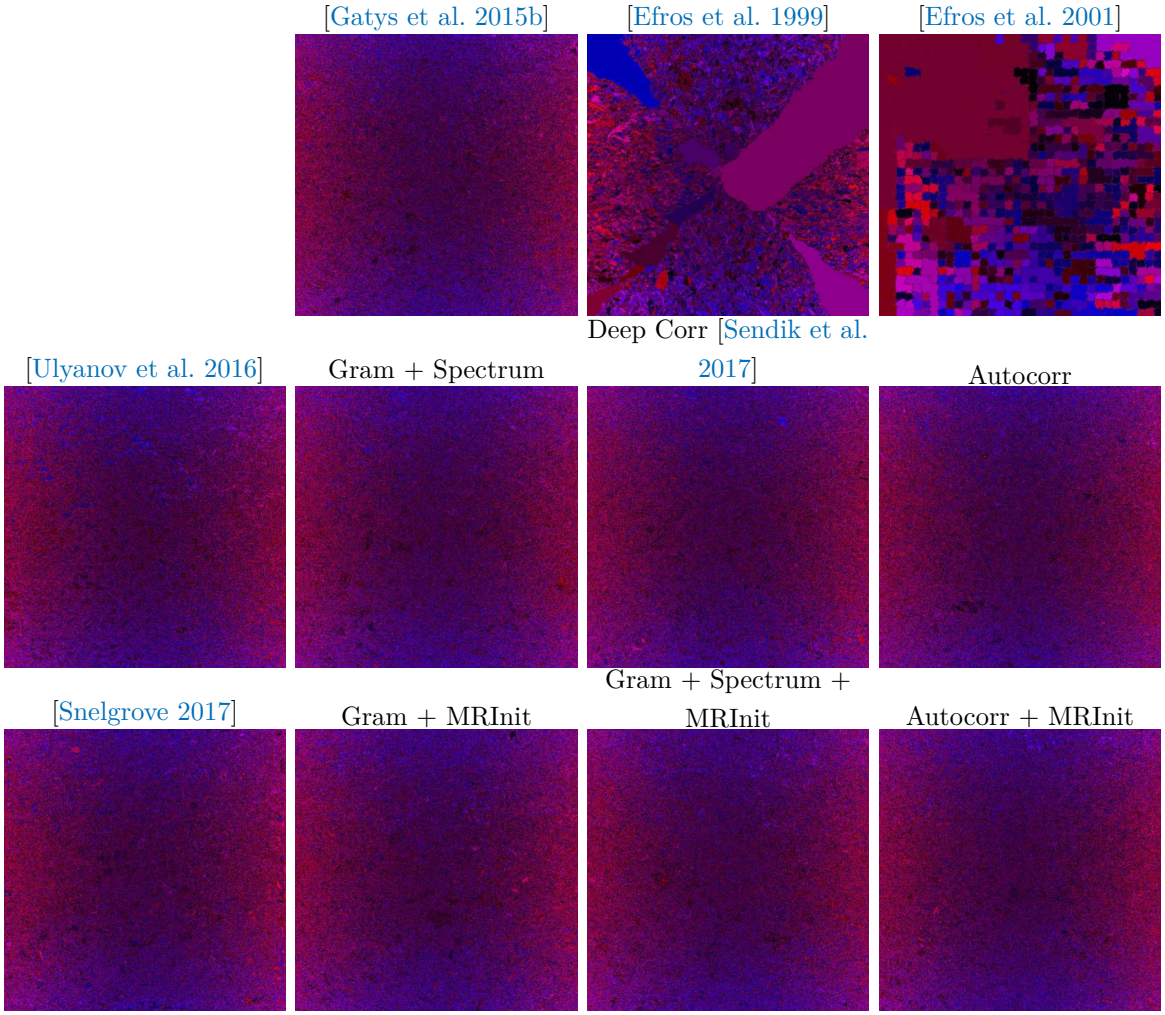


Figure A.29: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.11.

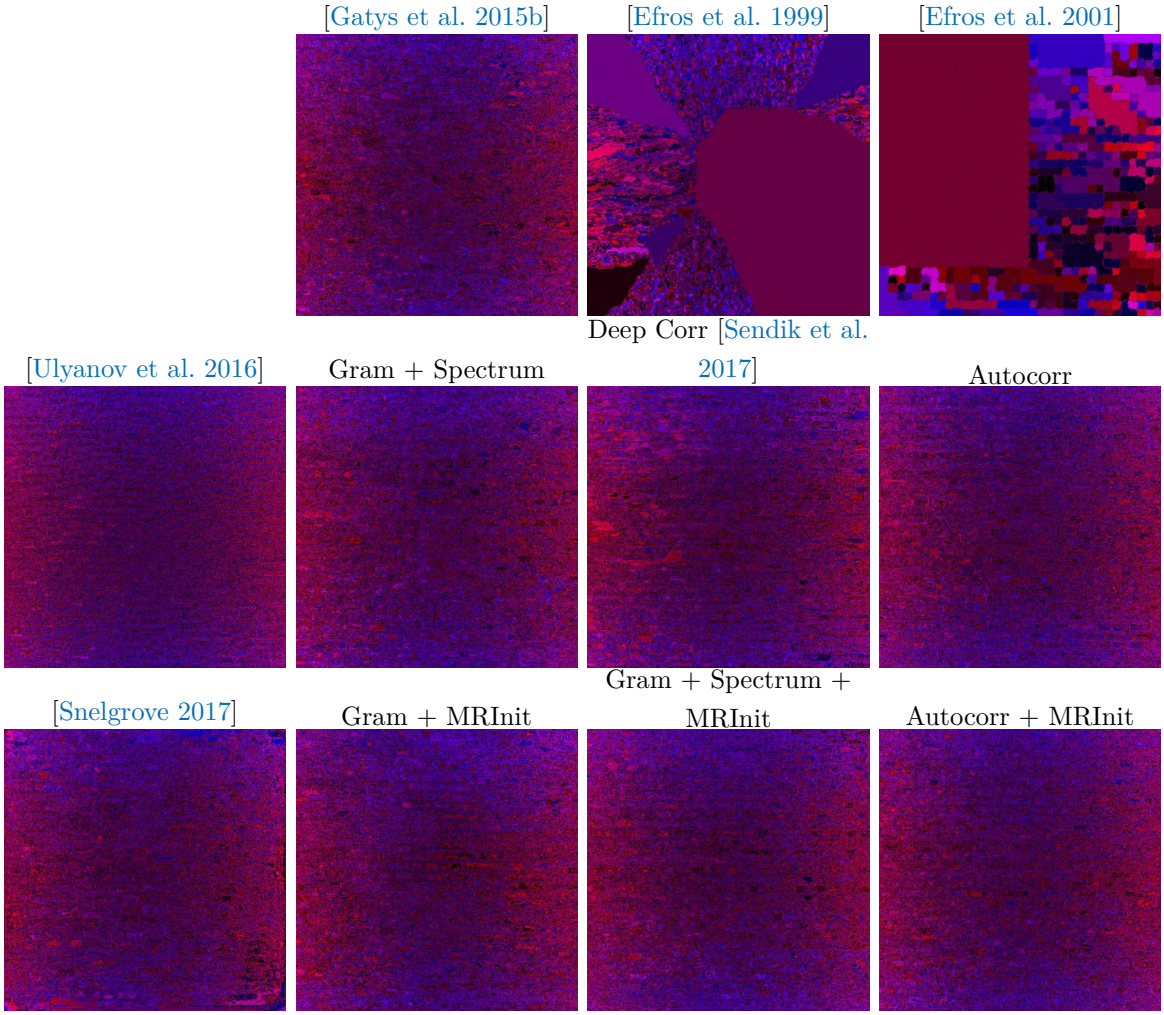


Figure A.30: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure 3.5.

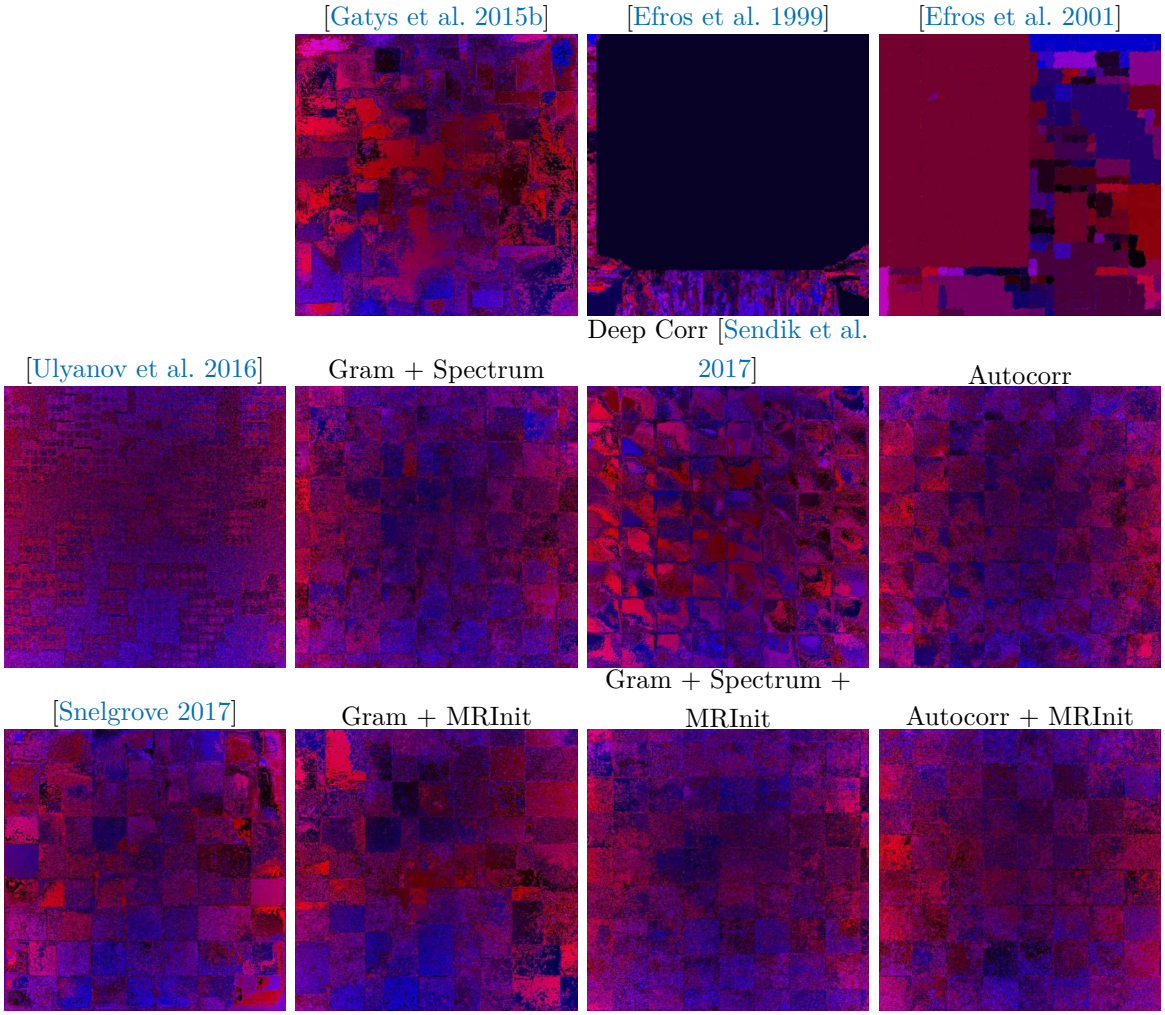


Figure A.31: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.12.

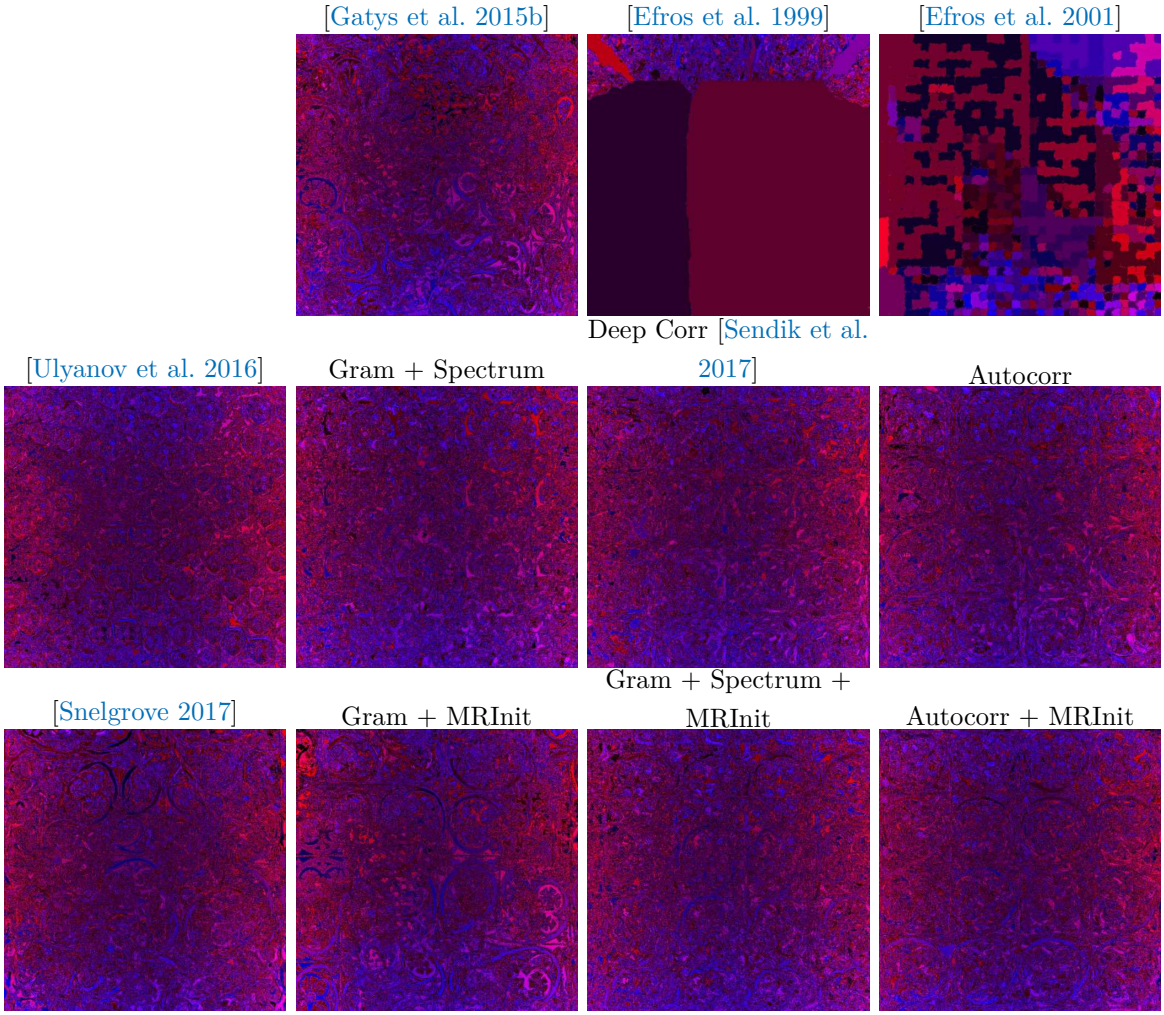


Figure A.32: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.13.

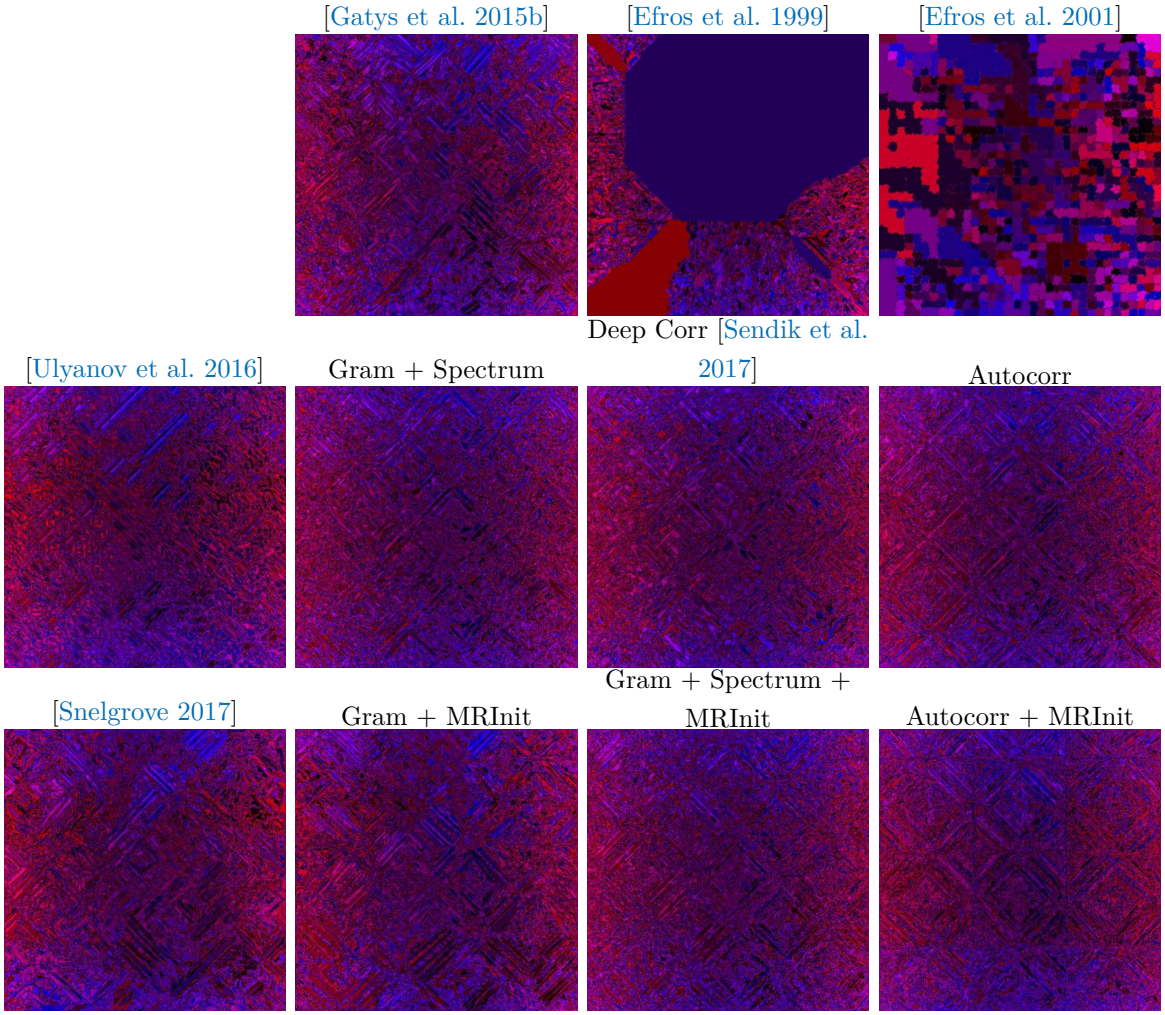


Figure A.33: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.14.

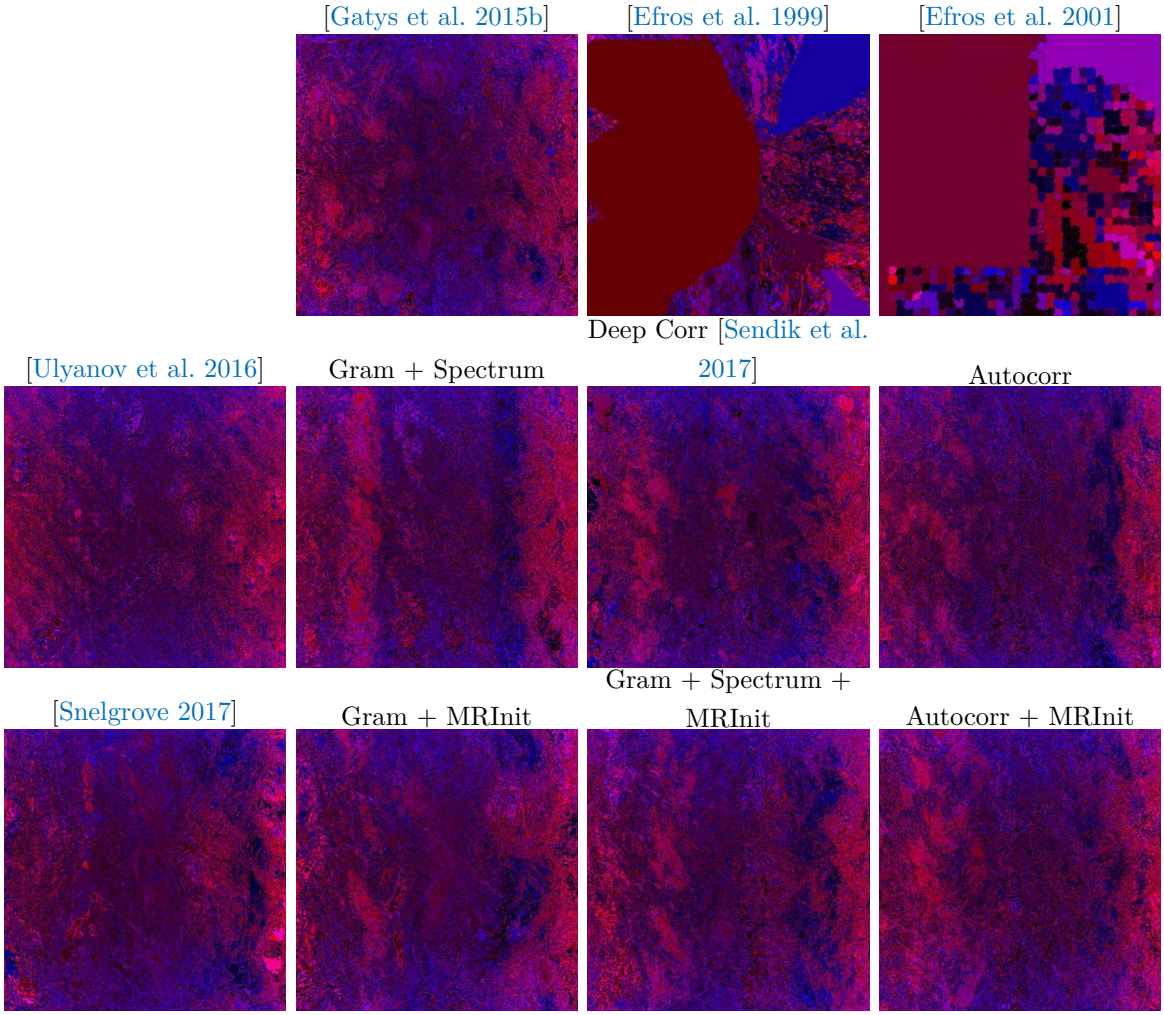


Figure A.34: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.15.

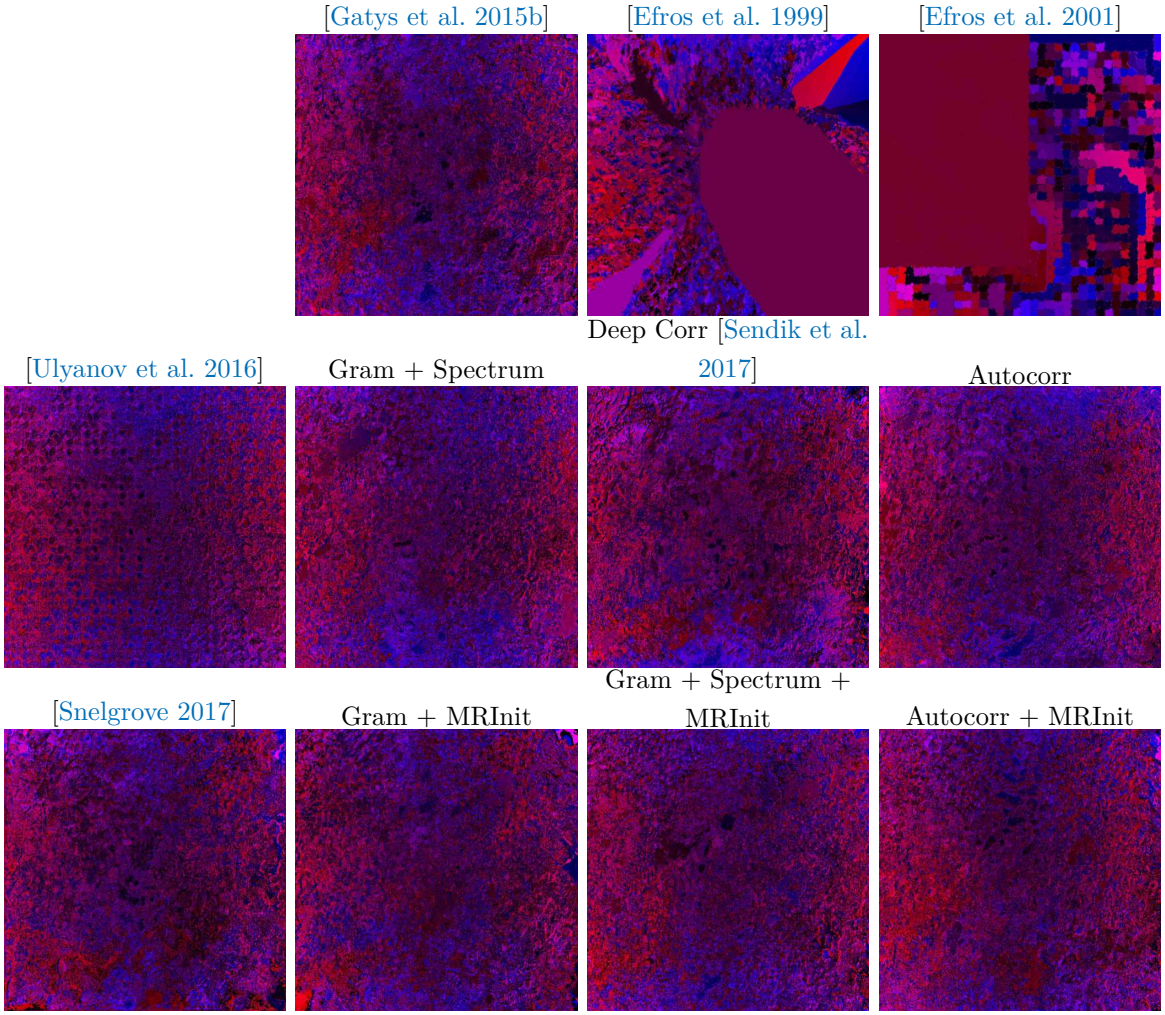


Figure A.35: Displacement map for synthesis results using different methods for a given reference image a constant color area means that the region is a certainly a copy of the reference image. The synthesis can be found figure A.16.

A.3 Screen-shots of the Perceptual Survey

In this section, one can find some screen shots about the online perceptual survey we conducted (Figures A.36 to A.38).

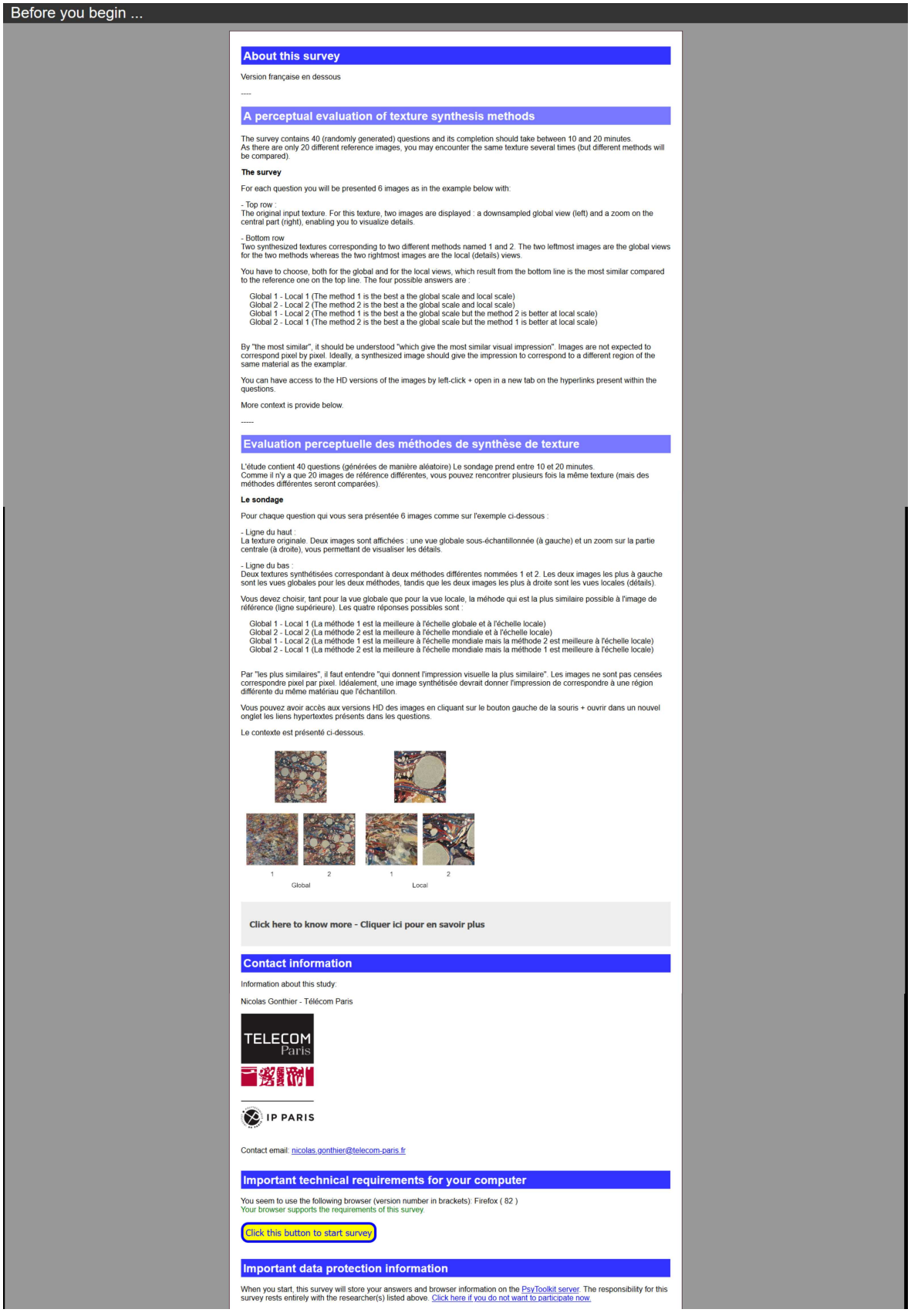


Figure A.36: Front page of the online perceptual survey we conducted.

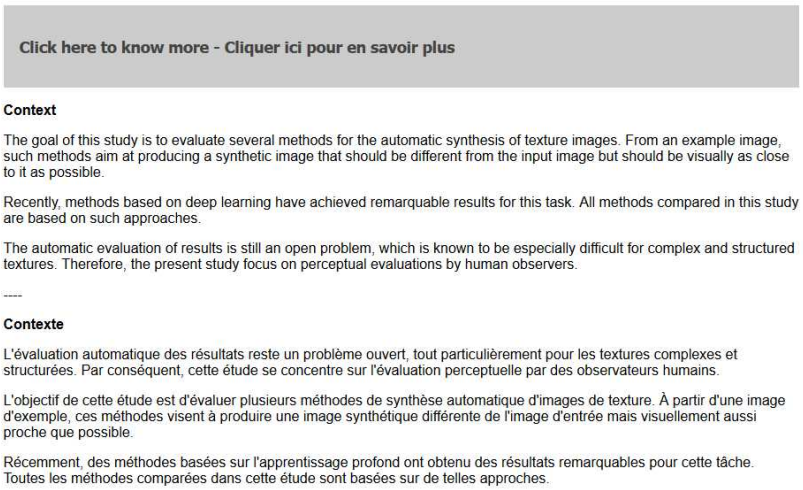


Figure A.37: Front page of the online perceptual survey we conducted.

The text shown to the user on the front page (Figure A.36) is the following one:

"The survey contains 40 (randomly generated) questions and its completion should take between 10 and 20 minutes.

As there are only 20 different reference images, you may encounter the same texture several times (but different methods will be compared).

The survey

For each question you will be presented 6 images as in the example below with:

- Top row:

The original input texture. For this texture, two images are displayed: a downsampled global view (left) and a zoom on the central part (right), enabling you to visualize details.

- Bottom row

Two synthesized textures corresponding to two different methods named 1 and 2. The two leftmost images are the global views for the two methods whereas the two rightmost images are the local (details) views.

You have to choose, both for the global and for the local views, which result from the bottom line is the most similar compared to the reference one on the top line. The four possible answers are:

Global 1 - Local 1 (The method 1 is the best a the global scale and local scale)

Global 2 - Local 2 (The method 2 is the best a the global scale and local scale)

Global 1 - Local 2 (The method 1 is the best a the global scale but the method 2 is better at local scale)

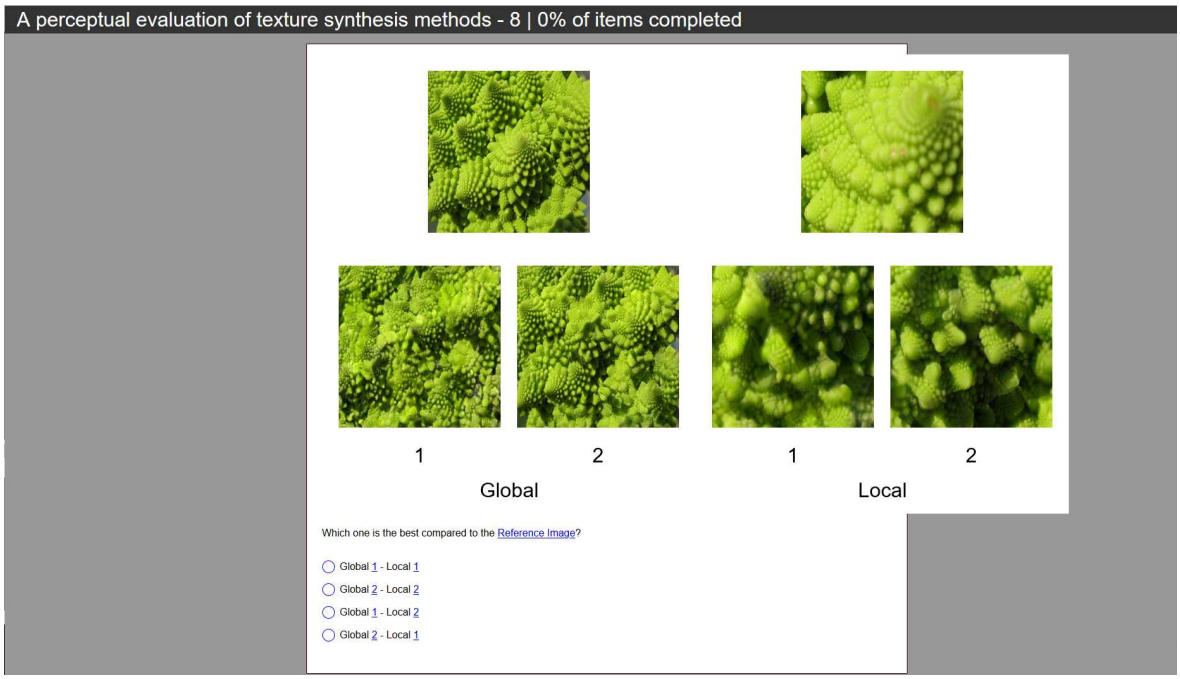
> Global 2 - Local 1 (The method 2 is the best a the global scale but the method 1 is better at local scale)
>
> By "the most similar", it should be understood "which give the most similar visual impression". Images are not expected to correspond pixel by pixel. Ideally, a synthesized image should give the impression to correspond to a different region of the same material as the examplar.
>
> You can have access to the HD versions of the images by left-click + open in a new tab on the hyperlinks present within the questions.
>
> More context is provided below."

The text in the drop-down menu (Figure A.37) is the following one:

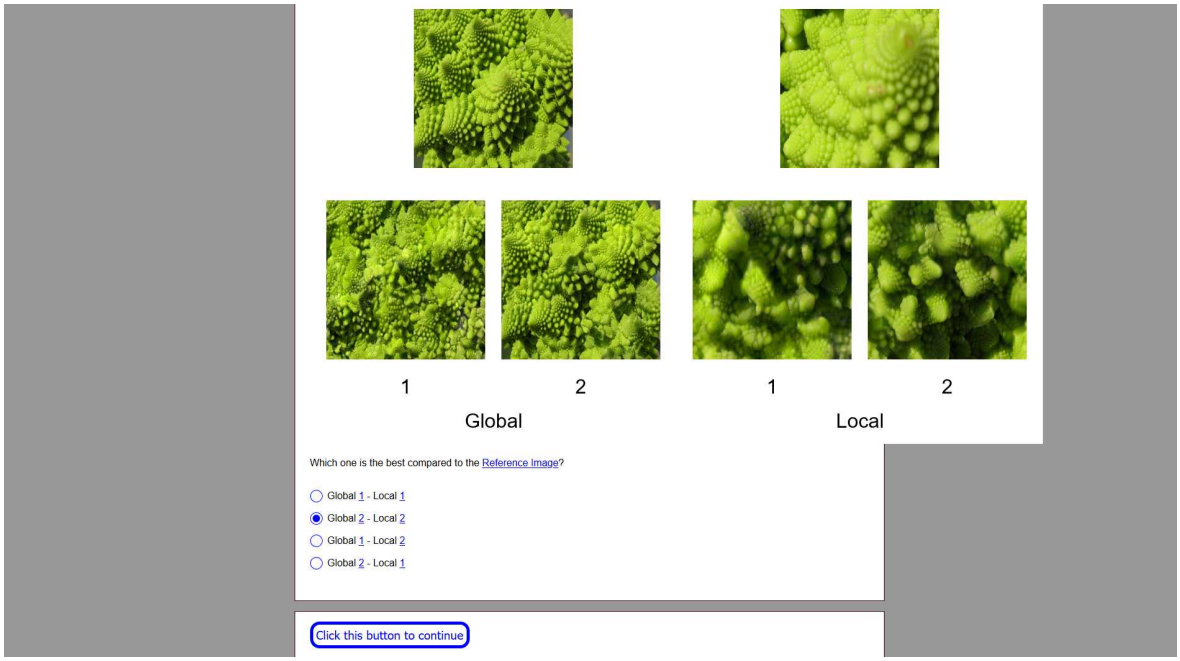
> "**Context** The goal of this study is to evaluate several methods for the automatic synthesis of texture images. From an example image, such methods aim at producing a synthetic image that should be different from the input image but should be visually as close to it as possible. Recently, methods based on deep learning have achieved remarquable results for this task. All methods compared in this study are based on such approaches. The automatic evaluation of results is still an open problem, which is known to be especially difficult for complex and structured textures. Therefore, the present study focus on perceptual evaluations by human observers."

We have to mention that participants were alone, remote[1] and without any time limit for answering the survey. We answered by email at several requests for clarification, but the majority of the participants do not ask us anything. Thus, the only explanation provided to them was the previously mentioned text. The user may access the high-definition version of the images by clicking on the hyperlinks provided within each question.

[1] Due to the COVID-19 pandemic.



(a) A question of the survey.



(b) The same question with an answer selected.

Figure A.38: A question of the online survey about texture synthesis.

A.4 Details about Winning Probabilities for the Perceptual Survey

Notations

- N: number of methods (N=5)
- M: number of duels (M=3170)
- K: number of reference images (K=20)

A.4.1 Strength and Winning Probability Estimation

A model-based approach to address this problem is the following one. Let $\beta_i \in \mathbb{R}$ represent the "strength" of method i (also called performance score), and let the outcome of a duel between methods i and j be determined by $\beta_i - \beta_j$. The **Bradley-Terry model** [Bradley et al. 1952] treats this outcome as an independent Bernoulli random variable with distribution Bernoulli(p_{ij}), where the log-odds corresponding to the probability p_{ij} that method i beats method j is modeled as:

$$\log \frac{p_{ij}}{1 - p_{ij}} = \beta_i - \beta_j, \tag{A.1}$$

equivalently, solving for p_{ij} yields

$$p_{ij} = \frac{e^{\beta_i - \beta_j}}{1 + e^{\beta_i - \beta_j}} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}}. \tag{A.2}$$

The Bradley-Terry model assigns scores to a fixed set of items based on pairwise comparisons of these items, where the log-odds of item i "beating" item j is given by the difference of their scores. Suppose we observe M total duels $(i_1, j_1), \ldots, (i_M, j_M)$ between these N methods, where each (i, j) is a pair of distinct methods in $\{1, \ldots, N\}$. Let $Y_1, \ldots, Y_M \in \{0, 1\}$ be such that $Y_m = 1$ if i_m beat j_m in the m-th duel and $Y_m = 0$ otherwise. The likelihood for the parameters $\theta = (\beta_1, \ldots, \beta_N)$ is then given by

$$\text{lik}\,(\beta_1, \ldots, \beta_N) = \prod_{m=1}^{M} p_{i_m j_m}^{Y_m} \left(1 - p_{i_m j_m}\right)^{1 - Y_m} = \prod_{m=1}^{M} \left(1 - p_{i_m j_m}\right) \left(\frac{p_{i_m j_m}}{1 - p_{i_m j_m}}\right)^{Y_m}, \tag{A.3}$$

where p_{ij} is given as a function of β_i, and β_j by equation (A.1). The log-likelihood is

$$\begin{aligned} l\,(\beta_1, \ldots, \beta_N) &= \sum_{m=1}^{M} Y_m \log \left(\frac{p_{i_m j_m}}{1 - p_{i_m j_m}}\right) + \log\left(1 - p_{i_m j_m}\right) \\ &= \sum_{m=1}^{M} Y_m \left(\beta_{i_m} - \beta_{j_m}\right) - \log\left(1 + e^{\beta_{i_m} - \beta_{j_m}}\right). \end{aligned} \tag{A.4}$$

We then compute the scores (parameters) $\theta = (\beta_1, \ldots, \beta_N)$ of all images by maximizing the likelihood l of equation A.4 with respect to all votes collected from the user study. Since $\hat{\beta}_i - \hat{\beta}_j$ is a linear combination of the coordinates of $\hat{\theta}$, it is approximately normal when $\hat{\theta}$ is approximately multivariate normal. Its mean is $\mathbb{E}\left[\hat{\beta}_i - \hat{\beta}_j\right] \approx \beta_i - \beta_j$, and

its variance is:

$$\begin{aligned}\operatorname{Var}\left[\hat{\beta}_i - \hat{\beta}_j\right] &= \operatorname{Cov}\left[\hat{\beta}_i - \hat{\beta}_j, \hat{\beta}_i - \hat{\beta}_j\right] \\ &= \operatorname{Var}\left[\hat{\beta}_i\right] + \operatorname{Var}\left[\hat{\beta}_j\right] - 2\operatorname{Cov}\left[\hat{\beta}_i, \hat{\beta}_j\right] \\ &\approx \left(I_Y^{-1}(\theta)\right)_{ii} + \left(I_Y^{-1}(\theta)\right)_{jj} - 2\left(I_Y^{-1}(\theta)\right)_{ij}.\end{aligned} \tag{A.5}$$

We may estimate the standard error of $\hat{\beta}_i - \hat{\beta}_j$ by the plug-in estimate:

$$\hat{\text{se}}_{ij} = \sqrt{\left(I_Y^{-1}(\hat{\theta})\right)_{ii} + \left(I_Y^{-1}(\hat{\theta})\right)_{jj} - 2\left(I_Y^{-1}(\hat{\theta})\right)_{ij}}, \tag{A.6}$$

with $I_{\mathbf{Y}}(\theta)$ equal to minus the Hessian matrix of the log-likelihood:

$$I_{\mathbf{Y}}(\theta) = -\nabla^2 l(\theta). \tag{A.7}$$

A $100(1-\alpha)\%$ confidence interval for $\beta_i - \beta_j$, assuming correctness of the Bradley-Terry model, is then given by $\hat{\beta}_i - \hat{\beta}_j \pm z(\alpha/2)\hat{\text{se}}_{ij}$. With $z(\alpha)$ the z-score (or standard score) in the two tails case, we use this confidence interval in Tables 3.1 to 3.3.

Then, the confidence interval for p_{ij} is approximated at the first order as:

$$\hat{\sigma}_{ij} = \frac{e^{\beta_i - \beta_j}}{(1 + e^{\beta_i - \beta_j})^2}\hat{\text{se}}_{ij}. \tag{A.8}$$

This approximation is used in Figures 3.12 to 3.14.

Winning probability We can also calculate the probability that a method i is chosen among all candidates as in [Um et al. 2019]. This *winning probability* is given by the average over j of the probability p_{ij} that a participant chooses the candidate i over j:

$$W_i = \frac{1}{N-1}\sum_{j \neq i}^{N} p_{ij} = \frac{1}{N-1}\sum_{j \neq i}^{N} \frac{e^{\beta_i - \beta_j}}{1 + e^{\beta_i - \beta_j}}. \tag{A.9}$$

In contrast to the pairwise probability p_{ij}, W_i represents the probability that a candidate i was preferred over all other candidates.

We can estimate the standard error of W_i as:

$$\Sigma_i = \frac{1}{N-1}\sqrt{\sum_{j \neq i}^{N} \hat{\sigma}_{ij}^2}, \tag{A.10}$$

with the hypothesis that the p_{ij} are independent.

A.4.2 Considering each Reference Image as an Individual Study

Previously, we gather all the votes per pair of methods without taking into account the reference image considering although we have a different number of duels per reference

image. Another way to do is to consider each reference image k as an independent study and compute the winning probability for each reference image. Then, as in [Um et al. 2019] we can estimate the Winning probability over K studies and the *near-convergence consistency metric*. The winning probability for one method across different studies (reference images) is defined as the mean over the reference image k of the winning probability $W_{i,k}$ for this image:

$$\mu_i = \frac{1}{K} \sum_{k=1}^{K} W_{i,k}. \tag{A.11}$$

The near-convergence consistency metric ε_i is defined as the standard deviation of its winning probabilities:

$$\varepsilon_i = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} (W_{i,k} - \mu_i)^2}, \tag{A.12}$$

where K denotes the number of different studies. This metric indicates how consistently a candidate performs over the others across different evaluation studies.

Another way to approximate the confidence interval is to use the previous method:

$$\epsilon_i = \frac{1}{K} \sum_{k=1}^{K} \Sigma_{i,k}, \tag{A.13}$$

with $\Sigma_{i,k}$ the standard error for method i given the reference image k.

A.4.2.1 Practical Details

Several solutions are possible for Maximum Likelihood Estimation:

- Minorization-maximization algorithm
- The iterative Luce Spectral Ranking algorithm
- Second order gradient ascent

We used the last one. We used the Python package Choix. In order to compute the standard error of the difference $\beta_i - \beta_j$, we used the pseudo-inverse of the Hessian matrix.

A.4.2.2 Results

These winning probabilities μ_i obtained by considering each reference images are displayed with the near-convergence consistency ε_i in Figures A.39 to A.41 and with the standard deviation ϵ_i in Figures A.42 to A.44. These results confirm the other winning probability results (from Figures 3.12 to 3.14), "Gram+Spectrum+MRInit" prevails at global scale for the regular textures whereas "Gram+MRInit" wins in most of the situation. Nevertheless the near-convergence consistency metric is always one order of magnitude bigger than the standard deviation. Further investigation should be necessary to determine which one is the most pertinent metric for uncertainty.

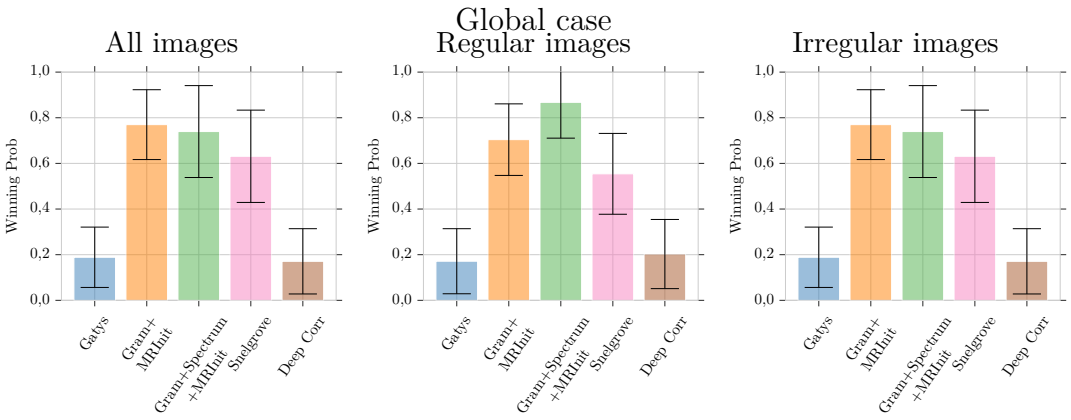


Figure A.39: Winning probabilities μ_i with near-convergence consistency ε_i for the different methods for the global case, with each reference image as an individual study.

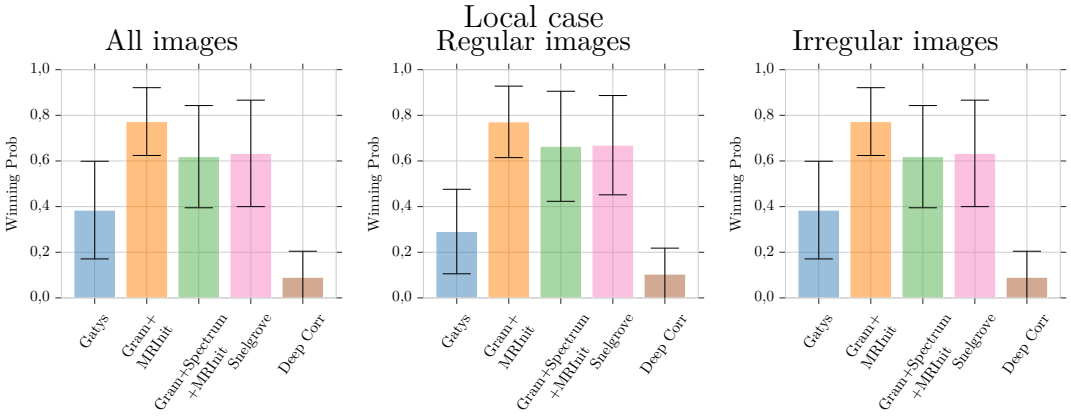


Figure A.40: Winning probabilities μ_i with near-convergence consistency ε_i for the different methods for the local case, with each reference image as an individual study.

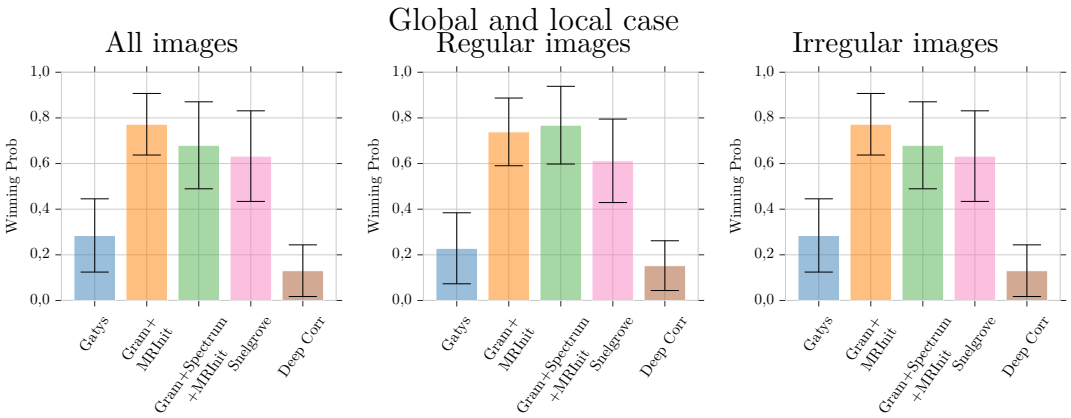


Figure A.41: Winning probabilities μ_i with near-convergence consistency ε_i for the different methods for both global and local case, with each reference image as an individual study.

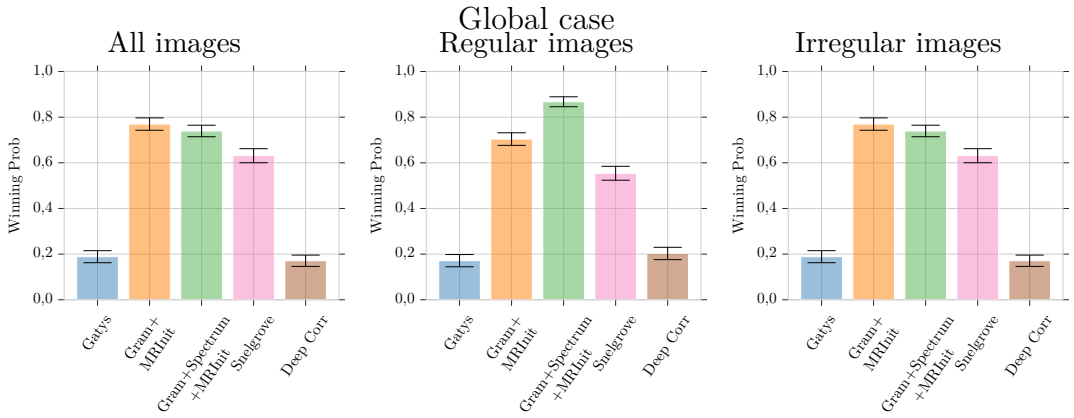


Figure A.42: Winning probabilities μ_i with standard error ϵ_i for the different methods for the global case, with each reference image as an individual study.

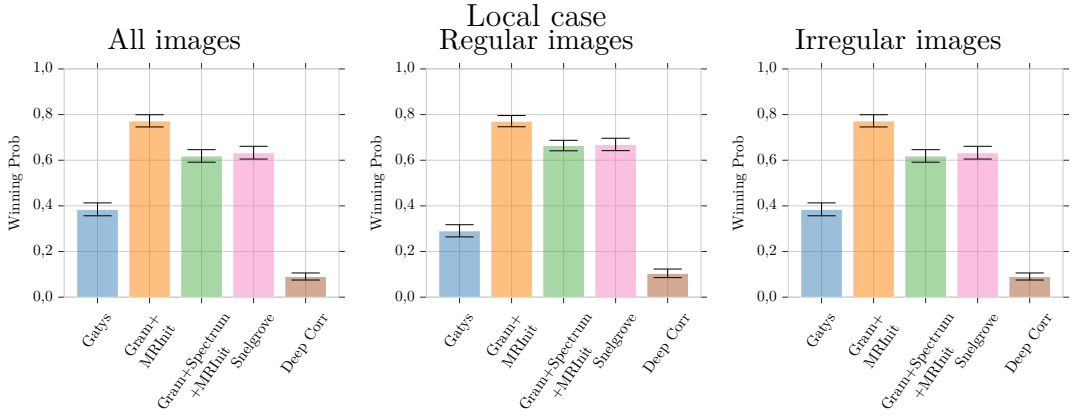


Figure A.43: Winning probabilities μ_i with standard error ϵ_i for the different methods for the local case, with each reference image as an individual study.

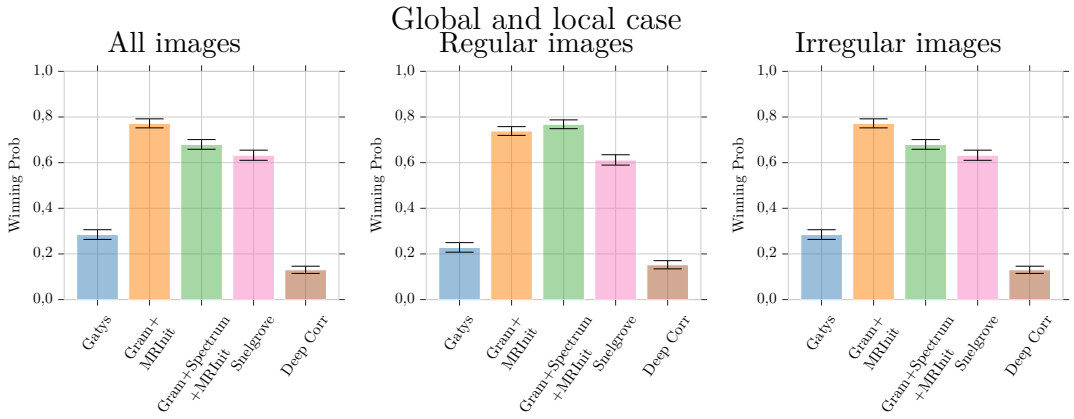


Figure A.44: Winning probabilities μ_i with standard error ϵ_i for the different methods for both global and local case, with each reference image as an individual study.

A.5 Weighting of the Spectrum Constraint

In Figure A.45, we display the result of the synthesis for different values of β, the parameter weighting the Spectrum constraint, using the method "Gram+Spectrum".

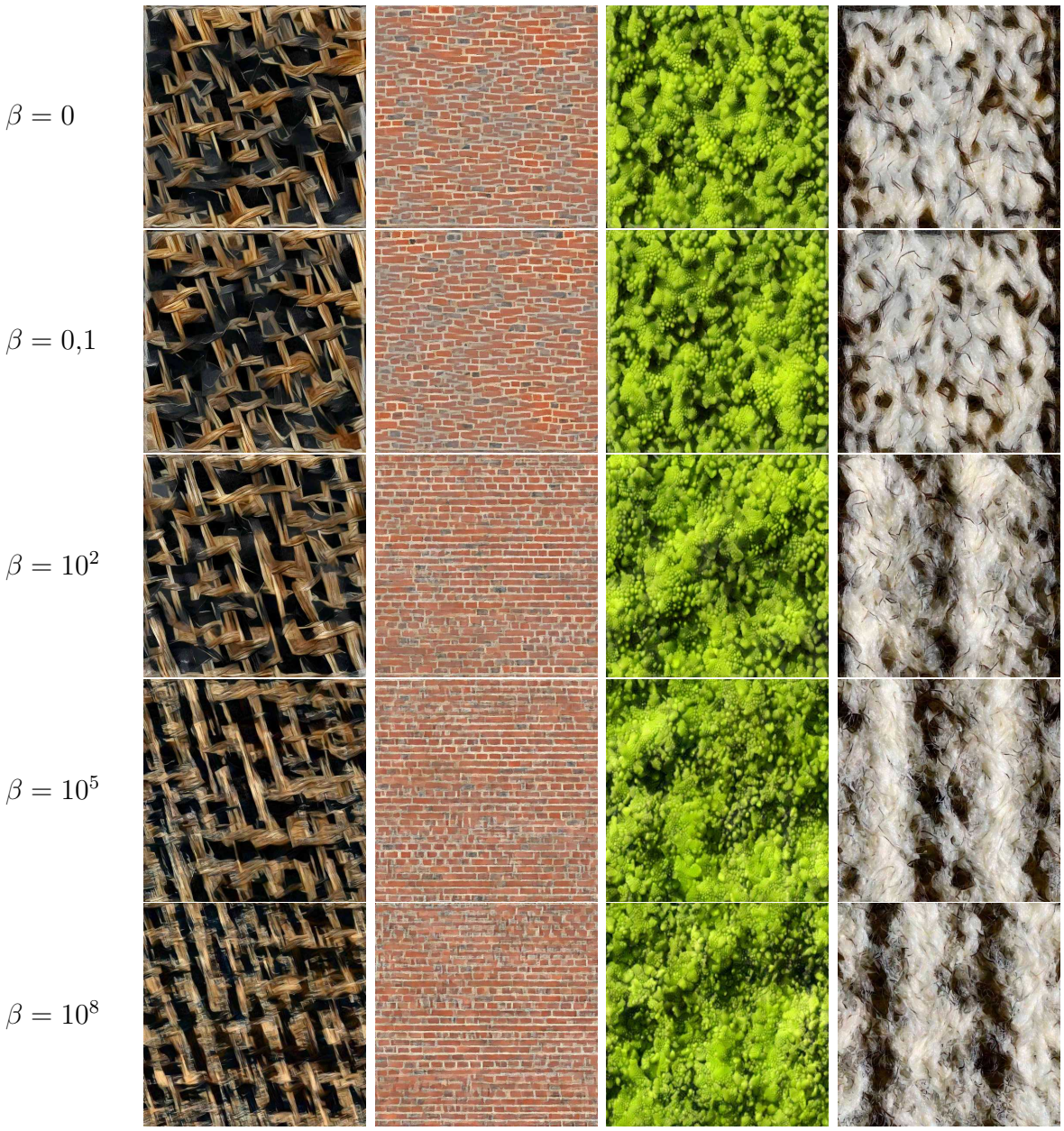


Figure A.45: Synthesis results using different β in equation (3.4) (original can be seen in Figure 3.15), from top to bottom : $\beta = 0$ (i.e. the original method from Gatys et al. [2015b]), $\beta = 10^{-1}, 10^2, 10^5, 10^8$. Only using both the Gram and spectrum constraints simultaneously yields a correct synthesis.

A.6 Additional Higher Resolution Synthesis

In this section, one can see in Figures A.46, A.47, A.48a, A.48b and A.49 to A.51 more synthesis of 2048×2048 images for diverse reference texture[2]. We consider methods "Gatys", "Gram+MRInit", "Gram+Spectrum+MRInit" (both using $K = 3$) and "Snelgrove" [Snelgrove 2017] (with 6 octaves/scales). "Gram+MRInit" provides stunning synthesis for complex texture (see Figures A.46b and A.47a) whereas "Gram+Spectrum+MRInit" yields to excellent synthesis in the regular case once again (see Figure A.48b) even if an adjustment of the β term could help to avoid transparency overlappings that we may observe at this resolution too (see Figure A.50a). At this high resolution, "Gatys + MRInit" or "Snelgrove" may be stuck in bad local minima (see Figure A.48b).

[2]Thanks to my Avenue d'Italie share-house for the pictures of Figures A.48b and A.49 to A.51.

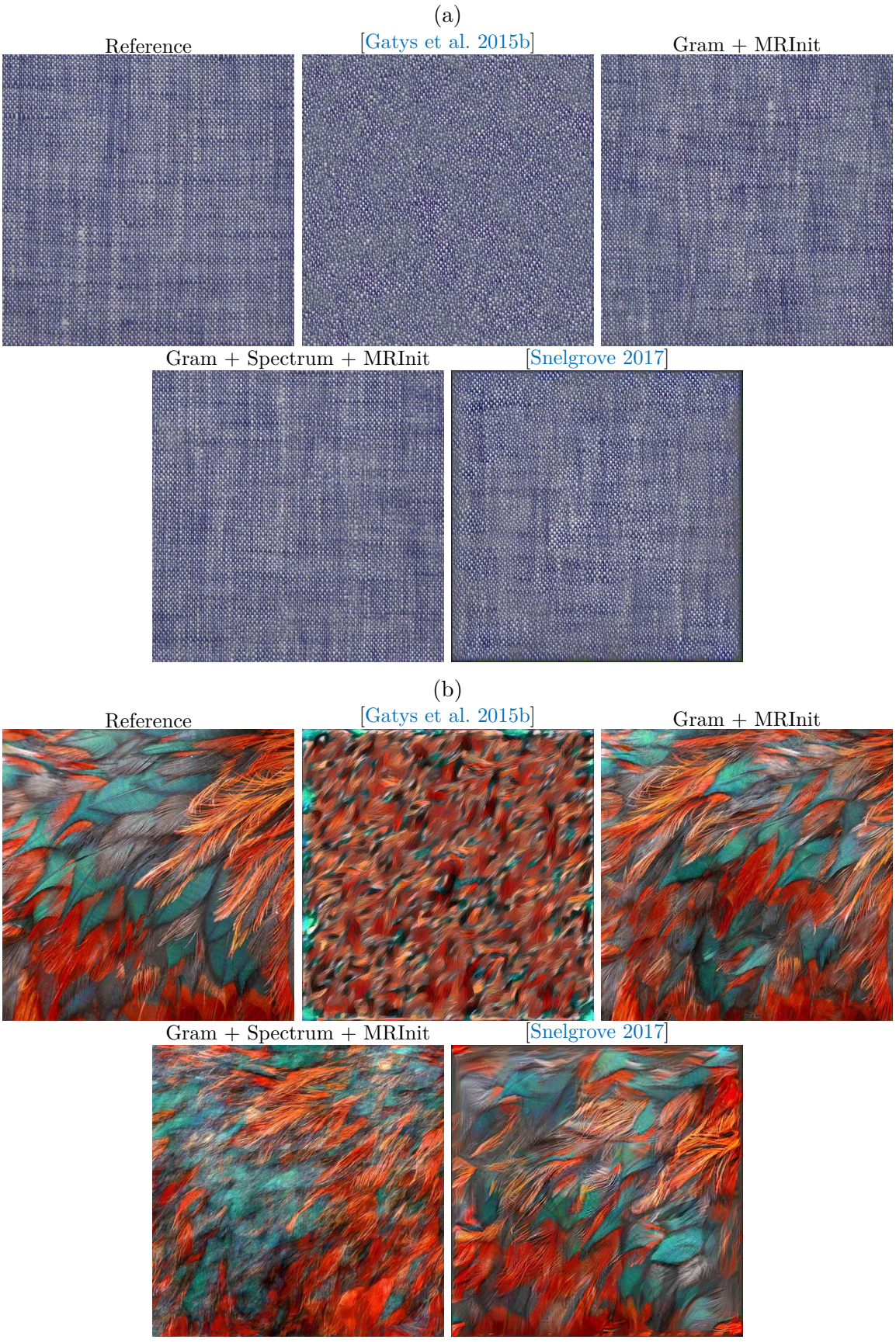


Figure A.46: Synthesis results using different methods for two given references of size 2048×2048.

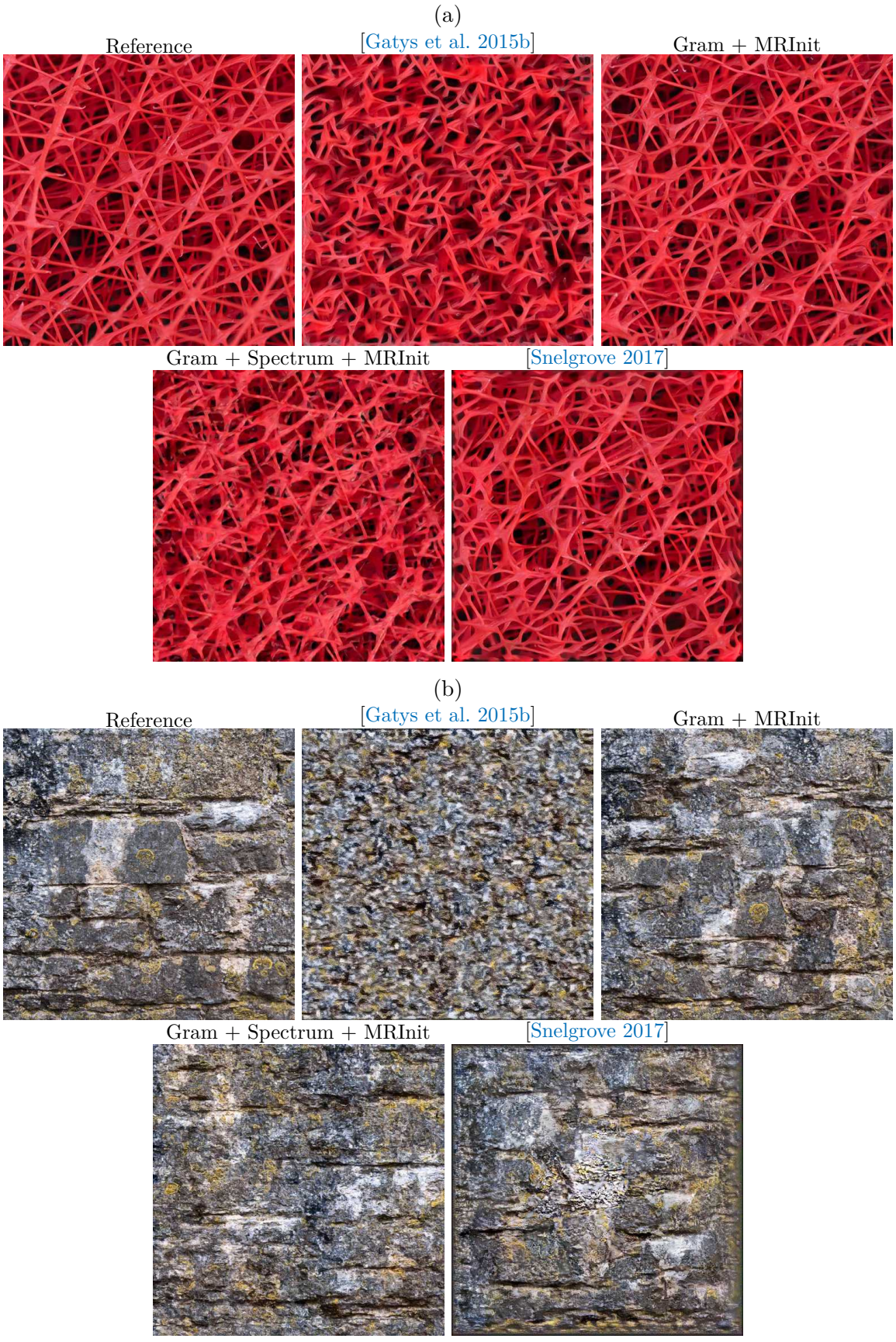


Figure A.47: Synthesis results using different methods for two given references of size 2048×2048.

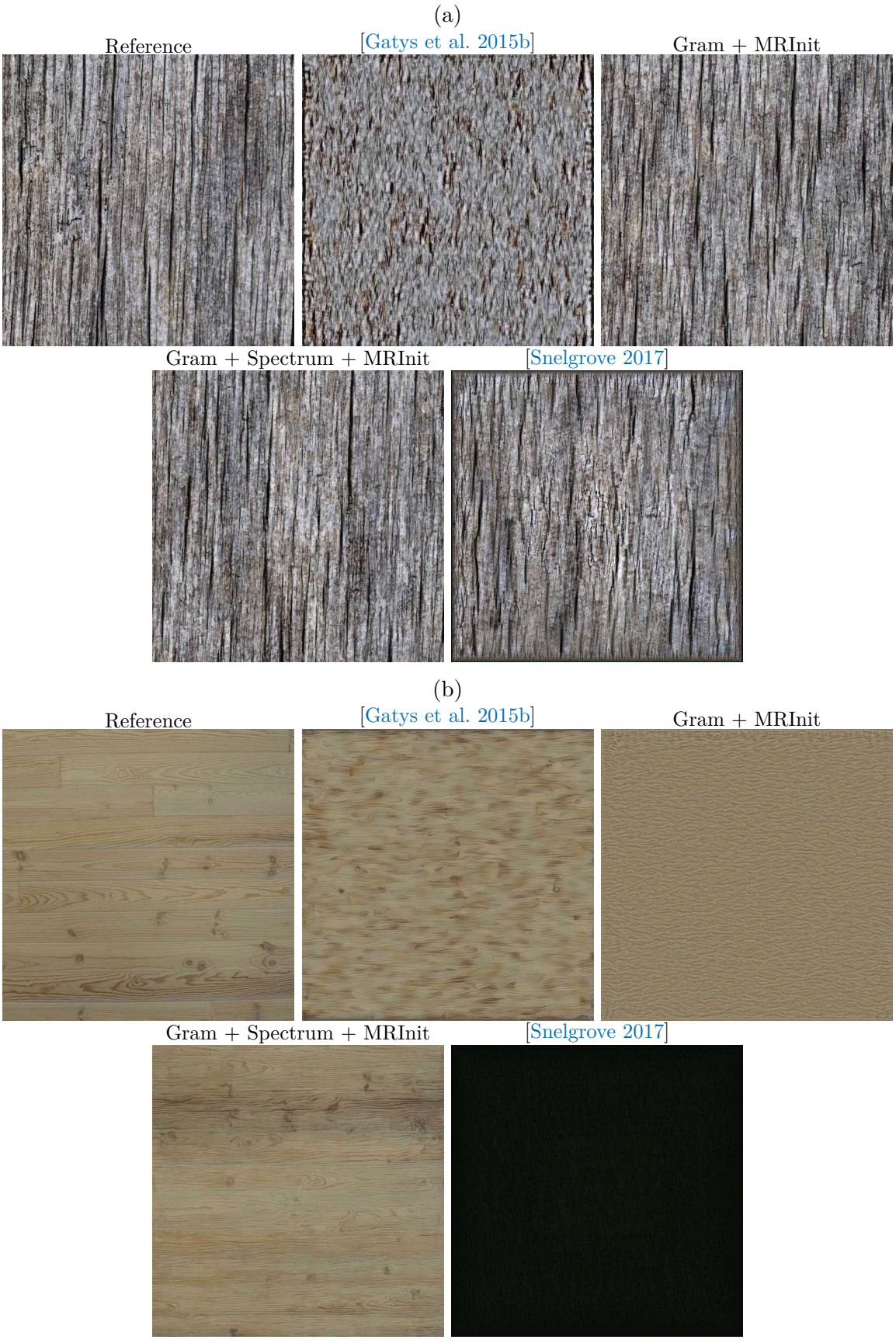


Figure A.48: Synthesis results using different methods for two given references of size 2048×2048.

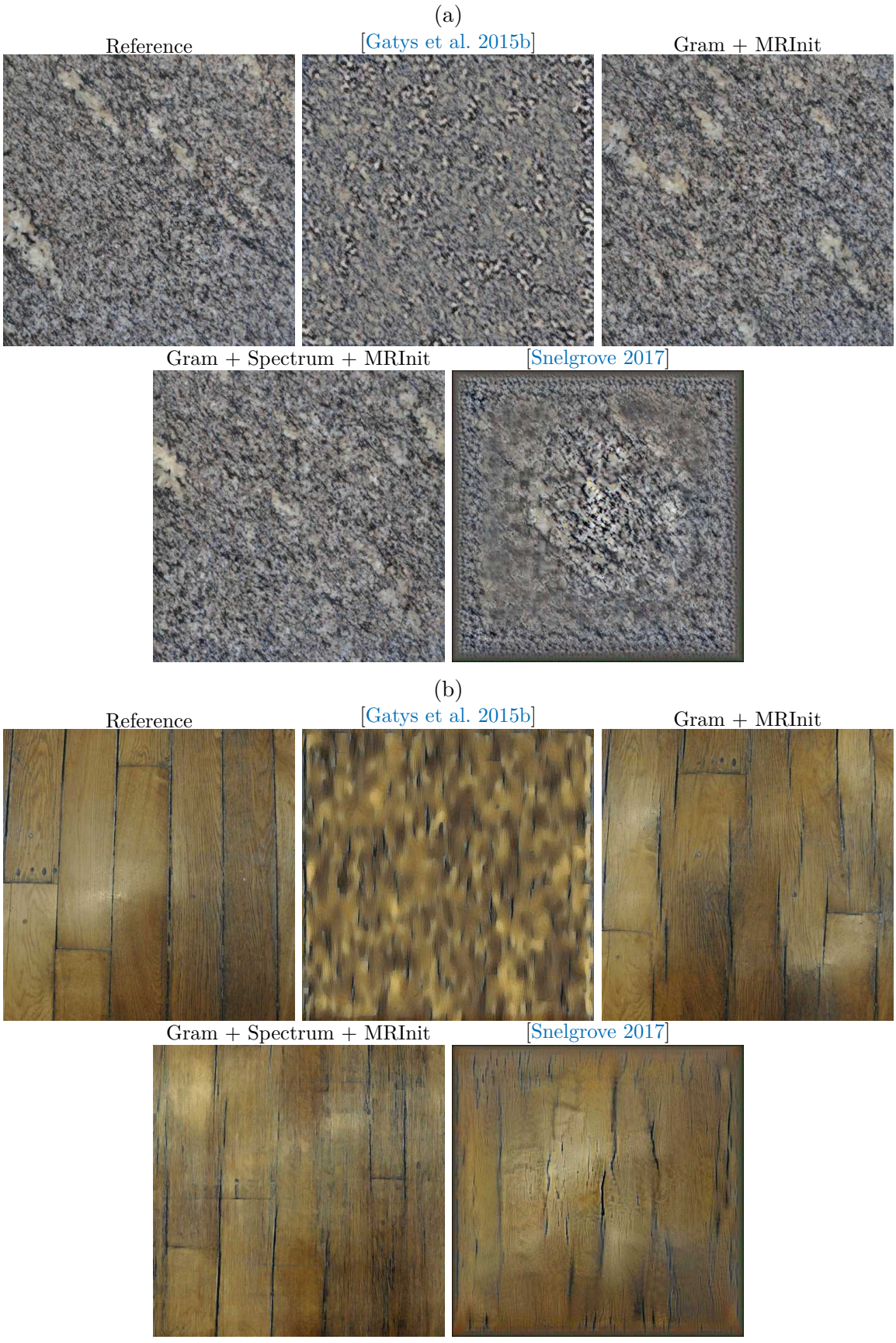


Figure A.49: Synthesis results using different methods for two given references of size 2048×2048.

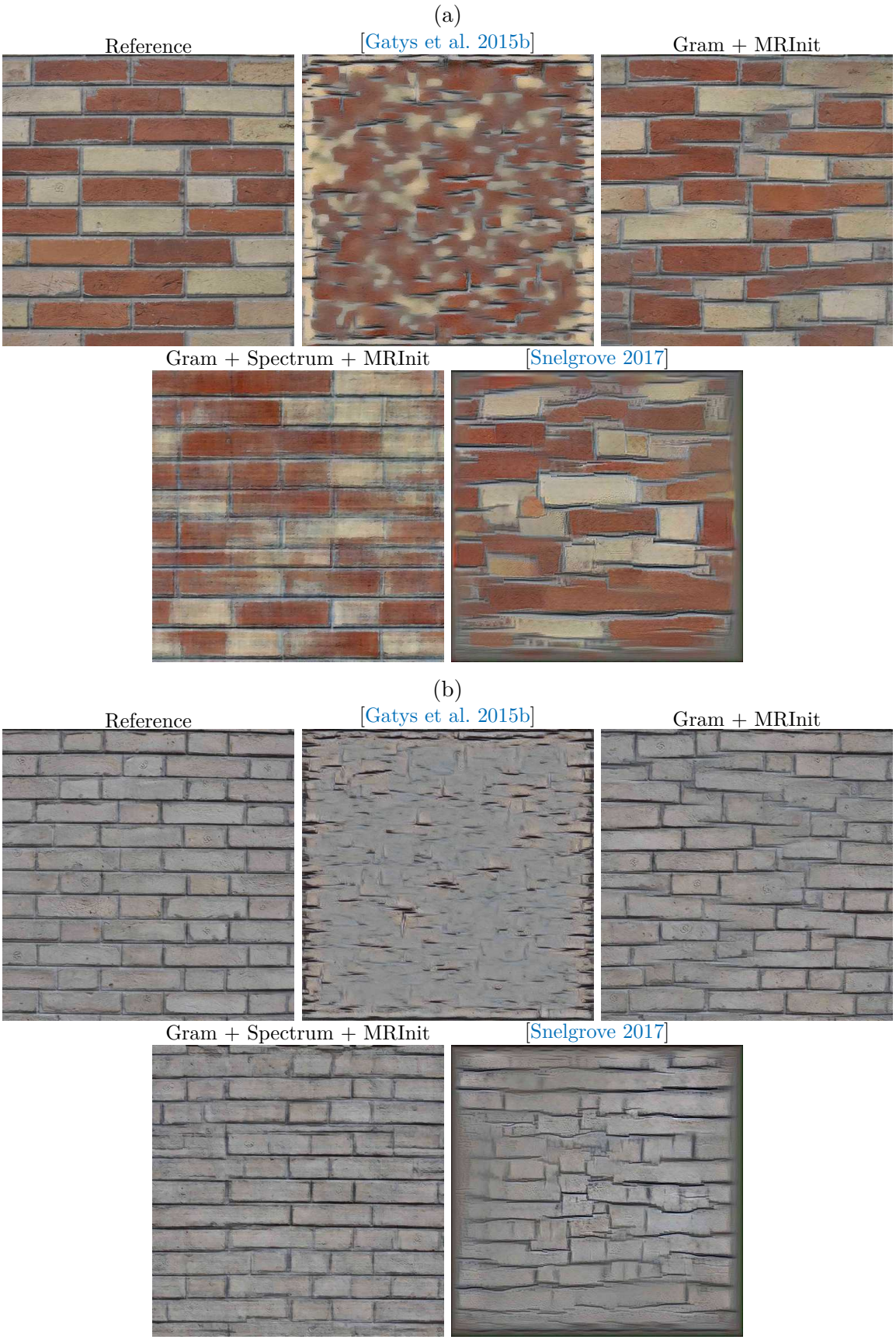


Figure A.50: Synthesis results using different methods for two given references of size 2048×2048.

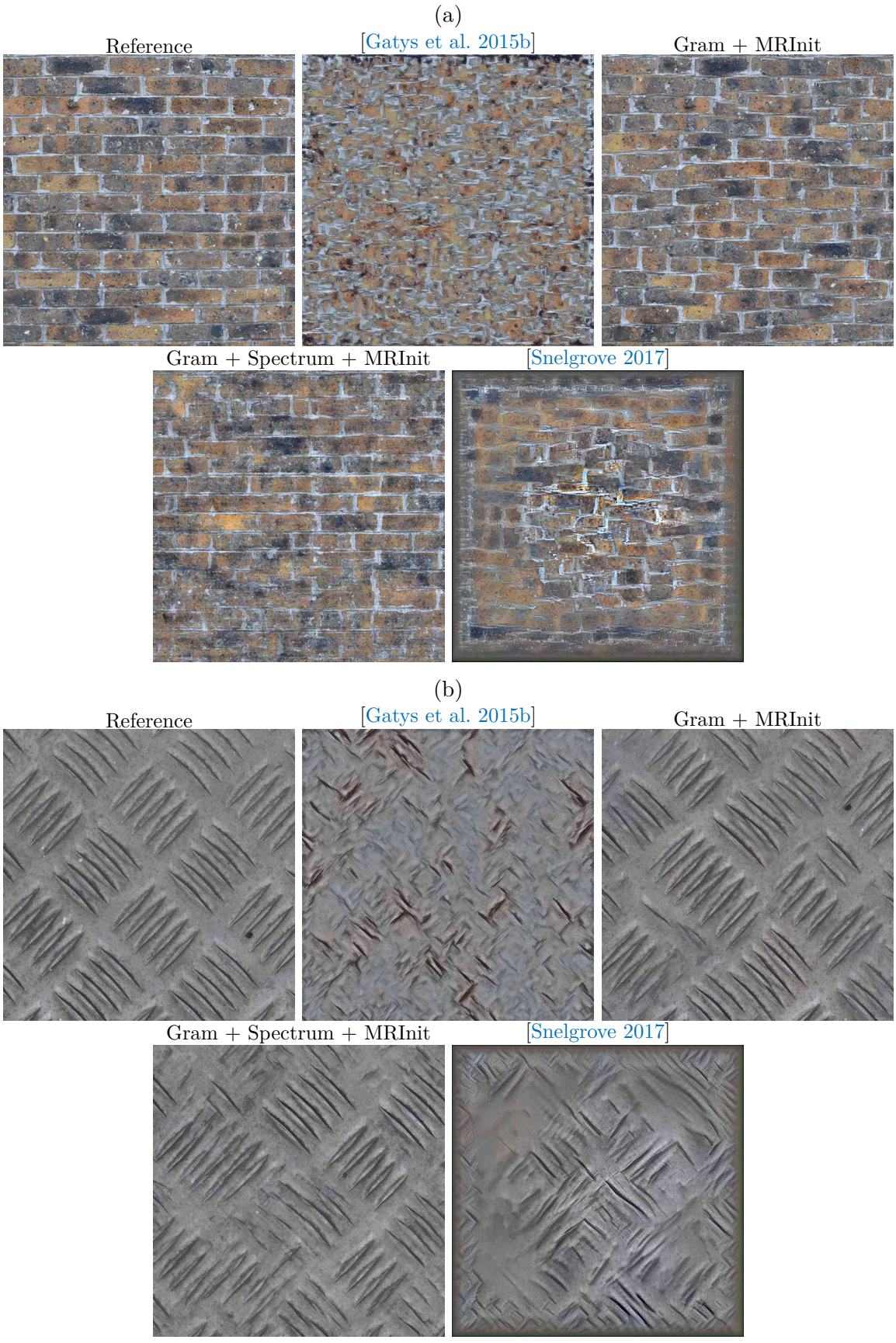


Figure A.51: Synthesis results using different methods for two given references of size 2048×2048.

A.7 Additional Texture Synthesis with Different Statistics

In this section, one can see in more synthesis of low resolution textures with diverse statistics of the feature maps used in the texture model.

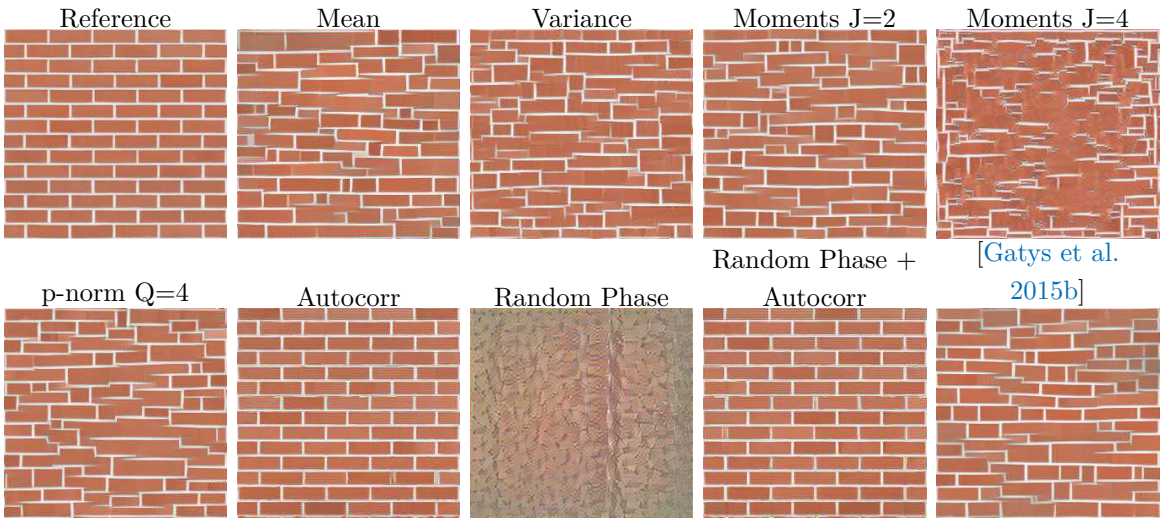


Figure A.52: Synthesis results using different methods for a given reference of size 248×265.

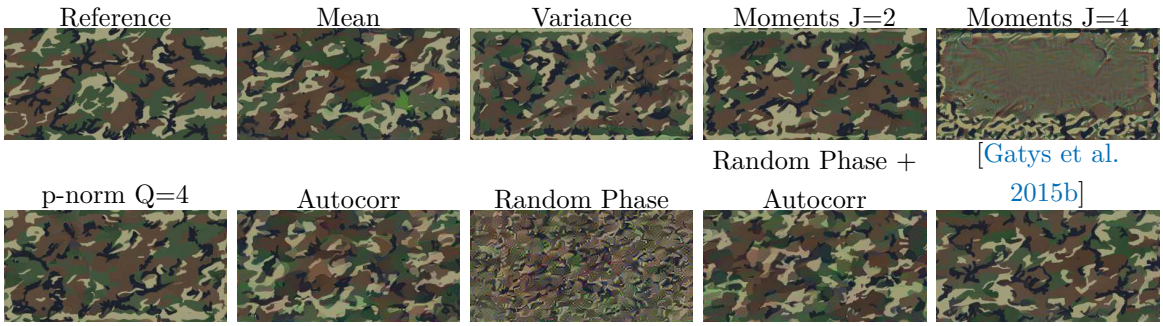


Figure A.53: Synthesis results using different methods for a given reference of size 182×363.

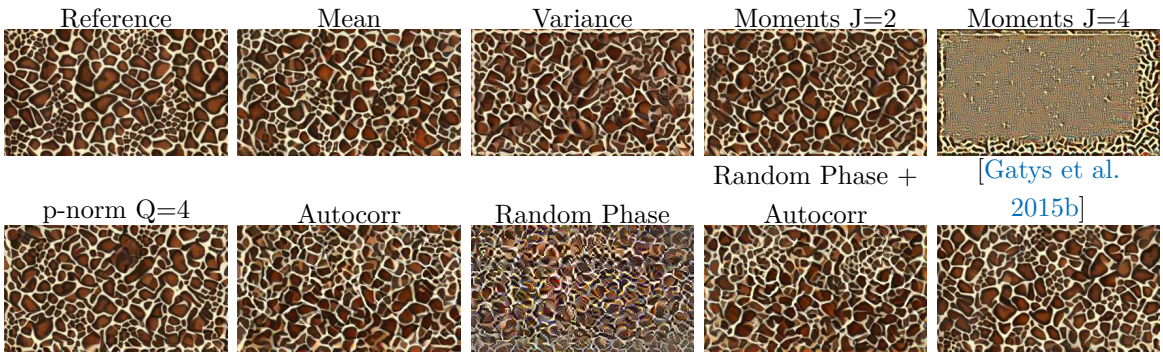


Figure A.54: Synthesis results using different methods for a given reference of size 193×342.

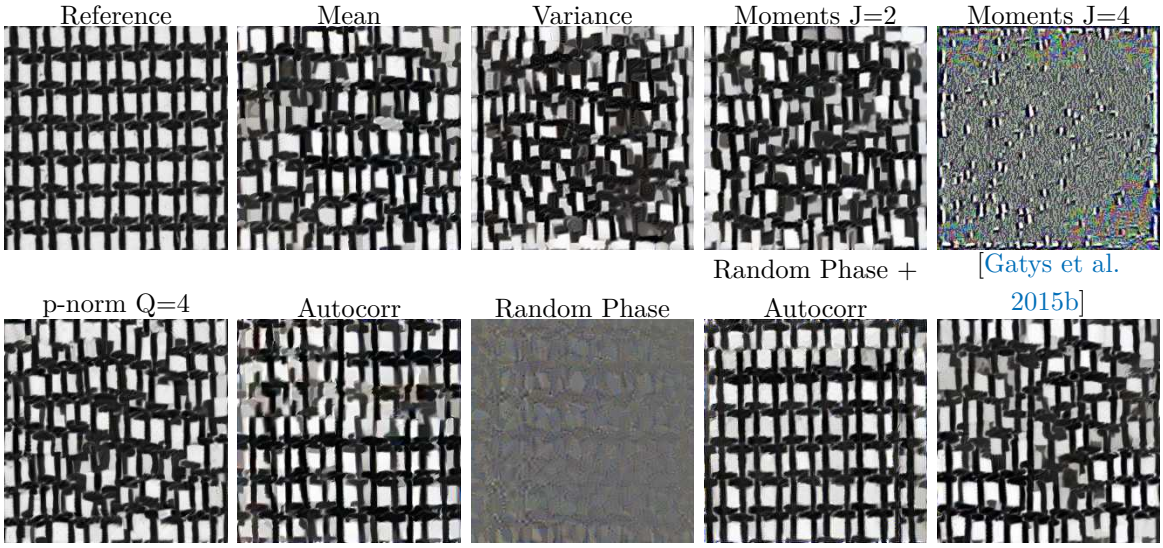


Figure A.55: Synthesis results using different methods for a given reference of size 256×256.

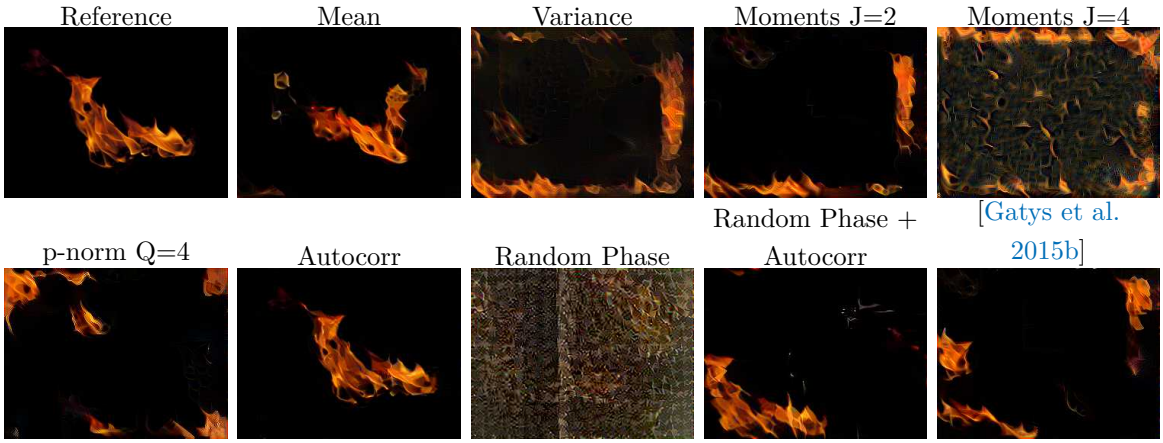


Figure A.56: Synthesis results using different methods for a given reference of size 224×293.

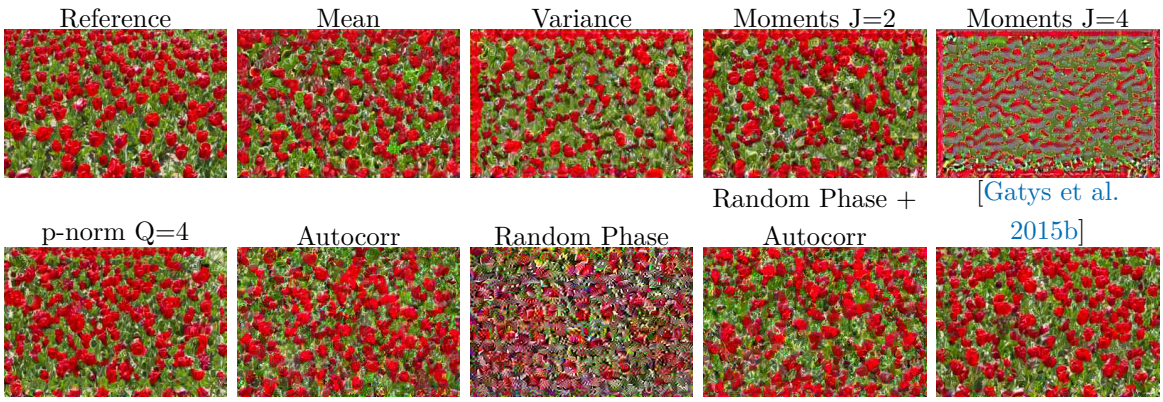


Figure A.57: Synthesis results using different methods for a given reference of size 209×314.

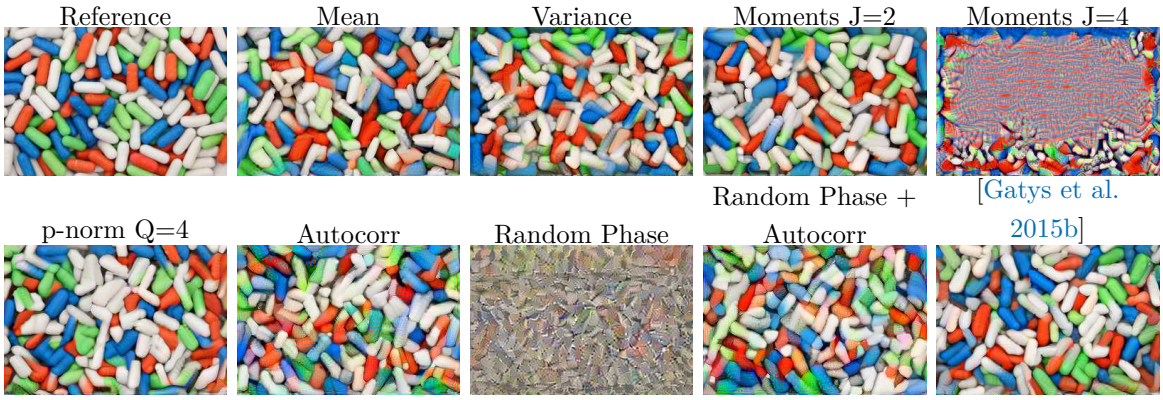


Figure A.58: Synthesis results using different methods for a given reference of size 209×314.

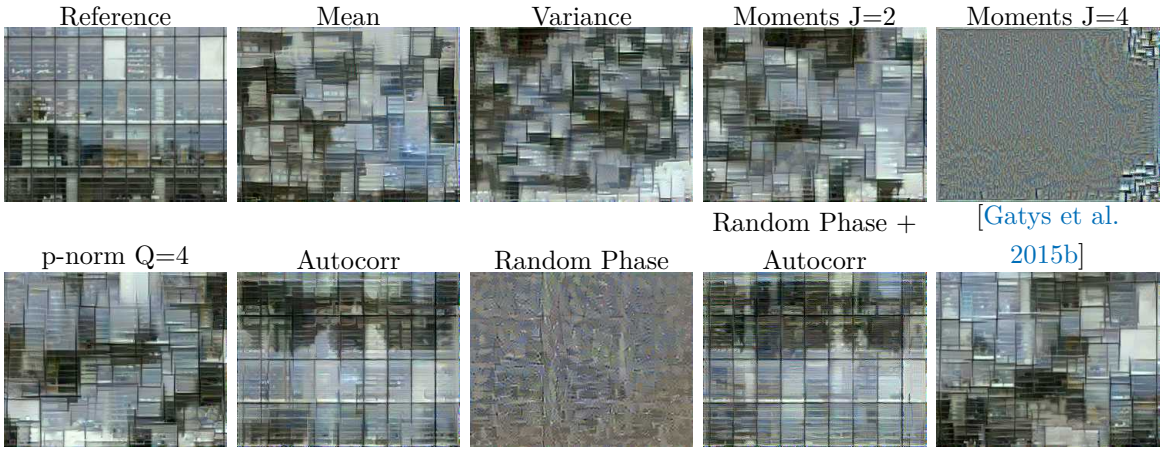


Figure A.59: Synthesis results using different methods for a given reference of size 227×290.

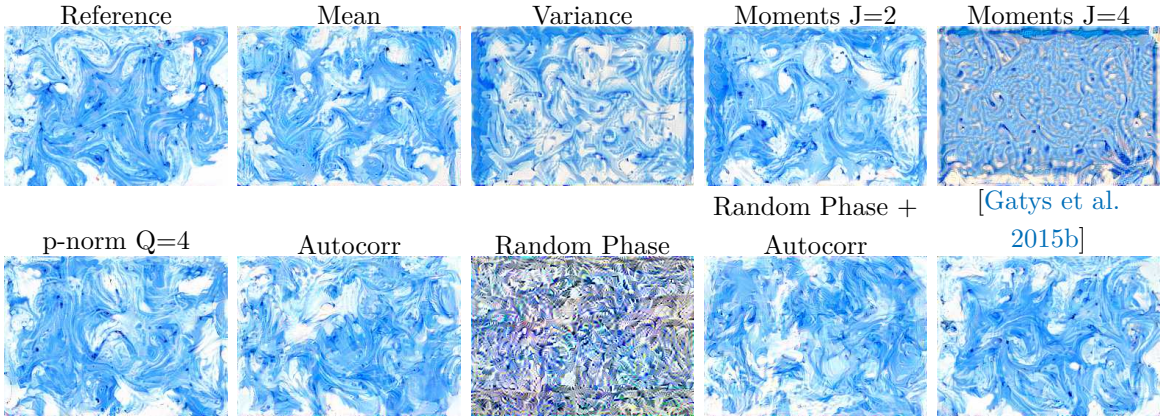


Figure A.60: Synthesis results using different methods for a given reference of size 217×304.

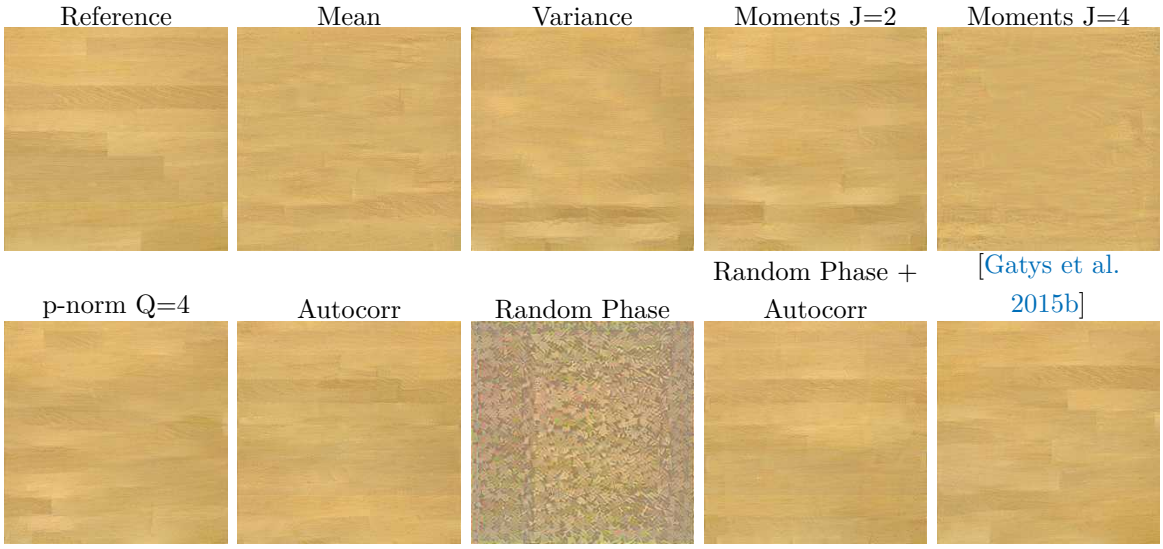


Figure A.61: Synthesis results using different methods for a given reference of size 256×256.

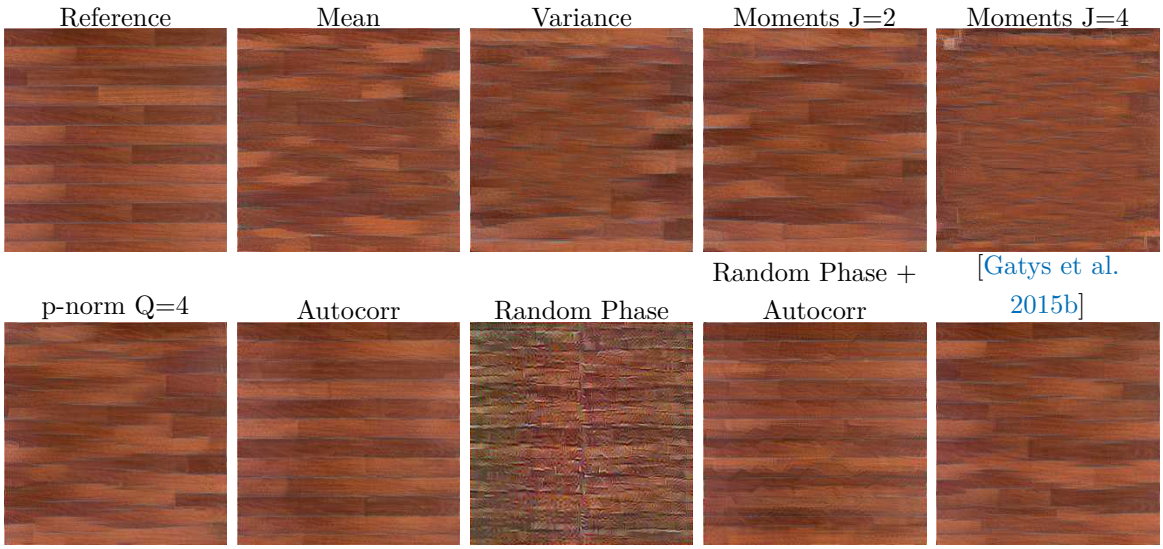


Figure A.62: Synthesis results using different methods for a given reference of size 256×256.

A.8 High Resolution Synthesis for the "Random Phase + Autocorrelation" Method

In this section, one can see in Figures A.63 and A.64, high-resolution images with "Random Phase + Autocorr" and "Random Phase + Autocorr + MRInit" methods defined in Section 3.4.1.2. The method using Random Phase feature maps combined to the autocorrelation fails to provide correct high resolution synthesis. It only captures the coarse grain regularities from the reference image. The multi-resolution version of this method may lead to really good synthesis for some of the reference images but only for a quarter of them. As mentioned before, it maybe exists an optimal value for α_L (Equation (3.20)) that permits to have a more reliable method.

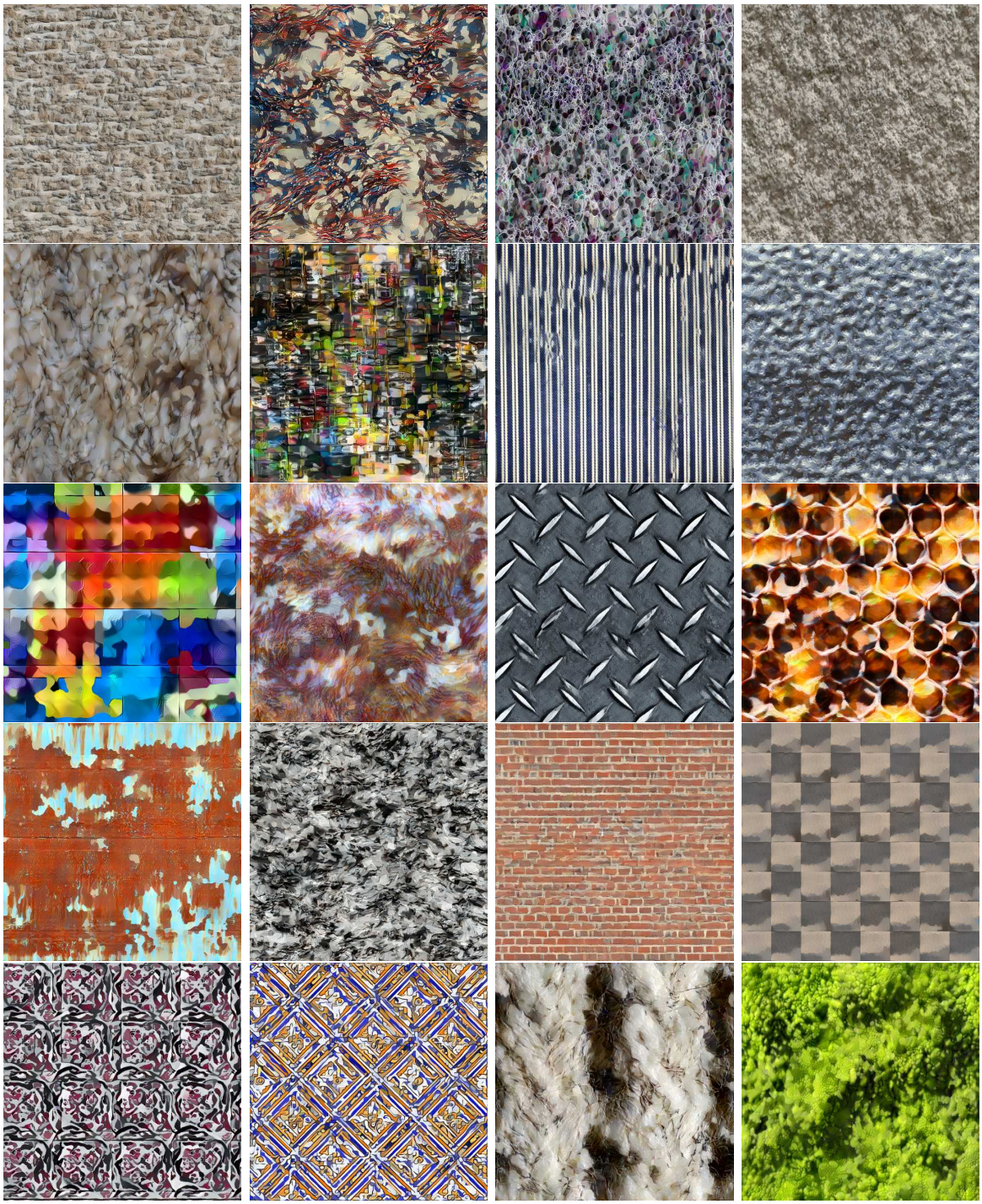


Figure A.63: Synthesis results with Random Phase + Autocorr method for reference of size 1048 × 1048. Reference images may be found in Figure 3.7.

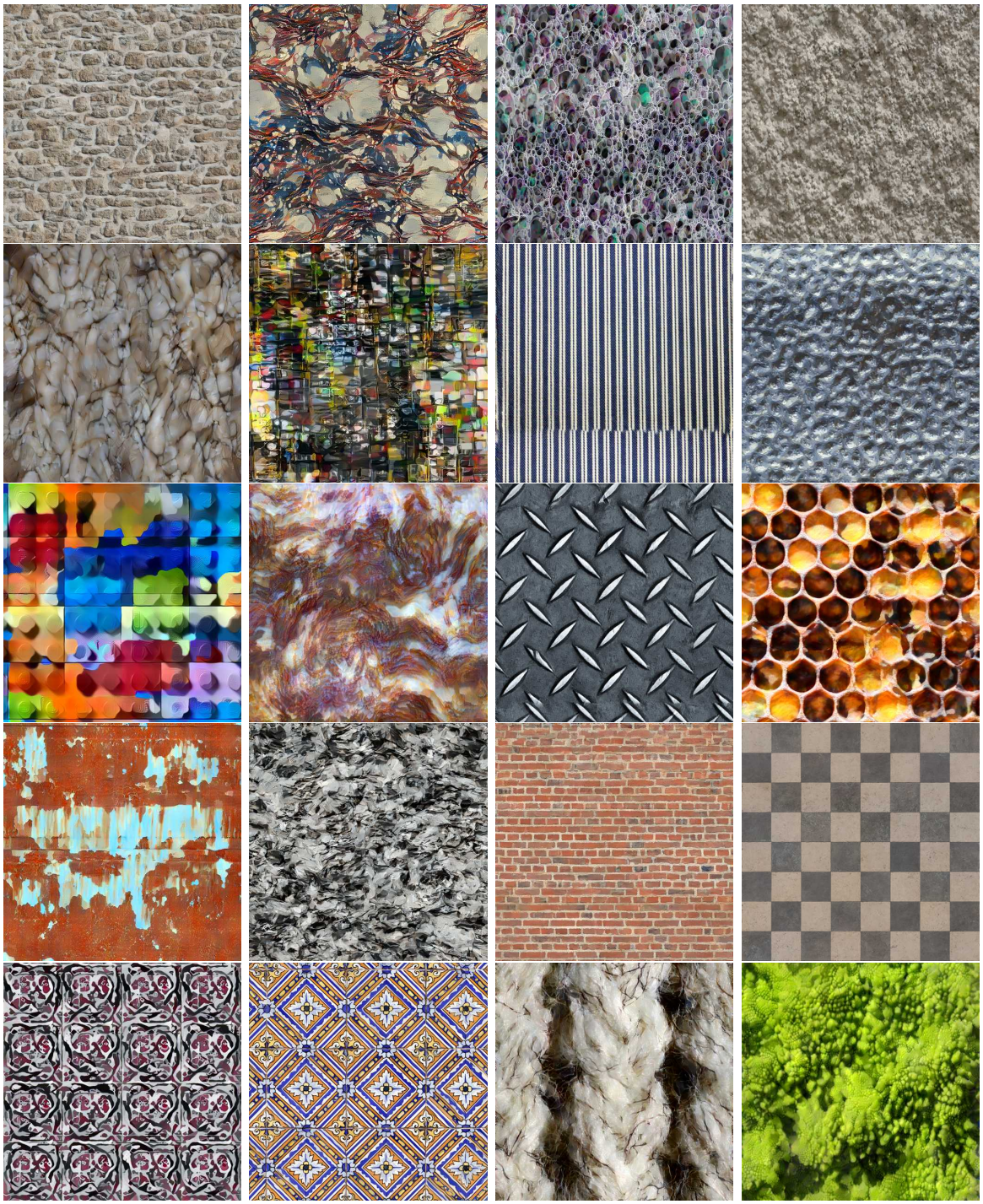


Figure A.64: Synthesis results with Random Phase + Autocorr + MRInit method for a reference image of size 1048 × 1048. Reference images may be found in Figure 3.7.

B

Additional Results for Transfer Learning and Analysis of Convolutional Neural Networks for Art Classification

Contents

B.1 Off-the-shelf Feature Extraction for Paintings and IconArt datasets

In this section, we provide addition classification performances for the off-the-shelf features extraction approach on the Paintings and IconArt datasets (Tables B.1 and B.2). First, we compare the use of different layers of VGG19 for extracting the deep features in the first four rows of the Tables B.1 and B.2. In this case, we train on the fixed deep features one linear one-vs-rest SVM per class with an optimization of the hyper-parameters with a k-fold validation (as in Section 4.3.1.1). We can observe that it is not the penultimate layer of VGG19 that provides the best performance but the one before (named "fc1"). The "relu7" layer is the ReLU output of the "fc2" layer. According to the results, the penultimate layer of a CNN is not always the best one for feature extraction because it may be too specified to the source set.

Then, we compare the different pretrained architecture (VGG19, ResNet50, InceptionV1 etc.). For the IconArt dataset, we can see that the features extracted from a ResNet provide better results than InceptionResNet v2 compared to the Paintings dataset (see Tables B.2 and 4.2). On the Paintings dataset, InceptionResNet v2 provides the best result. In [Uijlings et al. 2018], it is experimentally shown that the ResNet are some of the best CNN for features extraction. Moreover, we compare different classifier on the extracted features:

- SVM with the previously mentioned setup
- MLP 1 layer with learning rate equal 0.01, 20 epochs and a selection of the best model on validation set
- MLP 2 layers with a hidden layer of 256 units and the same parameters as above
- MLP 3 layers with a hidden layer of 256 units and then one with 128 units, a learning rate equal to 0.01 with a decay of 0.0005, a momentum equal to 0.99 and 20 epochs

The MLPs are trained by SGD with a momentum only in the last case. The SVM based approach always provides the best performance due to the optimization of the hyperparameters but training one SVM per class is more costly than training one MLP for all the classes together. A fairer comparison implies to do an optimization of hyperparameters of the MLPs (especially the learning rate, number of epochs and the regularization scheme). Nevertheless, the optimal number of layers for the MLP depends on the deep features used. This explains the performance gap between Section 4.3.1.1 and Section 4.3.2.2.

Finally, the use of a Faster R-CNN detection network on IconArt does not provide good performances. A better solution is proposed in Chapter 5.

Net	Layer	Method	angel	JCchild	crucifixion	Mary	nudity	ruins	StSeb	mean
VGG19	relu7	SVM	40.1	72.8	54.3	79.2	72.0	72.7	31.5	60.3
	fc2		58.5	75.1	53.8	83.3	73.5	73.5	24.5	63.2
	fc1		60.2	77.7	64.1	82.8	74.0	74.0	24.9	**65.4**
	block5_pool		55.2	75.6	52.9	82.8	72.3	71.7	20.9	61.6
		MLP 1 layer	44.1	70.5	35.0	79.0	68.1	66.7	9.4	53.3
		MLP 2 layers	46.6	71.1	38.7	78.5	66.5	69.0	9.4	54.2
		MLP 3 layers	57.3	76.4	45.6	83.2	72.6	68.6	6.7	58.6
ResNet50	activation_48	SVM	57.6	80.7	68.9	84.7	77.1	75.3	33.7	**68.3**
		MLP 1 layer	55.6	76.9	49.5	81.0	73.7	71.1	10.7	59.8
		MLP 2 layers	55.5	78.4	48.7	81.9	74.8	71.6	15.8	61.0
		MLP 3 layers	54.8	80.4	26.4	84.4	74.8	68.9	5.9	56.5
InceptionV1	avgpool	SVM	49.2	73.3	61.0	80.0	70.9	72.0	19.1	**60.8**
		MLP 1 layer	39.4	65.5	43.3	73.4	66.1	67.3	10.0	52.1
		MLP 2 layers	38.2	64.7	23.1	72.1	65.2	64.6	6.6	47.8
		MLP 3 layers	46.4	73.7	61.4	79.7	70.1	71.3	11.4	59.1
ResNet-152	fc7-2048-D	SVM	41.8	77.9	63.3	82.2	73.7	74.8	28.7	63.2
InceptionResNet v2	1536D		44.1	77.2	57.8	81.1	77.4	74.6	26.8	62.7
RES-152-COCO	fc7-2048-D	SVM on max objectness[a]	49.3	74.7	30.3	67.5	57.4	43.2	7.0	47.1

Table B.1: **IconArt classification test set** average precision (%) in the case of off-the-shelf features extraction with different networks, different layers and final classifier.

[a] This method is referred as MAX in the Chapter 5.

Net	Layer	Method	aeroplane	bird	boat	chair	cow	diningtable	dog	horse	sheep	train	mean
VGG19	fc2	SVM	67.1	50.6	93.0	74.6	61.3	70.2	56.1	78.8	67.1	85.5	**70.5**
	fc1		69.6	48.7	93.5	74.9	62.0	70.7	55.4	78.6	67.0	84.9	**70.5**
	block5_pool		66.9	47.0	92.2	74.7	57.7	69.5	51.4	76.7	67.2	83.4	68.7
		MLP 1 layer	46.6	36.5	89.0	65.2	46.7	58.9	41.3	72.3	54.2	76.4	58.7
		MLP 2 layers	47.6	37.7	90.0	71.0	45.7	62.7	45.8	73.8	60.5	75.9	61.1
		MLP 3 layers	21.5	44.3	91.2	71.3	37.0	62.4	50.5	76.6	58.6	78.7	59.2
ResNet50	activation_48	SVM	70.0	52.4	93.5	76.3	63.4	73.0	58.4	80.8	70.9	88.3	**72.7**
		MLP 1 layer	55.2	45.0	91.8	70.7	52.1	66.7	51.4	77.7	62.9	83.1	65.7
		MLP 2 layers	57.4	45.3	92.1	71.6	51.2	68.0	52.7	78.1	64.6	84.6	66.5
		MLP 3 layers	10.5	22.8	91.2	69.7	29.3	54.0	36.4	73.9	55.7	70.4	51.4
InceptionV1	avgpool	SVM	63.4	45.6	90.7	69.6	52.6	66.2	47.4	76.3	65.2	82.4	**65.9**
		MLP 1 layer	44.8	28.0	85.5	61.7	39.3	56.5	34.1	68.2	49.9	74.1	54.2
		MLP 2 layers	36.5	27.4	86.9	59.8	45.2	54.1	36.8	67.3	52.3	64.9	53.1
		MLP 3 layers	53.9	45.3	90.5	68.5	51.3	63.4	47.9	76.6	62.3	79.6	63.9

Table B.2: **Paintings test set** average precision (%) in the case of off-the-shelf features extraction with different networks, different layers and final classifier.

B.2 Analyzing Convolutional Neural Networks trained for Art Classification

In this section, the reader may find additional results about feature visualization of CNNs, for InceptionV1, VGG19 and ResNet50 models fine-tuned or trained on RASTA [Lecoutre et al. 2017].

B.2.1 High-level Layers of a Fine-tuned InceptionV1

Optimized Images and Maximal activation images for high-level layers of an InceptionV1 fine-tuned on RASTA can be found in Figures B.1 and B.2.

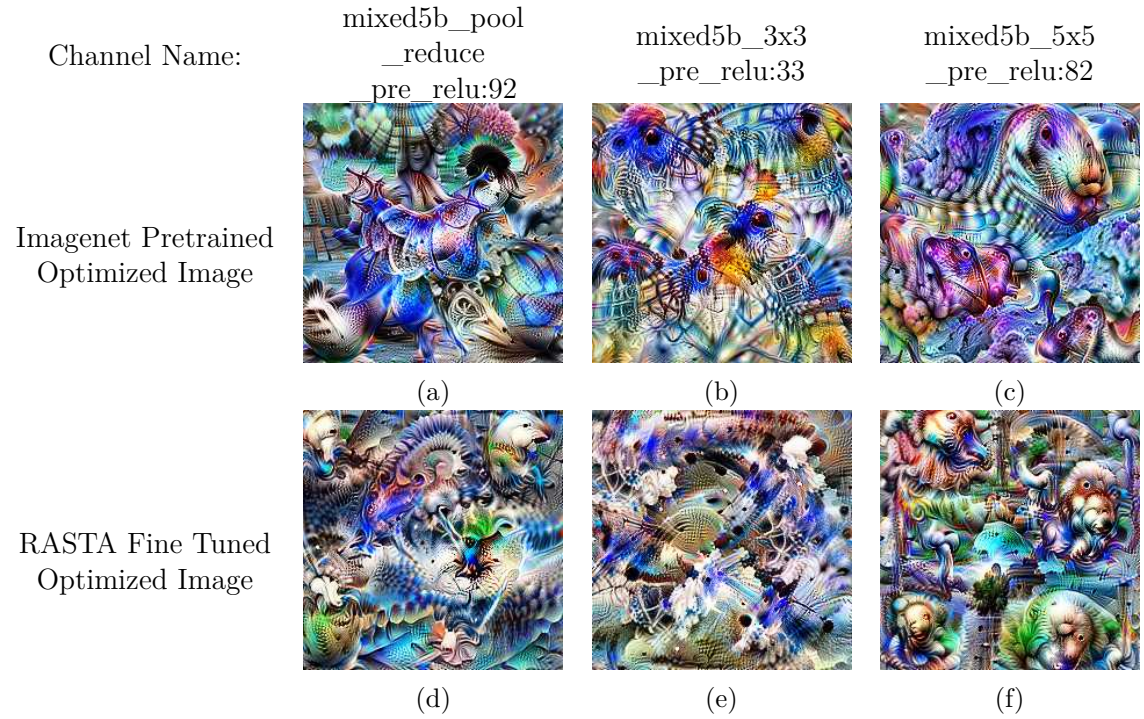


Figure B.1: Optimized Image from different high-level layers. First row InceptionV1 pretrained on ImagneNet, second row fine-tuned on RASTA [Lecoutre et al. 2017].

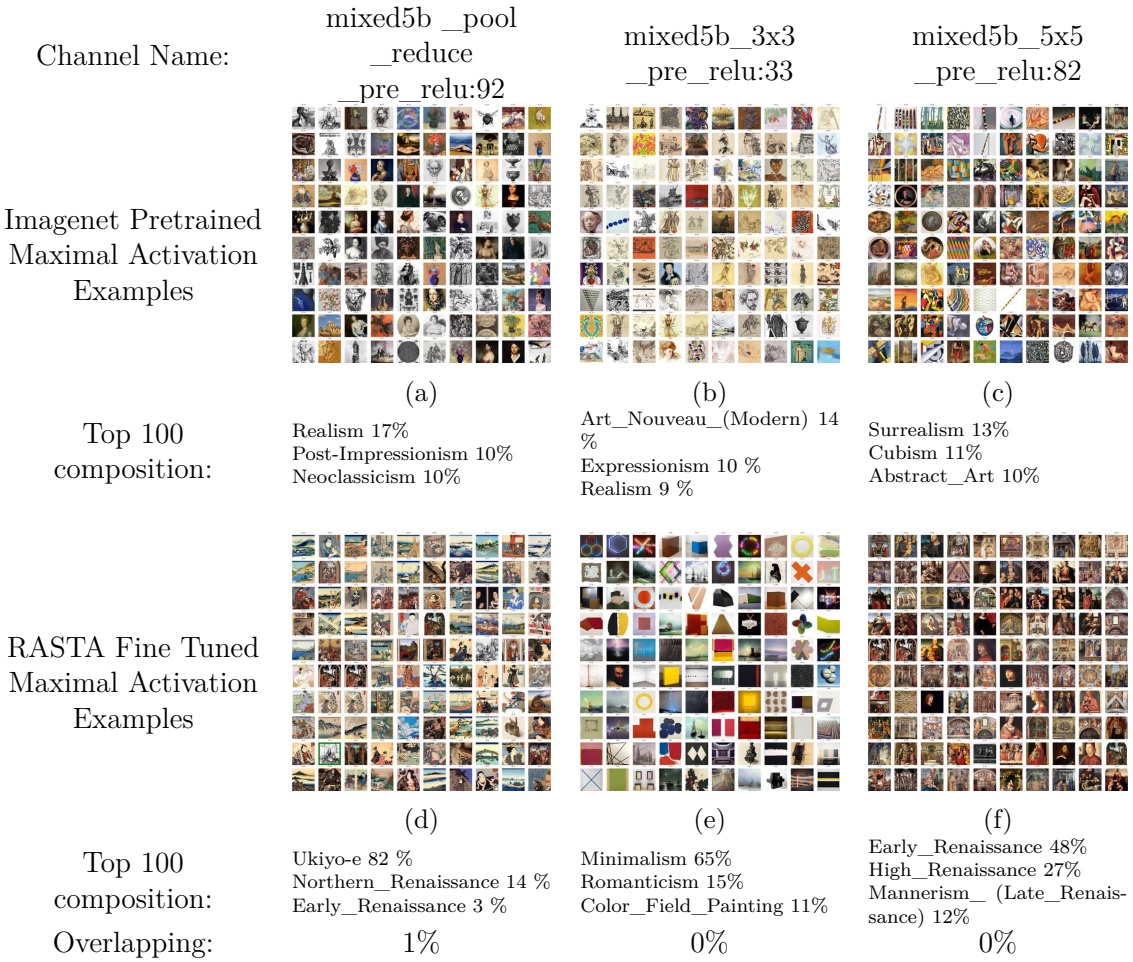


Figure B.2: Maximal Activation Examples for a given channel corresponding to Figure B.1. First row InceptionV1 pretrained on ImagneNet, second row fine-tuned on RASTA [Lecoutre et al. 2017]. The percentage of the 3 most common class is displayed under the images. The percentage of overlapping between the two sets of maximal activation images is displayed at the bottom of each column. The images surrounded by a green line are already present in the top 100 of the pre-trained model.

B.2.2 InceptionV1 Trained From Scratch

More complete example of optimized images and maximal activation examples are displayed in Figures B.3 and B.4.

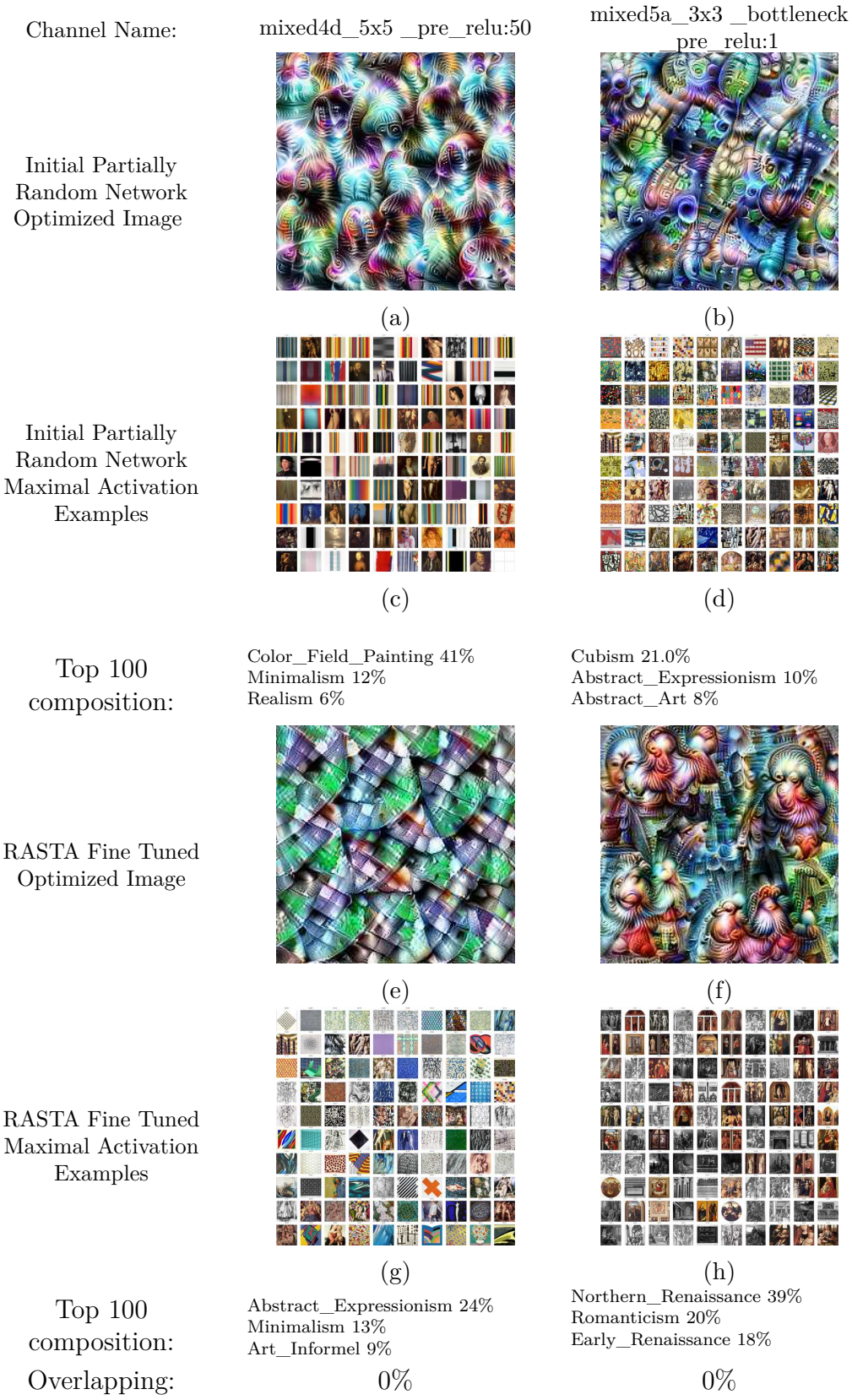


Figure B.3: Optimized Image and Maximal activation examples for the model fine-tuned with low-level layers initialized from ImagNet and upper layers initialized at random. First row: optimized images for the initial partially random model. Second row: top 100 maximal activation examples for the same model. Third and fourth rows: optimized images and maximal activation examples for the same channel of the model trained from scratch. The percentage of the 3 most common class is displayed under the images. The overlapping ratio between Top 100 maximal activation images is printed on the last line.

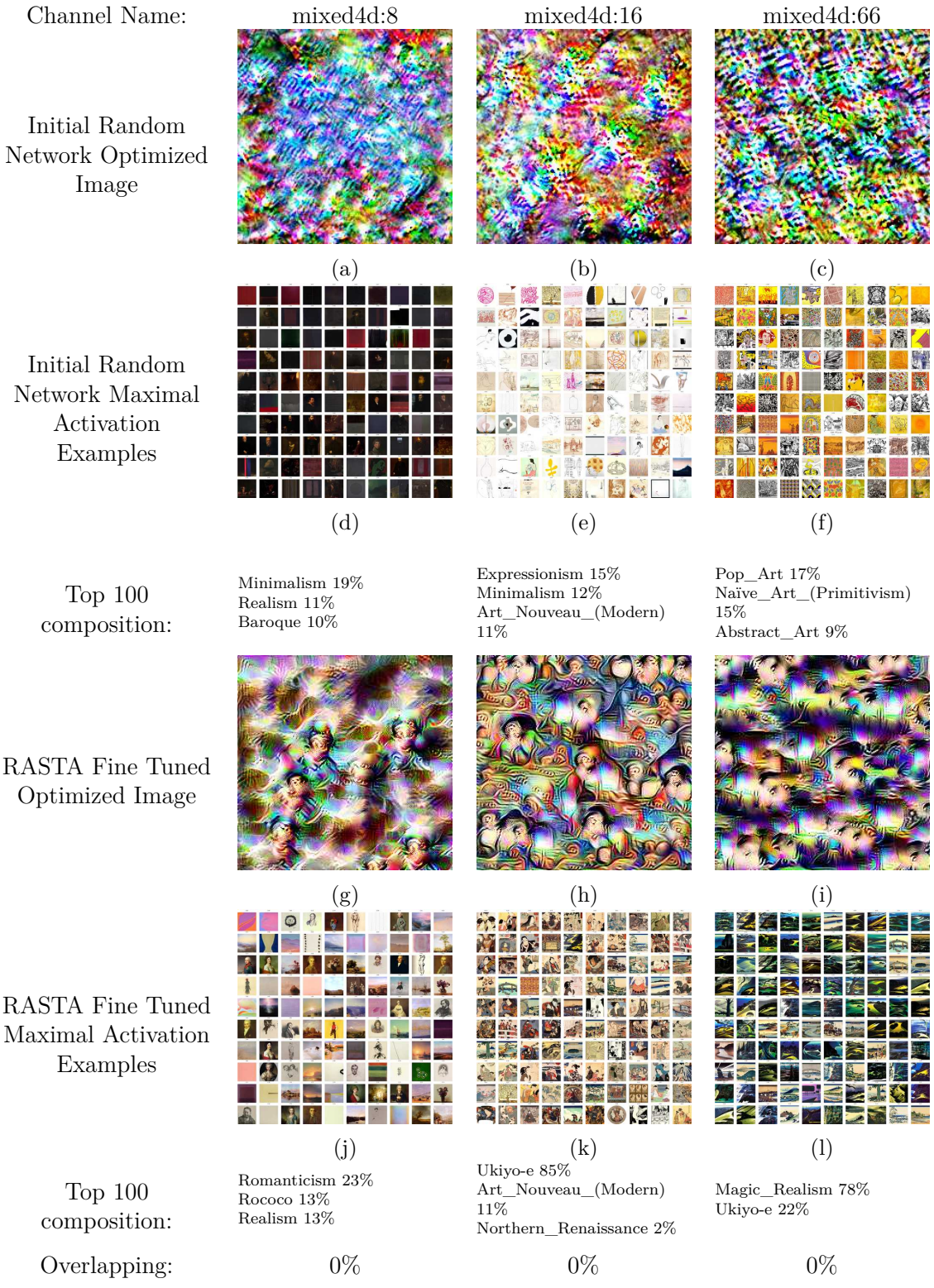
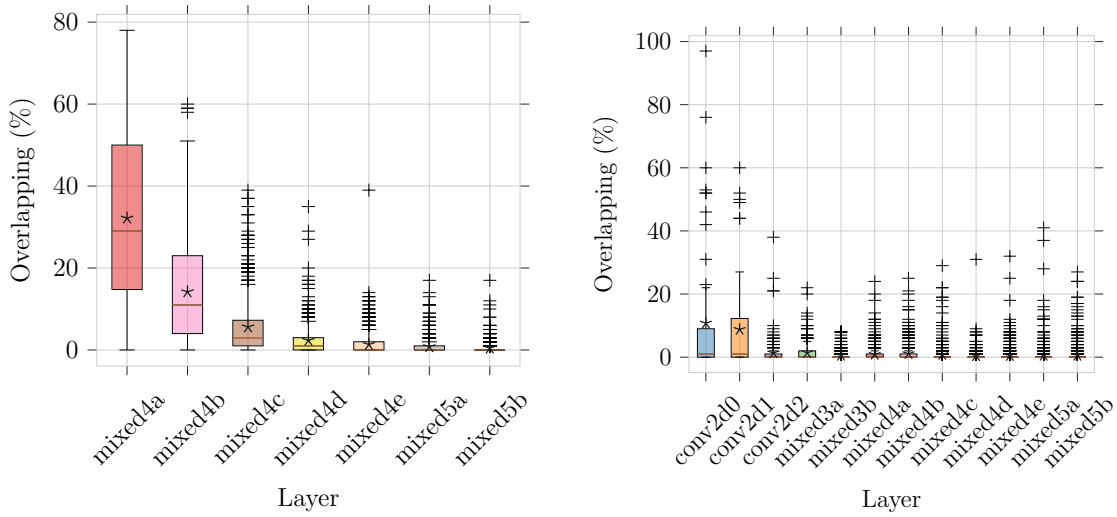


Figure B.4: First row: optimized images for the initial random model. Second row: top 100 maximal activation examples for the same model. Third and fourth rows: optimized images and maximal activation examples for the same channel of the model trained from scratch. The percentage of the 3 most common class is displayed under the images. The overlapping ratio between Top 100 maximal activation images is printed on the last line.



(a) Model fine-tuned on RASTA with low-level layers initialized from ImagNet and upper layers initialized at random.

(b) Model trained from scratch on RASTA.

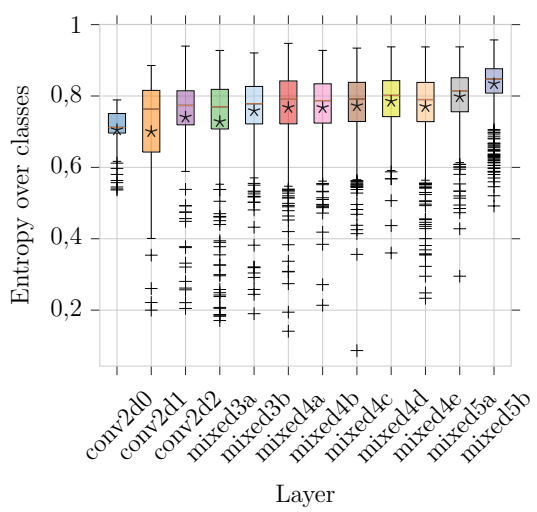
Figure B.5: Boxplots of the overlapping ratio on the top 100 maximal activation images for the two different models trained from scratch (between the random initialization and the trained model). For each box, the horizontal orange line corresponds to the average result and the star to the median. The crosses are outliers (i.e. points outside 1.5 times the interquartile range).

B.2.3 Quantitative Evaluations of InceptionV1

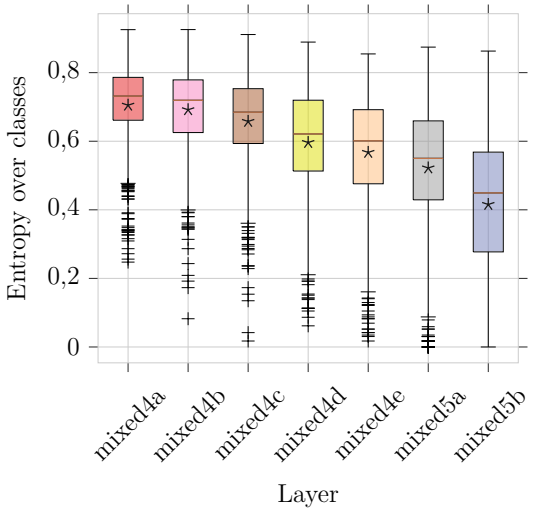
Boxplots of some metrics on the top 100 maximal activation images for other models can be found in Figures B.5 and B.6. We can observe a decreasing of the entropy for two models trained from scratch in Figures B.6b and B.6c.

B.2.4 Optimized Images for ResNet50 or VGG19 Fine-tuned on RASTA

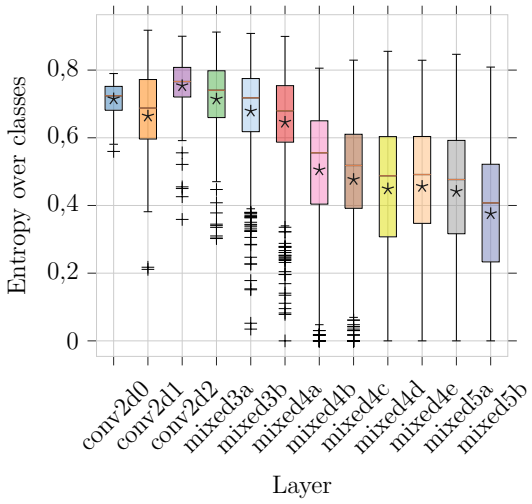
We also look at the optimized images from a VGG19 or a ResNet50 pretrained on ImageNet and fine-tuned on RASTA [Lecoutre et al. 2017] (see Figures B.7 and B.8). The models are trained with the Mode A (Table 4.4). The detailed architecture can be found in Tables D.2 and D.3. The low-level layers of these CNNs seems to stay identical. For ResNet50, we may see a face detector in some of the channels (see Figures B.7n and B.7p) or a "tree in front of a blue sky" (see Figure B.7g). A drapery detector might be recognized in high level filters from VGG19 (see Figure B.8p).



(a) ImageNet pretaining.



(b) Model fine-tuned on RASTA with low-level layers initialized from ImagNet and upper layers initialized at random.



(c) Model trained from scratch on RASTA.

Figure B.6: Boxplots of the entropy over classes on the top 100 maximal activation images for the pretrained model and two different models trained from scratch. For each box, the horizontal orange line corresponds to the average result and the star to the median. The crosses are outliers (i.e. points outside 1.5 times the interquartile range).

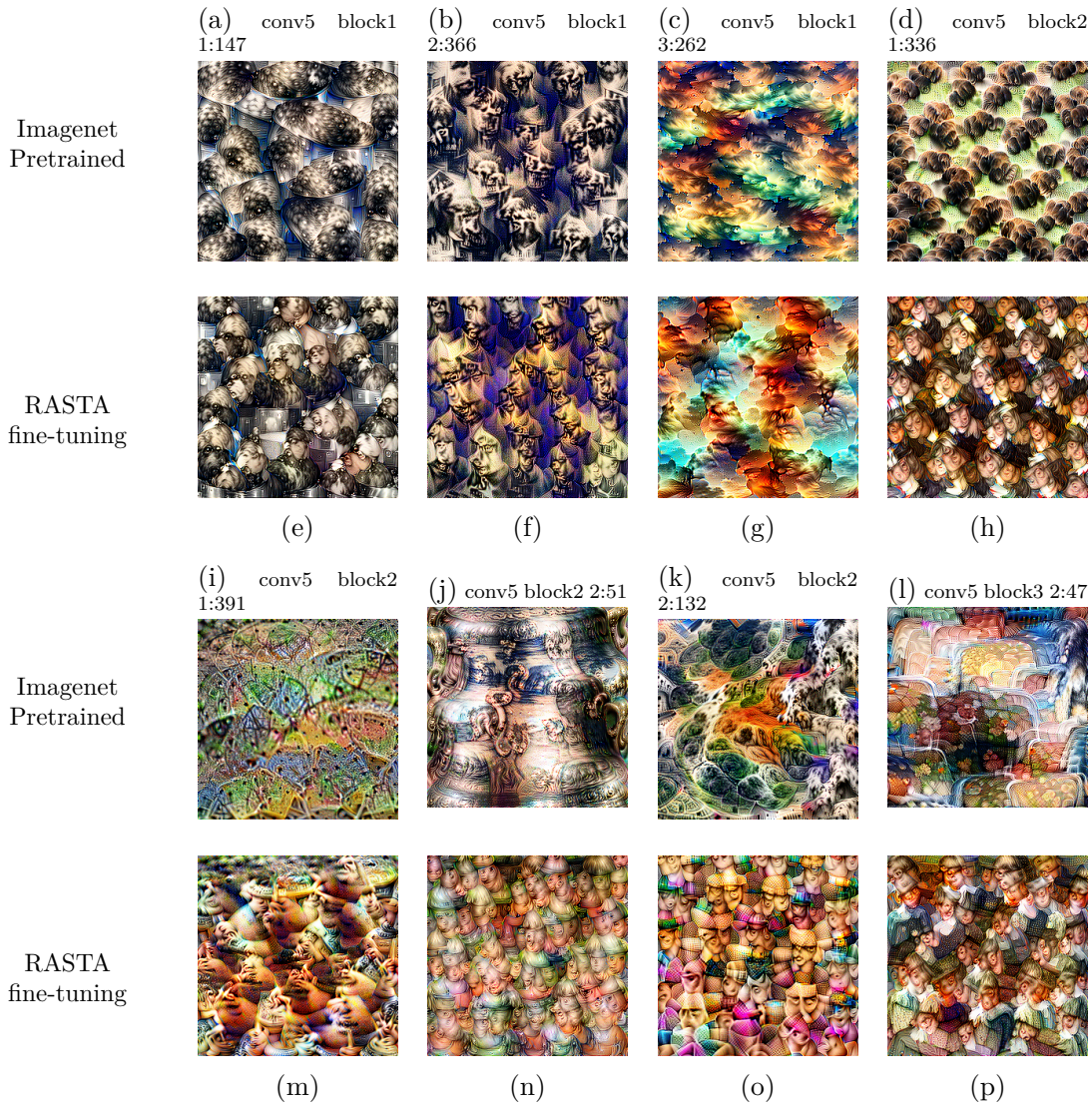


Figure B.7: Optimized Images for a ResNet50 fine-tuned for high level layers. First and third rows: optimized images from the model pretrained on ImageNet. Second and fourth ones: optimized images from the model fine-tuned on RASTA.

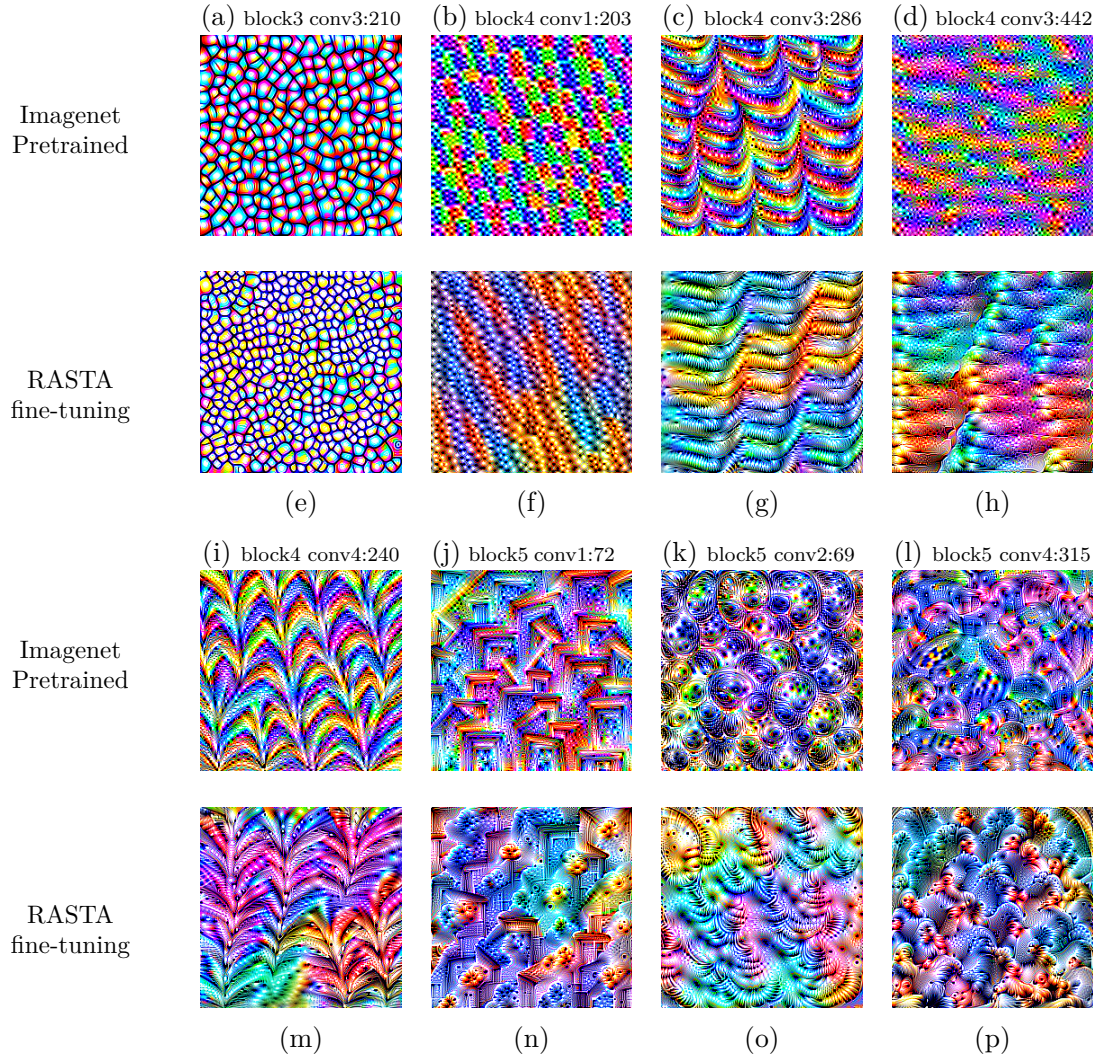


Figure B.8: Optimized Images for a VGG19 fine-tuned for mid and high level layers. First and third rows: optimized images from the model pretrained on ImageNet. Second and fourth ones: optimized images from the model fine-tuned on RASTA.

C

Additional Results for the Proposed Multiple Instance Model and its Variants

Contents

C.1 Detailed Results for the Sanity Check on PASCAL VOC 2007

Net	Method	aero	bicy	bird	boa	bot	bus	car	cat	cha	cow	dtab	dog	hors	mbik	pers	plnt	she	sofa	trai	tv	mean
RES-101-VOC07	FSD [He et al. 2015]	73.6	82.3	75.4	64.0	57.4	80.2	86.5	86.2	52.7	85.2	66.9	87.0	87.1	82.9	81.2	45.7	76.8	71.2	82.6	75.5	**75.0**
	FSD w/o B	68.3	63.3	71.7	54.1	54.4	68.8	84.6	79.6	46.9	82.5	46.8	80.4	78.0	69.5	74.6	40.4	75.0	64.5	74.7	70.4	67.4
	MAX [Crowley et al. 2016]	20.8	47.0	26.1	20.2	8.3	41.1	44.9	60.1	31.7	54.8	46.4	42.9	62.2	58.7	20.9	21.6	37.6	16.7	42.0	19.8	36.2
	MAXA	64.2	65.4	51.9	42.7	34.2	69.1	81.8	64.8	33.7	76.5	47.7	64.8	78.3	71.6	51.8	28.9	60.4	64.5	73.0	51.7	58.9
	MI-max w/o score	51.2	76.0	58.4	42.7	48.9	65.8	82.1	71.3	46.4	73.5	58.5	68.8	76.5	67.5	60.1	33.2	62.1	60.5	64.9	67.6	61.8 ± 0.7
	MI-max	63.5	78.4	68.5	54.0	50.7	71.8	85.6	77.1	52.7	80.0	60.1	78.3	80.5	73.5	74.7	37.4	71.2	65.2	75.7	67.7	68.3 ± 0.2
	Polyhedral MI-max w/o score	57.8	74.0	59.8	42.9	47.4	66.9	82.3	72.1	45.4	74.5	57.6	66.4	75.4	71.6	65.0	34.9	61.3	60.2	65.8	68.7	62.5 ± 0.9
	Polyhedral MI-max	65.6	79.6	70.4	51.3	52.4	72.1	85.1	75.3	53.0	83.2	61.1	74.4	80.1	76.4	75.5	39.5	72.5	70.1	73.4	70.0	69.0 ± 0.2
	MI-max-HL w/o score	57.8	76.7	56.0	44.6	47.4	70.1	77.1	71.8	44.2	74.8	53.1	64.0	74.8	66.9	48.5	30.9	63.9	58.3	69.8	64.3	60.7 ± 0.9
	MI-max-HL	65.7	79.9	71.5	50.3	52.2	75.6	83.8	79.0	51.4	80.4	59.3	79.3	82.6	76.1	75.5	35.4	72.2	68.1	74.9	70.2	**69.2** ± 0.4
RES-152-COCO	FSD [He et al. 2015]	91.0	90.4	88.3	61.2	77.7	92.2	82.2	93.2	67.0	89.4	65.8	88.0	92.0	89.5	88.5	56.9	85.1	81.0	89.8	85.2	**82.7**
	FSD w/o B	86.8	73.3	86.3	55.5	77.7	91.6	81.9	87.6	62.4	87.5	54.8	83.5	87.5	84.6	84.8	54.4	88.4	73.5	86.6	84.9	78.7
	MAX [Crowley et al. 2016]	58.8	64.7	52.4	8.6	20.8	55.2	66.8	76.1	19.4	66.3	6.7	59.7	56.4	43.3	15.5	18.3	80.3	7.6	71.8	32.6	44.1
	MAXA	85.9	62.3	76.7	58.2	41.3	84.6	88.1	81.7	41.4	83.3	7.2	75.8	71.2	72.5	58.5	33.0	76.7	61.4	86.5	62.8	65.5
	MI-max w/o score	82.4	88.8	78.5	58.5	76.3	91.8	88.7	88.7	62.6	74.4	56.7	80.8	86.0	82.9	1.2	51.6	78.1	79.5	86.5	84.3	73.9 ± 0.3
	MI-max	88.0	90.2	84.3	66.0	78.7	93.8	92.7	90.7	63.7	78.8	61.5	88.4	90.9	88.8	87.9	56.8	75.5	81.3	88.4	86.1	81.6 ± 0.3
	Polyhedral MI-max w/o score	83.6	86.3	79.4	57.7	77.0	91.2	90.7	86.5	67.1	76.1	64.8	81.4	86.0	83.8	40.8	47.3	85.8	80.7	84.9	82.8	76.7 ± 1.7
	Polyhedral MI-max	88.5	88.8	86.0	59.9	79.3	93.9	85.8	91.3	69.6	87.2	57.2	89.4	90.9	89.1	87.8	57.4	88.0	82.3	87.2	85.3	**82.3** ± 1.9
	MI-max-HL w/o score	79.5	83.1	71.1	39.9	66.3	88.1	86.9	84.0	56.7	72.9	47.0	71.9	84.3	83.0	60.5	42.9	77.7	71.7	80.8	72.3	71.0 ± 2.0
	MI-max-HL	86.7	86.4	85.1	44.4	73.9	93.1	93.5	91.4	64.4	68.0	58.3	87.4	90.4	88.7	85.6	51.5	80.2	76.1	85.3	85.1	78.8 ± 1.8

Table C.1: **VOC 2007 test** Average precision (%). Comparison of the Faster R-CNN detector (trained in a fully supervised manner: FSD) and our MI-max algorithm (trained in a weakly supervised manner) for two networks RES-101-VOC07 and RES-152-COCO. In blue the best fully supervised method and in red the best weakly supervised one. When it was computationally acceptable we calculated the standard deviation on 10 runs.

C.2 Detailed Detection Performances for Art Datasets

We report in Tables C.2 to C.6 the performances for the Weakly Supervised Object Detection task for 5 different non-photographic datasets: Watercolor2k, Clipart1k, Comic2k [Inoue et al. 2018], IconArt (Section 4.2.1) and "CASPA paintings" [Thomas et al. 2018]. "CASPA paintings" is the painting subset of the CASPA dataset[1] proposed in [Thomas et al. 2018] with bounding boxes associated to 8 visual categories (only animals) for most of the images.

We denote MI-max-S-C, the multiple instance perceptron with a naive selection of the hyperparameter C of the model, as defined in Section 5.3.3.4. The AP value for the detection task on the PeopleArt dataset with this method is 56.5 ± 2.2.

Net	Method	Model	bike	bird	car	cat	dog	person	mean
SSD	Mixed supervised with domain adaptation	DT+PL [Inoue et al. 2018]	76.5	54.9	46.0	37.4	38.5	72.3	**54.3**⋆
VGG16-IM	Weakly supervised fine-tuning	WSDDN [Bilen et al. 2016]	1.5	26.0	14.6	0.4	0.5	33.3	12.7
		SPN [Zhu et al. 2017b]	0.0	18.9	0.0	0.0	0.0	23.6	7.1
		PCL [Tang et al. 2018a]	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RES-152-COCO	Off-the-shelf Features extraction	MAX [Crowley et al. 2016]	76.0	33.8	33.0	20.8	22.7	19.8	34.3
		MAXA	60.6	39.2	39.6	30.9	32.0	61.2	43.9
		MI-SVM [Andrews et al. 2003]	66.8	20.9	7.6	14.1	8.5	13.2	21.8
		mi-SVM [Andrews et al. 2003]	10.6	10.9	1.4	2.0	0.8	5.9	5.3
		MI_Net [Wang et al. 2018b]	77.6	32.4	35.5	24.7	16.2	18.0	34.1 ± 1.0
		MI_Net_DS [Wang et al. 2018b]	73.4	22.4	25.8	17.6	11.2	10.3	26.8 ± 2.4
		MI_Net_RC [Wang et al. 2018b]	32.3	19.2	20.1	6.7	6.8	15.4	16.7 ± 6.3
		mi_Net [Wang et al. 2018b]	66.4	30.3	14.9	14.4	8.6	20.5	25.8 ± 3.5
		MI-max	84.1	47.4	48.2	30.9	27.9	58.2	**49.5** ± 0.9
		MI-max-S-C	78.2	46.1	45.6	27.5	30.0	53.7	46.8 ± 2.2
		Polyhedral MI-max	77.8	44.7	45.5	25.6	26.7	59.2	46.6 ± 1.3
		MI-max-HL	79.3	46.1	43.6	26.9	28.8	57.0	47.0 ± 1.6

Table C.2: **Watercolor2k (test set)** Detection Average Precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. The best mixed supervised method is highlighted in green and the best weakly supervised one in red. We use a grid search for MAX and MAXA. When it was computationally acceptable we calculated the standard deviation on 10 runs. DT+PL and WSDDN performances come from the original paper [Inoue et al. 2018].

[1]This dataset can be found here: http://people.cs.pitt.edu/~chris/artistic_objects/

Net	Method	Model	aero plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motor bike	person	potted plant	sheep	sofa	train	tv monitor	mean
SSD Yolov2 Faster RCNN	Mixed supervised with domain adaptation	DT+PL [Inoue et al. 2018]	35.7	61.9	26.2	45.9	29.9	74.0	48.7	2.8	53.0	72.7	50.2	19.3	40.9	83.3	62.4	42.4	22.8	38.5	49.3	59.5	46.0*
		DT+PL [Inoue et al. 2018]																					39.9*
		DT+PL [Inoue et al. 2018]																					34.9*
VGG16-IM	Weakly supervised fine-tuning	WSDDN [Bilen et al. 2016]	1.6	3.6	0.6	2.3	0.1	11.7	4.5	0.0	3.2	0.1	2.8	2.3	0.9	0.1	14.4	16.0	4.5	0.7	1.2	18.3	4.4
		SPN [Zhu et al. 2017b]	0.0	12.5	0.8	0.1	0.0	12.5	1.0	0.0	0.1	4.8	6.4	0.0	5.3	5.0	2.3	0.0	0.0	0.0	22.5	2.5	3.8
		PCL [Tang et al. 2018a]	0.4	0.0	0.3	1.1	0.1	0.0	5.9	0.0	0.9	0.0	0.3	3.8	0.3	0.0	3.6	1.5	0.0	0.7	0.0	4.4	1.2
RES-152-COCO	Off-the-shelf Features extraction	MAX [Crowley et al. 2016]	15.2	12.6	15.7	23.3	2.2	34.5	19.0	0.0	15.6	7.7	2.4	4.6	24.7	41.9	15.6	32.6	0.4	0.0	46.4	22.9	16.9
		MAXA	24.7	29.2	19.7	31.6	6.0	37.0	34.6	0.0	30.6	1.7	4.2	0.9	12.7	53.0	35.4	34.0	0.7	4.9	50.3	29.5	22.0
		MI-SVM [Andrews et al. 2003]	10.3	35.8	8.4	22.4	15.5	25.0	28.3	8.7	26.9	4.8	14.3	0.0	18.4	45.0	22.6	16.4	1.5	7.9	51.9	22.4	19.3
		mi-SVM [Andrews et al. 2003]	1.0	4.1	8.1	6.4	1.5	4.5	16.0	4.4	10.4	4.1	2.7	0.1	10.6	20.5	6.2	3.1	0.2	2.6	8.6	8.5	6.2
		MI_Net [Wang et al. 2018b]	21.3	45.6	26.8	22.2	37.4	47.6	42.8	18.4	40.0	28.1	21.7	4.3	24.8	24.3	27.9	22.2	7.2	29.7	47.0	53.9	29.7 ± 1.5
		MI_Net_DS [Wang et al. 2018b]	12.9	44.1	15.0	12.1	25.1	30.5	11.8	14.0	26.4	14.4	16.8	4.3	8.9	12.6	16.4	15.2	5.1	23.5	30.5	39.1	18.9 ± 2.4
		MI_Net_RC [Wang et al. 2018b]	1.6	2.0	0.2	0.0	0.6	0.1	3.2	0.4	0.6	0.6	0.1	0.0	0.5	0.3	2.2	1.9	0.3	0.6	2.3	0.0	0.9 ± 0.8
		mi_Net [Wang et al. 2018b]	20.0	43.6	28.7	23.9	36.3	50.4	43.2	20.2	43.6	34.3	25.7	3.9	22.1	25.2	30.3	9.7	5.3	28.0	41.3	55.2	29.5 ± 1.2
		MI-max	42.4	46.4	25.0	45.6	45.6	52.6	43.7	24.0	45.5	42.4	29.1	5.9	35.5	52.3	55.5	50.0	2.1	15.7	60.3	47.9	**38.4** ± 0.8
		MI-max-S-C	32.8	40.3	10.9	39.5	42.9	66.2	39.6	21.2	45.6	45.2	27.1	9.5	28.6	46.9	50.3	26.6	0.8	20.8	54.2	43.2	34.6 ± 1.9
		Polyhedral MI-max	32.6	36.3	15.7	27.8	32.6	52.8	42.3	7.1	41.5	20.8	14.4	2.0	30.5	57.6	54.7	32.9	1.7	10.2	58.1	38.4	30.5 ± 2.3
		MI-max-HL	31.8	46.6	25.5	31.3	45.1	41.6	43.1	8.6	46.9	33.9	8.7	3.7	29.8	43.5	54.4	51.9	2.7	14.6	48.6	47.7	33.0 ± 1.2

Table C.3: **Clipart1k (test set)** Detection Average Precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. We use a grid search for MAX and MAXA. The best mixed supervised method is highlighted in green and the best weakly supervised one in red. When it was computationally acceptable we calculated the standard deviation on 10 runs. DT+PL and WSDDN performances come from the original paper [Inoue et al. 2018].

Net	Method	Model	bike	bird	car	cat	dog	person	mean
SSD	Mixed supervised with	DT+PL	76.5	54.9	46.0	37.4	38.5	72.3	54.3*
VGG16-IM	domain adaptation	[Inoue et al. 2018]							
	Weakly supervised fine-tuning	WSDDN [Bilen et al. 2016]	1.5	26.0	14.6	0.4	0.5	33.3	12.7
		SPN [Zhu et al. 2017b]	0.0	0.0	0.0	3.1	0.0	4.1	1.2
		PCL [Tang et al. 2018a]	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RES-152-COCO	Off-the-shelf Features extraction	MAX[Crowley et al. 2016]	15.2	2.7	29.4	2.3	16.8	4.9	11.9
		MAXA	36.8	5.6	27.1	8.2	6.1	34.8	19.8
		MI-SVM [Andrews et al. 2003]	34.2	3.0	20.0	5.2	2.5	12.9	13.0
		mi-SVM [Andrews et al. 2003]	10.8	2.3	5.5	3.2	2.1	3.6	4.6
		MI_Net [Wang et al. 2018b]	42.9	15.5	33.1	11.8	13.4	20.4	22.8 ± 1.1
		MI_Net_DS [Wang et al. 2018b]	40.8	13.3	32.5	5.7	9.1	16.1	19.6 ± 1.6
		MI_Net_RC [Wang et al. 2018b]	19.8	5.4	16.4	2.8	9.8	13.9	11.4 ± 4.4
		mi_Net [Wang et al. 2018b]	42.1	10.9	24.5	8.8	8.8	22.1	19.5 ± 2.1
		MI-max	45.3	9.7	33.7	14.4	21.6	37.0	**27.0** ± 0.8
		MI-max-S-C	47.4	4.3	30.6	13.6	11.9	28.0	22.6 ± 1.2
		Polyhedral MI-max	44.9	5.2	26.2	14.1	11.0	38.4	23.3 ± 1.6
		MI-max-HL	43.0	5.1	31.5	11.8	13.8	36.4	23.6 ± 0.5

Table C.4: **Comic2k (test set)** Detection Average Precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. The best mixed supervised method is highlighted in green and the best weakly supervised one in red. When it was computationally acceptable we calculated the standard deviation on 10 runs. DT+PL and WSDDN performances come from the original paper [Inoue et al. 2018].

Net	Method	Model	bear	bird	cat	cow	dog	elephant	horse	sheep	mean
VGG16-IM	Weakly supervised fine-tuning	SPN [Zhu et al. 2017b]	0.5	0.1	1.6	0.9	0.5	1.4	0.6	0.0	0.7
		PCL [Tang et al. 2018a]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RES-152-COCO	Off-the-shelf Features extraction	MAX[Crowley et al. 2016]	22.0	2.1	14.5	3.5	14.2	8.8	12.8	0.5	9.8
		MAXA	26.3	13.1	26.9	5.4	8.3	18.1	14.9	3.9	14.6
		MI-SVM [Andrews et al. 2003]	9.3	0.2	6.7	1.5	0.1	0.6	0.9	0.4	2.5
		mi-SVM [Andrews et al. 2003]	1.3	1.6	3.0	0.8	1.0	0.3	1.5	0.3	1.2
		MI_Net [Wang et al. 2018b]	32.8	5.4	14.1	5.2	6.2	15.0	11.1	4.2	11.7 ± 1.6
		MI_Net_DS [Wang et al. 2018b]	29.0	1.6	8.3	3.0	3.2	5.9	7.1	2.6	7.6 ± 1.2
		MI_Net_RC [Wang et al. 2018b]	16.9	0.9	6.6	2.6	2.9	8.2	4.7	2.1	5.6 ± 2.1
		mi_Net [Wang et al. 2018b]	26.7	8.9	12.5	1.5	3.4	7.1	5.1	2.4	8.4 ± 1.7
		MI-max	28.3	15.7	25.6	5.3	13.7	17.2	18.8	5.1	16.2 ± 0.4
		MI-max-S-C	26.9	15.5	25.4	5.1	15.2	18.6	18.5	4.9	**16.3** ± 0.7
		Polyhedral MI-max	26.2	16.9	23.9	5.4	10.1	9.7	18.8	4.5	14.4 ± 0.7
		MI-max-HL	26.5	15.7	26.3	4.8	14.2	10.1	11.5	6.2	14.4 ± 0.9

Table C.5: **CASPA paintings (test set)** Detection Average Precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. We use a grid search for MAX and MAXA. When it was computationally acceptable we calculated the standard deviation on 10 runs.

Net	Method	Model	angel	JCchild	crucifixion	Mary	nudity	ruins	StSeb	mean
VGG16-IM	Weakly supervised fine-tuning	SPN [Zhu et al. 2017b]	0.0	0.8	22.3	12.0	6.8	10.4	1.2	7.7
		PCL [Tang et al. 2018a]	2.9	0.3	1.0	26.3	2.3	7.2	1.4	5.9
RES-152-COCO	Off-the-shelf Features extraction	MAX[Crowley et al. 2016]	1.4	1.3	11.5	2.8	3.8	0.3	4.5	3.7
		MAXA	1.3	4.4	18.2	28.0	15.3	0.2	16.4	12.0
		MI-SVM [Andrews et al. 2003]	0.7	4.4	21.6	0.6	1.0	0.0	0.0	4.0
		mi-SVM [Andrews et al. 2003]	1.3	5.1	3.9	3.6	2.9	0.3	2.2	2.8
		MI_Net [Wang et al. 2018b]	9.7	42.6	21.1	6.9	17.6	5.1	2.5	15.1 ± 1.5
		MI_Net_DS [Wang et al. 2018b]	8.6	35.6	19.6	5.3	15.9	3.2	3.1	13.0 ± 1.7
		MI_Net_RC [Wang et al. 2018b]	8.2	36.9	20.5	4.8	16.2	1.6	0.9	12.7 ± 1.6
		mi_Net [Wang et al. 2018b]	8.2	28.4	15.1	11.2	15.8	6.8	4.5	12.9 ± 1.2
		MI-max	0.3	0.1	42.7	4.4	21.9	0.6	13.7	12.0 ± 0.9
		MI-max-S-C	3.6	35.4	34.2	5.0	23.8	1.1	10.7	**16.3** ± 1.6
		Polyhedral MI-max	3.1	9.8	33.0	7.4	29.2	0.1	8.5	13.0 ± 2.2
		MI-max-HL	4.3	6.7	35.7	15.6	24.0	0.1	15.2	14.5 ± 1.8

Table C.6: **IconArt detection test set** Detection Average Precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. We use a grid search for MAX and MAXA. In red, the best weakly supervised method. When it was computationally acceptable we calculated the standard deviation on 10 runs.

C.3 Classification Score for Artistic Datasets

We report in Tables C.7 to C.12 the classification performances obtained with weakly supervised detectors for 6 different non-photographic datasets : PeopleArt [Westlake et al. 2016], Watercolor2k, Clipart1k, Comic2k [Inoue et al. 2018], IconArt (Section 4.2.1) and CASPApaintings [Thomas et al. 2018]. Theses tables have been shortly commented in Section 5.3.4. We compare the five variants of our models (defined in Section 5.2): namely MI-max, Polyhedral MI-max, MI-max w/o score, Polyhedral MI-max w/o score and MI-max-HL with alternative solutions. They are MI_Net, MI_Net_DS, MI_Net_RC and mi_Net from Wang et al. [2018b], mi-SVM and MI-SVM from Andrews et al. [2003], MAX [Crowley et al. 2016] and MAXA a variant of the latter. These methods have been described in Section 5.3.2.3. The MI_Net model provides the best classification performances on the six datasets. Generally, the MI_Net and mi_Net models outperforms the MI_Net_DS, MI_Net_RC ones. Surprisingly, the MAXA approach provides scores comparable to the MAX one, whereas it outperforms it on the detection task. Then, the use of score is beneficial for the Polyhedral MI-max but not the MI-max one. Finally, the MI-max-HL under performs on the classification task.

The results on the IconArt dataset in Table C.12 should be compared to the off-the-shelf classification CNN (Table B.1) and the other transfer learning methods (Table 4.7).

Network	Method	Model	person
RES-152-COCO	Features extraction	MAX [Crowley et al. 2016]	93.3
		MAXA	92.1
		MI-SVM [Andrews et al. 2003]	91.1
		mi-SVM [Andrews et al. 2003]	91.8
		MI_Net [Wang et al. 2018b]	**94.4** ± 0.1
		MI_Net_DS [Wang et al. 2018b]	90.9 ± 3.5
		MI_Net_RC [Wang et al. 2018b]	74.9 ± 23.4
		mi_Net [Wang et al. 2018b]	93.8 ± 0.2
		MI-max	92.6 ± 0.5
		MI-max w/o score	92.5 ± 0.3
		MI-max-S-C	92.8 ± 0.5
		Polyhedral MI-max	93.9 ± 0.2
		Polyhedral MI-max w/o score	89.7 ± 3.4
		MI-max-HL	93.0 ± 0.1

Table C.7: **People-Art (test set)** Classification Average precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. We use a grid search on the hyperparameter of MAX and MAXA. Standard deviation computed on 10 runs of the algorithm.

Net	Method	Model	bike	bird	car	cat	dog	person	mean
RES-152-COCO	Off-the-shelf Features extraction	MAX[Crowley et al. 2016]	83.9	79.8	93.6	44.1	52.6	97.8	75.3
		MAXA	84.1	82.3	84.4	49.4	49.6	98.0	74.6
		MI-SVM [Andrews et al. 2003]	91.3	83.0	79.8	54.8	36.7	98.6	74.0
		mi-SVM [Andrews et al. 2003]	51.2	74.5	71.0	32.1	23.6	96.2	58.1
		MI_Net [Wang et al. 2018b]	90.9	84.6	93.5	56.7	42.7	98.8	**77.9** ± 0.5
		MI_Net_DS [Wang et al. 2018b]	91.0	81.6	83.2	48.7	41.1	95.0	73.4 ± 4.8
		MI_Net_RC [Wang et al. 2018b]	37.5	68.9	63.9	26.6	31.1	94.4	53.7 ± 11.8
		mi_Net [Wang et al. 2018b]	82.7	85.4	89.8	52.9	41.0	98.8	75.1 ± 4.1
		MI-max	90.2	85.1	92.4	52.3	47.2	98.6	77.7 ± 0.4
		MI-max w/o score	80.9	52.4	62.4	37.6	29.2	53.0 5	52.6 ± 1.7
		MI-max-S-C	85.6	82.8	92.1	49.3	48.5	98.2	76.1 ± 1.5
		Polyhedral MI-max	88.0	85.1	94.1	46.0	43.3	98.5	75.8 ± 1.7
		Polyhedral MI-max w/o score	25.9	70.3	88.0	21.4	23.8	96.8	54.3 ± 5.5
		MI-max-HL	90.9	84.4	93.1	48.8	43.2	98.0	76.4 ± 1.1

Table C.8: **Watercolor2k (test set)** Classification Average precision . Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. We use a grid search on the hyperparameter of MAX and MAXA. Standard deviation computed on 10 runs of the algorithm.

Net	Method	Model	aero plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motor bike	person	potted plant	sheep	sofa	train	tv monitor	mean
RES-152-COCO	Off-the-shelf Features extraction	MAX[Crowley et al. 2016]	33.2	51.9	49.8	58.5	48.4	37.7	65.9	4.7	68.8	38.0	44.7	22.3	44.6	44.2	90.6	76.0	13.0	27.0	73.6	66.4	48.0
		MAXA	40.2	70.0	51.0	68.9	42.7	36.1	72.2	6.2	68.8	32.5	49.0	10.1	48.2	64.1	92.2	56.7	20.8	35.2	79.8	64.7	50.5
		MI-SVM [Andrews et al. 2003]	48.7	75.9	66.9	80.2	46.9	48.5	72.7	27.3	76.0	56.7	58.6	9.2	56.6	66.5	94.6	62.5	25.5	31.0	85.7	65.7	57.8
		mi-SVM [Andrews et al. 2003]	27.3	26.9	52.7	65.9	21.1	16.0	57.7	14.5	72.4	35.7	39.7	10.1	41.4	55.7	92.5	41.5	19.4	19.9	48.5	49.3	40.4
		MI_Net [Wang et al. 2018b]	61.7	78.3	68.7	78.9	64.2	58.0	82.2	33.2	80.1	67.1	61.9	18.6	61.2	65.9	96.0	78.7	29.9	52.0	76.5	81.3	**64.7** ± 0.8
		MI_Net_DS [Wang et al. 2018b]	50.0	76.7	60.6	72.5	54.6	58.4	60.8	30.8	72.1	60.8	60.6	18.9	51.9	64.8	87.8	76.3	29.9	51.0	64.7	79.7	59.2 ± 2.3
		MI_Net_RC [Wang et al. 2018b]	17.2	9.5	18.5	11.1	13.7	4.4	23.7	6.0	25.8	9.1	14.7	5.8	12.1	3.6	58.9	27.3	12.2	10.1	12.7	9.9	15.3 ± 1.6
		mi_Net [Wang et al. 2018b]	59.1	78.9	68.5	79.5	63.0	61.6	81.4	35.2	80.5	69.2	62.1	18.5	63.1	63.2	96.2	77.6	23.1	51.8	74.7	81.1	64.4 ± 0.6
		MI-max	63.6	84.7	60.5	76.4	61.6	51.8	76.8	31.2	78.6	58.9	62.3	19.0	56.2	52.9	95.0	79.1	21.7	38.4	82.5	74.0	61.3 ± 0.8
		MI-max w/o score	60.0	81.7	59.7	71.0	58.6	49.6	77.6	25.3	79.8	45.3	62.1	15.3	47.8	58.4	95.7	81.7	22.1	42.8	77.5	75.4	59.4 ± 1.7
		MI-max-S-C	55.4	76.7	45.4	75.3	59.3	67.2	72.0	31.0	77.6	54.3	59.1	28.0	46.5	52.8	95.0	75.8	15.5	45.0	75.9	69.1	58.8 ± 2.8
		Polyhedral MI-max	53.6	75.0	52.5	61.5	50.0	54.7	77.3	14.7	76.0	40.0	54.5	12.8	51.4	59.8	94.8	75.4	20.5	30.0	78.1	65.5	54.9 ± 2.0
		Polyhedral MI-max w/o score	22.4	46.7	33.8	29.0	28.3	24.1	70.3	7.5	66.4	13.5	44.2	8.2	24.2	11.8	90.8	66.6	5.3	16.0	33.7	37.0	34.0 ± 2.7
		MI-max-HL	54.0	79.0	59.6	64.9	62.1	43.1	75.5	18.0	75.6	47.9	55.2	17.2	46.3	45.8	93.3	75.4	18.5	34.7	70.0	71.1	55.4 ± 0.9

Table C.9: **Clipart1k (test set)** Classification Average precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. We use a grid search on the hyperparameter of MAX and MAXA. Standard deviation computed on 10 runs of the algorithm.

Net	Method	Model	bike	bird	car	cat	dog	person	mean
RES-152-COCO	Off-the-shelf Features extraction	MAX [Crowley et al. 2016]	53.5	25.0	53.3	16.6	34.1	97.4	46.6
		MAXA	54.8	27.0	53.8	23.1	20.0	97.4	46.0
		MI-SVM [Andrews et al. 2003]	54.9	28.1	48.8	40.4	26.4	97.2	49.3
		mi-SVM [Andrews et al. 2003]	44.2	24.2	25.4	23.1	20.1	95.8	38.8
		MI_Net [Wang et al. 2018b]	61.6	37.9	56.6	46.9	38.0	98.2	**56.5** ± 0.9
		MI_Net_DS [Wang et al. 2018b]	61.1	34.0	55.8	44.1	32.1	98.0	54.2 ± 1.7
		MI_Net_RC [Wang et al. 2018b]	38.2	21.3	32.0	22.2	31.6	96.5	40.3 ± 8.1
		mi_Net [Wang et al. 2018b]	62.5	32.6	52.5	47.9	33.4	98.3	54.5 ± 2.3
		MI-max	58.7	34.2	53.6	41.6	40.0	97.8	54.3 ± 0.7
		MI-max w/o score	63.4	36.2	52.8	43.1	36.3	97.8	54.9 ± 2.0
		MI-max-S-C	61.1	25.9	46.4	38.3	25.6	97.5	49.1 ± 1.3
		Polyhedral MI-max	59.4	25.1	42.8	41.0	27.9	97.7	49.0 ± 2.3
		Polyhedral MI-max w/o score	47.8	9.0	22.4	17.2	11.9	96.1	34.1 ± 2.7
		MI-max-HL	56.9	24.0	51.9	30.5	29.8	97.4	48.4 ± 1.1

Table C.10: **Comic2k (test set)** Classification Average precision. Comparison of the proposed MI-max method to alternative approaches. We use a grid search on the hyperparameter of MAX and MAXA. Standard deviation computed on 10 runs of the algorithm.

Net	Method	Model	bear	bird	cat	cow	dog	elephant	horse	sheep	mean
RES-152-COCO	Off-the-shelf Features extraction	MAX [Crowley et al. 2016]	51.9	51.5	65.9	48.0	36.8	38.7	68.2	26.7	48.5
		MAXA	50.0	49.5	70.4	50.1	28.2	38.2	55.9	32.4	46.8
		MI-SVM [Andrews et al. 2003]	52.7	48.5	69.5	52.3	23.5	28.0	49.1	48.8	46.6
		mi-SVM [Andrews et al. 2003]	18.9	51.4	54.4	42.3	28.6	10.3	45.1	40.0	36.4
		MI_Net [Wang et al. 2018b]	63.3	38.8	75.8	58.5	32.9	40.6	66.9	54.8	**54.0** ± 1.6
		MI_Net_DS [Wang et al. 2018b]	61.2	29.3	60.7	49.2	29.7	27.8	47.5	42.3	43.5 ± 4.6
		MI_Net_RC [Wang et al. 2018b]	37.3	24.5	55.5	44.4	26.3	27.9	44.1	37.2	37.1 ± 3.7
		mi_Net [Wang et al. 2018b]	66.0	58.0	77.5	60.6	34.8	35.6	64.8	52.3	56.2 ± 1.0
		MI-max	51.8	53.1	64.5	51.6	37.0	33.0	52.5	45.9	48.7 ± 0.5
		MI-max w/o score	54.0	53.0	64.7	52.0	33.7	31.5	53.3	46.3	48.6 ± 0.6
		MI-max-S-C	52.8	53.9	63.4	52.1	36.8	33.1	52.2	46.6	48.9 ± 0.7
		Polyhedral MI-max	51.9	55.4	61.5	52.2	33.2	25.5	52.7	47.1	47.5 ± 0.9
		Polyhedral MI-max w/o score	37.1	48.4	52.3	45.7	25.9	9.1	46.4	36.9	37.7 ± 2.7
		MI-max-HL	53.9	53.4	60.9	48.4	37.2	26.5	39.9	43.3	45.4 ± 1.7

Table C.11: **CASPA paintings (test set)** Classification Average precision (%). Comparison of the proposed MI-max method to alternative approaches. We use a grid search on the hyperparameter of MAX and MAXA. Standard deviation computed on 10 runs of the algorithm.

Net	Method	Model	angel	JCchild	crucifixion	Mary	nudity	ruins	StSeb	mean
RES-152-COCO	Off-the-shelf Features extraction	MAX [Crowley et al. 2016]	49.3	74.7	30.3	67.5	57.4	43.2	7.0	47.1
		MAXA	47.2	73.2	48.5	78.5	62.3	44.8	14.3	52.7
		MI-SVM [Andrews et al. 2003]	49.1	63.5	72.6	76.3	62.5	52.5	6.9	54.8
		mi-SVM [Andrews et al. 2003]	63.5	76.9	78.6	81.0	72.0	62.2	29.0	66.2
		MI_Net [Wang et al. 2018b]	70.4	83.9	86.2	83.9	78.9	54.4	26.1	**69.1** ± 0.3
		MI_Net_DS [Wang et al. 2018b]	68.0	82.0	86.2	81.8	78.2	51.6	26.6	67.8 ± 0.2
		MI_Net_RC [Wang et al. 2018b]	67.7	80.3	85.1	77.2	76.8	47.8	23.2	65.4 ± 2.0
		mi_Net [Wang et al. 2018b]	70.6	83.6	87.1	85.6	78.1	72.0	28.9	72.3 ± 0.6
		MI-max	57.2	59.3	79.6	70.3	65.7	47.2	17.4	56.7 ± 1.0
		MI-max w/o score	61.0	64.2	83.4	70.5	67.4	57.8	18.7	60.4 ± 1.1
		MI-max-S-C	62.4	75.1	82.0	71.6	67.0	51.4	15.1	60.7 ± 1.3
		Polyhedral MI-max	65.6	75.1	84.0	75.0	71.4	42.9	19.0	61.9 ± 0.8
		Polyhedral MI-max w/o score	51.0	54.9	82.2	65.0	63.5	44.7	11.6	53.3 ± 3.5
		MI-max-HL	52.8	65.1	74.7	69.2	63.9	17.7	14.3	51.1 ± 2.4

Table C.12: **IconArt classification test set** Classification Average precision (%). Comparison of the proposed MI-max, Polyhedral MI-max and MI-max-HL methods to alternative approaches. We use a grid search on the hyperparameter of MAX and MAXA. Standard deviation computed on 10 runs of the algorithm.

C.4 Instance-level Classification on Birds, Newsgroups and SIVAL

This section contains the Tables C.13 to C.15 reporting all results on individual sets from **Birds** [Briggs et al. 2012], **Newsgroups** [Craven et al. 2008] and **SIVAL** [Rahmani et al. 2005]. Results in bold are best results and results with no statistically significant difference from the best ($\alpha = 0.05$). The results in bold may differ from [Carbonneau et al. 2016a] before we used a two-sided T-test only based on the mean and standard deviation of the performances. We show AUC, UAR and F1-score metrics. The performances of mi-SVM, MI-SVM, EM-DD, SI-SVM, MIL-Boost and RSIS-EoSVM methods come from the supplementary materials of Carbonneau et al. [2016a].

Method	mi-SVM [Andrews et al. 2003]	MI-SVM [Andrews et al. 2003]	EM-DD [Zhang et al. 2002]	SI-SVM	MIL-Boost [Viola et al. 2005]	RSIS-EoSVM [Carbonneau et al. 2016b]	Polyhedral MI-max	MI-max-HL	MI-max
Birds: UAR	4	2	1	12	1	2	1	7	4
Brown Creeper	**72.00** (0.90)	58.40 (1.90)	65.30 (1.10)	**72.00** (1.10)	54.30 (1.40)	50.10 (0.50)	57.72 (11.10)	**72.46** (7.24)	**66.53** (5.31)
Winter Wren	65.60 (1.00)	56.90 (0.70)	58.70 (0.50)	**70.10** (1.30)	56.50 (0.40)	58.90 (0.80)	53.92 (4.15)	58.64 (3.75)	57.37 (1.48)
Pacific-slope Flycatcher	78.80 (0.60)	57.10 (2.70)	71.20 (1.50)	**81.40** (0.90)	50.00 (0.00)	66.90 (2.70)	52.00 (6.53)	74.97 (8.40)	50.36 (5.53)
Red-breasted Nuthatch	83.60 (1.70)	60.50 (2.00)	71.80 (1.70)	**85.10** (1.30)	62.30 (1.10)	76.30 (1.00)	57.09 (13.52)	77.74 (8.02)	75.67 (4.30)
Dark-eyed Junco	50.00 (0.00)	**55.40** (1.70)	50.00 (0.00)	50.80 (0.80)	50.00 (0.00)	50.00 (0.00)	49.92 (1.89)	**51.80** (8.77)	49.11 (2.21)
Olive-sided Flycatcher	63.80 (2.30)	61.00 (1.70)	66.90 (0.80)	**75.10** (2.00)	50.00 (0.00)	54.00 (1.00)	51.74 (6.17)	**66.26** (12.32)	48.83 (2.56)
Hermit Thrush	**50.00** (0.00)	**50.00** (0.00)	**50.00** (0.00)	**50.00** (0.00)	**50.00** (0.00)	**50.00** (0.00)	**51.72** (6.94)	**52.16** (8.72)	**57.01** (12.07)
Chestnut-backed Chickadee	67.10 (5.00)	66.20 (0.80)	66.70 (2.20)	**78.30** (1.20)	50.90 (0.30)	**79.10** (2.30)	52.35 (8.67)	**79.01** (9.11)	73.79 (6.22)
Varied Thrush	**92.00** (1.20)	58.20 (5.80)	80.90 (1.90)	**92.20** (1.60)	60.00 (1.90)	49.10 (0.10)	73.15 (23.37)	**91.96** (9.34)	**92.23** (4.71)
Hermit Warbler	64.30 (3.30)	52.90 (1.30)	58.10 (3.10)	**76.80** (2.20)	49.80 (0.00)	50.00 (0.00)	49.61 (1.33)	49.65 (5.70)	49.58 (1.32)
Swainson Thrush	**79.70** (1.90)	76.80 (2.10)	71.60 (1.60)	**82.00** (1.50)	49.90 (0.00)	49.90 (0.00)	59.46 (15.80)	**82.25** (7.21)	**79.95** (5.90)
Hammonds Flycatcher	92.20 (0.40)	55.10 (2.40)	70.60 (0.70)	**92.80** (0.20)	71.90 (0.70)	84.80 (1.00)	72.07 (16.05)	75.74 (8.26)	84.92 (2.05)
Western Tanager	81.70 (2.00)	73.60 (3.80)	71.50 (2.40)	**87.10** (2.50)	50.70 (0.80)	50.20 (0.50)	49.16 (5.58)	78.38 (11.13)	48.74 (1.28)
Birds: F1	4	4	7	4	1	3	1	5	4
Brown Creeper	27.90 (1.50)	25.10 (4.60)	**45.90** (2.50)	20.80 (1.00)	14.80 (4.40)	6.00 (1.00)	19.27 (12.29)	**46.07** (13.74)	**40.05** (11.50)
Winter Wren	38.20 (1.80)	23.70 (2.30)	29.60 (1.50)	**41.00** (2.10)	22.90 (1.40)	29.70 (2.20)	19.96 (6.41)	27.96 (7.68)	25.07 (4.13)
Pacific-slope Flycatcher	38.40 (1.10)	19.60 (5.90)	**55.40** (2.90)	31.10 (1.50)	0.00 (0.00)	37.10 (4.70)	10.11 (6.31)	40.82 (20.12)	3.61 (11.45)
Red-breasted Nuthatch	62.40 (2.10)	31.70 (5.10)	59.20 (3.70)	52.10 (1.30)	38.80 (2.70)	**66.60** (1.80)	18.23 (18.08)	51.74 (18.44)	50.56 (6.54)
Dark-eyed Junco	0.00 (0.00)	**17.60** (5.40)	0.00 (0.00)	1.80 (1.80)	0.00 (0.00)	0.00 (0.00)	1.54 (1.08)	4.15 (8.69)	0.72 (1.65)
Olive-sided Flycatcher	22.70 (3.60)	29.80 (3.60)	**46.20** (2.00)	22.90 (2.30)	0.00 (0.00)	13.20 (3.20)	5.93 (3.62)	20.64 (17.50)	1.03 (2.84)
Hermit Thrush	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**1.40** (3.12)	**2.58** (6.00)	**4.08** (5.96)
Chestnut-backed Chickadee	37.10 (10.50)	45.10 (1.60)	45.00 (4.90)	56.60 (2.40)	3.50 (1.20)	**63.70** (3.40)	9.16 (10.14)	41.86 (20.59)	43.71 (9.43)
Varied Thrush	31.70 (1.60)	18.50 (12.30)	**74.80** (3.10)	24.10 (1.30)	25.80 (4.20)	0.00 (0.00)	36.91 (35.28)	**67.81** (26.72)	**79.77** (10.70)
Hermit Warbler	**19.30** (4.10)	7.70 (2.80)	**20.50** (7.10)	**18.90** (1.30)	0.00 (0.00)	0.00 (0.00)	1.47 (1.54)	1.78 (6.27)	0.67 (2.06)
Swainson Thrush	41.50 (3.00)	**65.10** (3.70)	57.50 (3.10)	38.20 (1.80)	0.00 (0.00)	0.00 (0.00)	16.63 (22.33)	**61.46** (16.84)	**62.19** (9.43)
Hammonds Flycatcher	**88.40** (0.50)	16.10 (6.00)	58.20 (1.50)	**88.30** (0.50)	60.70 (1.40)	81.80 (1.30)	59.16 (26.63)	65.78 (14.77)	81.11 (2.95)
Western Tanager	**57.80** (3.30)	**59.10** (7.30)	**56.70** (4.60)	46.90 (3.40)	2.10 (2.10)	1.40 (1.40)	1.57 (3.22)	**47.29** (26.49)	0.10 (0.67)
Birds: AUC	0	0	7	9	0	3	4	6	4
Brown Creeper	79.50 (0.90)	76.80 (3.10)	**93.70** (0.60)	80.00 (0.90)	83.10 (2.00)	66.90 (1.40)	84.71 (6.51)	88.94 (2.76)	89.54 (4.92)
Winter Wren	69.40 (0.80)	66.80 (1.50)	62.00 (0.90)	**82.20** (0.90)	66.90 (1.50)	62.20 (0.50)	61.10 (5.95)	57.78 (3.73)	62.62 (3.14)
Pacific-slope Flycatcher	85.90 (1.20)	71.90 (3.20)	86.50 (1.20)	**88.10** (1.10)	52.90 (1.90)	**87.20** (1.40)	60.70 (17.82)	**87.30** (5.26)	49.15 (13.05)
Red-breasted Nuthatch	90.10 (1.50)	81.40 (2.80)	**98.30** (0.30)	91.60 (1.10)	96.10 (0.50)	84.70 (0.90)	79.76 (15.20)	92.32 (2.70)	94.57 (1.38)
Dark-eyed Junco	72.00 (3.10)	71.00 (3.40)	62.20 (5.10)	**77.70** (3.20)	48.30 (1.10)	59.30 (2.80)	53.80 (10.85)	52.52 (16.97)	60.68 (11.73)
Olive-sided Flycatcher	81.10 (1.80)	82.80 (4.10)	**94.50** (0.40)	86.50 (1.10)	46.60 (0.10)	81.60 (1.80)	67.75 (11.33)	75.17 (19.76)	49.97 (13.67)
Hermit Thrush	50.60 (6.80)	58.70 (5.00)	44.70 (2.20)	**71.90** (5.40)	55.10 (5.10)	**68.40** (3.20)	**62.71** (23.11)	**62.77** (20.19)	**68.62** (24.56)
Chestnut-backed Chickadee	86.20 (2.10)	82.70 (2.00)	**89.40** (2.30)	**89.50** (1.20)	83.90 (2.60)	**88.30** (2.00)	**77.12** (25.08)	**88.09** (4.35)	**88.69** (3.56)
Varied Thrush	96.70 (0.60)	47.30 (10.70)	**99.10** (0.50)	95.60 (1.50)	85.30 (2.40)	92.10 (1.10)	**93.46** (14.45)	**99.17** (1.63)	**99.11** (1.50)
Hermit Warbler	76.30 (3.90)	76.40 (2.00)	74.60 (2.40)	**89.20** (1.70)	55.60 (3.40)	75.80 (2.80)	51.40 (12.53)	34.85 (11.91)	63.72 (6.78)
Swainson Thrush	86.60 (2.10)	88.60 (1.80)	**90.30** (1.90)	**90.70** (1.10)	44.70 (1.80)	81.10 (1.80)	**82.25** (20.42)	**88.97** (5.95)	**89.11** (5.77)
Hammonds Flycatcher	96.00 (0.50)	94.80 (0.40)	90.90 (1.00)	**98.40** (0.20)	92.40 (0.70)	90.50 (0.40)	91.07 (2.16)	90.96 (1.83)	91.96 (1.75)
Western Tanager	91.30 (1.50)	92.10 (2.20)	**96.10** (1.30)	**95.00** (1.70)	62.90 (5.10)	51.80 (1.50)	40.88 (15.40)	**90.13** (12.47)	43.55 (7.77)

Table C.13: Detailed AUC, UAR and F1-score metrics results from the experiments on instance-level classification on **Birds dataset** [Briggs et al. 2012]. Each reported result is the average of 10-folds along with the standard error. Results in bold are best results and results with no statistically significant difference from the best ($\alpha = 0.05$). They come with the number of wins over the different classes for each metrics.

Method	mi-SVM [Andrews et al. 2003]	MI-SVM [Andrews et al. 2003]	EM-DD [Zhang et al. 2002]	SI-SVM	MIL-Boost [Viola et al. 2005]	RSIS-EoSVM [Carbonneau et al. 2016b]	Polyhedral MI-max	MI-max-HL	MI-max
Newsgroups UAR	12	11	1	3	0	9	16	8	10
alt.atheism	**84.10** (3.30)	67.00 (4.60)	51.60 (1.30)	**86.60** (2.10)	54.10 (1.50)	70.90 (4.10)	65.36 (19.00)	66.60 (3.52)	74.26 (9.73)
comp.graphics	**86.00** (2.70)	**86.80** (2.90)	49.40 (0.10)	79.30 (5.60)	52.10 (1.10)	81.40 (3.80)	70.72 (19.90)	58.21 (7.64)	71.02 (10.36)
comp.os.ms-windows.misc	**68.20** (2.50)	**69.80** (2.00)	52.70 (1.30)	58.00 (3.70)	50.00 (0.00)	**66.80** (4.90)	**63.07** (15.31)	58.08 (7.08)	63.90 (7.17)
comp.sys.ibm.pc.hardware	**76.40** (2.90)	**74.30** (4.70)	50.30 (0.70)	64.90 (4.30)	51.30 (0.90)	73.20 (2.50)	62.81 (16.97)	58.05 (6.99)	67.03 (9.93)
comp.sys.mac.hardware	75.40 (4.00)	**79.60** (3.10)	51.00 (1.10)	72.40 (6.10)	51.90 (1.00)	73.20 (3.40)	**68.76** (18.90)	59.28 (8.51)	69.29 (9.56)
comp.windows.x	**69.70** (4.70)	**73.30** (5.30)	51.60 (1.00)	66.90 (5.50)	60.80 (3.10)	**70.00** (3.00)	**72.05** (18.73)	**69.11** (12.32)	**68.55** (9.69)
misc.forsale	**71.40** (3.80)	66.60 (4.00)	52.50 (1.50)	**67.50** (4.50)	55.30 (2.00)	65.10 (2.10)	**63.42** (15.95)	57.80 (8.25)	64.19 (8.78)
rec.autos	61.60 (5.20)	56.20 (2.90)	51.80 (1.10)	61.50 (5.60)	52.20 (1.20)	**76.70** (3.90)	**67.59** (18.61)	61.78 (9.19)	69.53 (7.96)
rec.motorcycles	**84.20** (5.90)	69.20 (6.10)	58.70 (3.00)	65.90 (7.30)	51.40 (1.40)	76.20 (2.90)	**70.05** (20.29)	**75.39** (11.76)	74.93 (9.22)
rec.sport.baseball	**73.40** (4.70)	67.40 (3.70)	55.60 (1.90)	52.50 (3.70)	52.80 (1.20)	**73.30** (2.10)	**67.47** (18.75)	**70.14** (12.71)	**71.15** (10.77)
rec.sport.hockey	63.20 (6.70)	**78.90** (4.60)	52.80 (1.80)	58.20 (5.60)	75.40 (3.30)	**79.90** (3.20)	**79.45** (20.79)	**77.76** (12.31)	**78.10** (10.24)
sci.crypt	**68.80** (6.30)	52.80 (1.50)	**73.60** (2.30)	68.30 (6.30)	62.30 (2.60)	**75.90** (3.60)	**70.56** (19.72)	**77.93** (12.21)	**73.91** (9.36)
sci.electronics	84.10 (5.00)	78.00 (4.90)	49.60 (0.10)	77.90 (4.10)	52.20 (1.20)	**89.90** (0.90)	**78.10** (21.33)	53.29 (5.76)	71.02 (9.58)
sci.med	69.20 (6.50)	**80.20** (3.40)	52.50 (1.70)	71.00 (6.10)	53.00 (1.20)	73.40 (2.60)	66.81 (18.39)	70.08 (10.42)	**74.19** (8.77)
sci.space	54.30 (4.30)	**82.30** (2.60)	57.90 (2.60)	57.80 (4.70)	53.50 (1.60)	75.60 (2.20)	**68.98** (19.28)	70.66 (12.22)	73.43 (9.68)
soc.religion.christian	66.70 (5.40)	**77.10** (4.70)	57.20 (2.50)	48.50 (2.40)	59.60 (1.70)	72.10 (2.90)	**65.52** (19.44)	**73.00** (12.55)	**71.38** (10.92)
talk.politics.guns	**78.30** (3.60)	64.90 (6.40)	58.80 (4.40)	70.60 (7.00)	53.80 (1.30)	**75.90** (4.50)	**65.22** (19.78)	69.85 (11.14)	**73.62** (9.79)
talk.politics.mideast	**84.30** (1.60)	74.80 (4.40)	62.60 (2.70)	**85.30** (4.50)	78.70 (2.80)	78.80 (2.80)	**74.56** (20.54)	**75.88** (13.41)	**78.43** (10.05)
talk.politics.misc	**72.80** (3.30)	**73.90** (5.00)	52.00 (1.20)	58.30 (4.30)	58.50 (2.20)	**74.50** (2.70)	**65.06** (18.34)	**68.63** (12.03)	**69.96** (9.33)
talk.religion.misc	54.80 (3.30)	**71.30** (4.60)	51.20 (1.00)	56.80 (3.90)	52.40 (1.30)	67.20 (3.30)	**63.01** (16.21)	60.95 (8.93)	**65.22** (8.24)
Newsgroups F1	10	10	1	0	0	16	2	9	8
alt.atheism	**64.70** (5.50)	40.70 (8.60)	4.00 (2.70)	43.50 (8.10)	14.10 (4.90)	50.40 (8.90)	25.74 (28.30)	43.76 (4.90)	52.13 (16.98)
comp.graphics	**70.10** (5.10)	**74.00** (2.90)	0.00 (0.00)	56.60 (8.80)	6.90 (3.60)	**72.30** (6.50)	37.70 (31.99)	23.71 (20.16)	47.38 (17.69)
comp.os.ms-windows.misc	**44.10** (5.60)	**44.80** (3.70)	7.20 (3.00)	20.00 (8.90)	0.00 (0.00)	**41.40** (9.40)	21.73 (22.05)	22.15 (17.53)	33.37 (14.16)
comp.sys.ibm.pc.hardware	**59.90** (4.20)	**55.50** (9.30)	2.00 (2.00)	28.30 (7.20)	4.20 (2.80)	**60.20** (5.20)	20.34 (23.64)	23.16 (17.33)	37.15 (19.71)
comp.sys.mac.hardware	**59.00** (8.10)	**64.40** (4.00)	4.70 (3.20)	28.10 (7.80)	6.90 (3.50)	**59.70** (6.00)	33.15 (30.28)	26.15 (21.64)	44.70 (18.87)
comp.windows.x	**48.20** (9.40)	**50.10** (9.80)	5.10 (2.70)	35.20 (9.50)	29.90 (7.70)	**52.10** (5.80)	**40.61** (31.00)	**45.70** (26.14)	**46.20** (20.71)
misc.forsale	**48.20** (7.10)	39.70 (8.60)	7.50 (3.90)	35.40 (8.70)	17.10 (6.30)	**44.20** (5.40)	18.49 (19.26)	20.29 (19.16)	31.27 (17.20)
rec.autos	23.30 (8.20)	17.70 (7.80)	5.70 (2.90)	22.50 (8.00)	7.10 (3.70)	**62.20** (8.10)	30.90 (28.51)	32.21 (22.66)	44.56 (15.19)
rec.motorcycles	49.20 (8.50)	39.10 (10.80)	19.80 (5.50)	17.70 (7.20)	4.00 (4.00)	**60.20** (5.30)	32.67 (30.23)	**58.97** (20.55)	**55.12** (17.25)
rec.sport.baseball	**52.90** (9.50)	43.60 (8.70)	16.00 (4.80)	5.70 (1.90)	9.90 (4.10)	**58.80** (3.70)	29.38 (27.47)	**49.17** (25.59)	**48.66** (19.83)
rec.sport.hockey	29.80 (11.80)	**66.70** (8.50)	9.20 (3.90)	16.50 (6.80)	59.80 (5.70)	**72.70** (5.90)	**55.72** (35.48)	**66.44** (21.44)	**66.59** (18.00)
sci.crypt	35.50 (11.20)	8.70 (4.50)	**53.70** (3.70)	23.30 (6.80)	37.00 (6.60)	**60.50** (7.70)	31.31 (27.55)	**62.06** (21.40)	**51.00** (18.18)
sci.electronics	69.00 (9.20)	63.60 (9.00)	0.00 (0.00)	63.20 (8.10)	7.60 (3.90)	**87.10** (1.40)	54.15 (37.77)	10.19 (16.28)	52.03 (18.92)
sci.med	32.70 (9.70)	**66.70** (6.50)	6.40 (3.70)	31.90 (7.50)	8.70 (3.60)	60.70 (4.80)	28.38 (26.17)	50.12 (21.12)	53.37 (15.74)
sci.space	11.20 (7.40)	**70.60** (3.50)	19.80 (6.00)	8.60 (3.20)	11.30 (4.90)	64.30 (3.70)	29.96 (26.74)	49.94 (22.55)	51.25 (17.24)
soc.religion.christian	39.00 (10.70)	**60.40** (9.20)	17.30 (5.80)	2.90 (0.20)	30.90 (4.90)	**56.20** (5.10)	25.80 (29.15)	**52.87** (24.55)	**46.79** (21.19)
talk.politics.guns	**61.60** (5.70)	28.50 (10.00)	21.90 (9.20)	33.10 (12.00)	12.90 (4.30)	**57.10** (8.30)	24.65 (27.30)	**48.22** (22.48)	**49.08** (17.20)
talk.politics.mideast	**74.40** (2.40)	59.40 (7.10)	35.00 (7.20)	66.10 (5.30)	70.40 (4.80)	69.10 (4.40)	42.30 (32.09)	**61.13** (25.88)	**63.29** (17.64)
talk.politics.misc	53.10 (5.30)	46.80 (9.30)	5.70 (3.20)	18.30 (9.50)	26.10 (6.50)	**60.50** (3.70)	21.35 (22.79)	**43.97** (24.60)	43.67 (16.80)
talk.religion.misc	11.10 (5.60)	**42.60** (7.00)	3.30 (2.30)	6.50 (2.50)	8.40 (4.40)	**45.00** (6.00)	19.33 (19.85)	28.62 (20.66)	32.90 (15.68)
Newsgroups AUC	8	16	0	12	0	1	14	5	2
alt.atheism	**96.50** (1.80)	**94.50** (3.10)	17.30 (3.60)	93.00 (2.90)	54.10 (1.50)	92.40 (2.30)	85.84 (12.50)	80.91 (3.94)	81.53 (13.44)
comp.graphics	**89.30** (1.60)	**87.40** (3.60)	9.50 (2.70)	**88.90** (3.80)	52.10 (1.10)	**87.40** (2.40)	**85.40** (12.61)	69.77 (16.69)	62.27 (18.16)
comp.os.ms-windows.misc	81.80 (4.20)	85.50 (2.60)	32.90 (5.50)	**91.40** (2.60)	50.00 (0.00)	77.70 (4.10)	72.61 (17.80)	68.71 (12.67)	65.25 (14.62)
comp.sys.ibm.pc.hardware	**91.50** (2.70)	88.90 (4.00)	15.70 (3.40)	**92.90** (2.90)	51.30 (0.90)	87.50 (2.20)	82.22 (13.14)	75.36 (13.70)	58.73 (18.00)
comp.sys.mac.hardware	84.70 (3.20)	**90.20** (4.00)	21.40 (5.00)	**91.50** (3.40)	51.90 (1.00)	85.30 (2.90)	**85.63** (12.59)	78.64 (12.09)	64.91 (16.69)
comp.windows.x	93.50 (2.10)	94.00 (2.90)	22.70 (3.20)	**97.10** (1.10)	60.80 (3.10)	92.10 (2.10)	**89.97** (10.27)	85.27 (9.31)	78.73 (14.75)
misc.forsale	**89.80** (3.90)	**90.50** (2.50)	34.80 (8.70)	**92.10** (3.30)	55.30 (2.00)	76.00 (2.80)	**81.47** (15.04)	72.75 (11.94)	74.70 (14.46)
rec.autos	76.10 (7.40)	**97.30** (1.10)	19.00 (6.40)	73.70 (9.50)	52.20 (1.20)	84.30 (3.50)	82.28 (15.47)	82.49 (12.37)	70.64 (16.41)
rec.motorcycles	92.00 (3.80)	**97.80** (1.20)	27.20 (5.20)	88.40 (3.60)	51.40 (1.40)	90.00 (3.00)	**92.19** (11.24)	89.03 (10.05)	88.43 (12.39)
rec.sport.baseball	91.40 (3.50)	**97.60** (1.30)	19.30 (4.60)	78.10 (5.20)	52.80 (1.20)	88.80 (2.10)	**89.69** (11.50)	89.09 (10.04)	85.46 (14.27)
rec.sport.hockey	71.60 (9.50)	**99.00** (0.40)	25.60 (6.60)	67.20 (8.20)	83.20 (4.30)	93.80 (1.80)	**97.88** (4.78)	**94.90** (6.38)	**92.16** (11.38)
sci.crypt	96.30 (2.00)	**98.10** (1.30)	63.40 (6.40)	**98.70** (0.50)	62.30 (2.60)	91.90 (3.80)	**92.69** (11.92)	93.33 (7.32)	89.24 (11.07)
sci.electronics	**94.70** (2.30)	**96.50** (2.00)	7.60 (3.00)	**96.20** (2.10)	52.20 (1.20)	93.80 (1.40)	**90.33** (10.52)	**91.48** (11.14)	50.07 (20.08)
sci.med	**90.90** (4.00)	**92.70** (2.30)	24.90 (5.40)	**89.50** (4.20)	53.00 (1.20)	89.30 (2.80)	**88.10** (11.63)	**85.85** (9.67)	82.18 (11.83)
sci.space	77.10 (5.70)	**97.20** (0.80)	29.40 (6.70)	72.60 (3.90)	53.50 (1.60)	90.80 (2.30)	**89.13** (11.49)	89.87 (7.91)	87.35 (11.63)
soc.religion.christian	88.70 (6.90)	**95.10** (2.80)	39.10 (8.20)	83.90 (4.50)	59.60 (1.70)	87.20 (3.70)	**89.26** (12.36)	**90.49** (8.45)	78.94 (17.86)
talk.politics.guns	**97.60** (0.80)	**97.10** (1.20)	34.60 (7.50)	**96.20** (2.10)	53.80 (1.30)	92.50 (2.60)	**89.11** (14.66)	80.92 (11.76)	82.33 (14.31)
talk.politics.mideast	**96.70** (2.00)	**95.40** (3.00)	44.50 (5.20)	**97.30** (1.80)	78.70 (2.80)	92.90 (1.80)	**93.12** (8.41)	**93.96** (7.60)	**90.36** (11.65)
talk.politics.misc	93.40 (2.20)	93.90 (2.10)	12.30 (5.40)	**95.90** (1.90)	58.50 (2.20)	91.30 (2.50)	82.77 (13.68)	82.93 (12.19)	73.91 (16.65)
talk.religion.misc	72.10 (6.60)	**93.70** (1.90)	23.30 (4.00)	68.70 (5.90)	52.40 (1.30)	87.20 (4.00)	79.95 (14.45)	77.60 (11.51)	78.64 (13.75)

Table C.14: Detailed AUC, UAR and F1-score metrics results from the experiments on instance-level classification on **Newsgroups dataset** [Craven et al. 2008]. Each reported result is the average of 10-folds along with the standard error. Results in bold are best results and results with no statistically significant difference from the best ($\alpha = 0.05$). They come with the number of wins over the different classes for each metrics.

Method	mi-SVM [Andrews et al. 2003]	MI-SVM [Andrews et al. 2003]	EM-DD [Zhang et al. 2002]	SI-SVM	MIL-Boost [Viola et al. 2005]	RSIS-EoSVM [Carbonneau et al. 2016b]	Polyhedral MI-max	MI-max-HL	MI-max
SIVAL: UAR	7	0	0	20	0	0	0	2	0
ajaxorange	82.50 (1.10)	53.50 (0.60)	56.90 (0.80)	**86.50** (0.70)	58.30 (0.60)	67.50 (1.50)	54.39 (6.37)	60.35 (4.01)	58.16 (1.97)
apple	**77.10** (3.00)	59.10 (2.10)	58.10 (1.50)	**76.10** (1.80)	58.50 (1.10)	50.00 (0.00)	52.02 (7.80)	**75.40** (8.64)	67.20 (8.93)
banana	77.00 (2.50)	55.40 (1.20)	56.00 (1.30)	**86.80** (0.90)	53.60 (0.80)	50.00 (0.00)	53.33 (8.95)	72.43 (8.89)	57.10 (7.92)
bluescrunge	**75.60** (1.70)	59.40 (2.30)	56.90 (2.50)	71.40 (2.10)	60.50 (0.70)	50.00 (0.00)	56.36 (11.66)	**73.64** (8.86)	63.65 (3.62)
candlewithholder	74.80 (1.80)	55.90 (1.50)	56.90 (0.60)	**78.80** (1.80)	56.10 (1.10)	62.60 (1.30)	53.51 (6.48)	65.14 (4.96)	61.36 (3.88)
cardboardbox	75.10 (1.40)	52.70 (1.00)	54.80 (0.60)	**76.80** (1.70)	52.80 (0.60)	54.20 (1.20)	51.44 (4.38)	58.01 (4.44)	55.69 (3.42)
checkeredscarf	86.40 (0.90)	55.50 (0.50)	53.70 (0.30)	**89.70** (1.00)	54.30 (0.40)	59.60 (0.90)	56.37 (10.15)	69.70 (8.33)	58.81 (3.52)
cokecan	86.00 (1.30)	57.70 (1.70)	57.20 (0.80)	**88.30** (0.90)	58.90 (0.70)	62.00 (0.90)	55.98 (8.27)	64.18 (3.06)	60.65 (2.99)
dataminingbook	83.10 (1.80)	56.20 (1.00)	56.70 (1.80)	**85.40** (1.40)	61.30 (0.50)	50.00 (0.00)	53.90 (9.02)	73.07 (9.36)	56.73 (4.97)
dirtyrunningshoe	79.90 (1.70)	53.10 (0.60)	52.50 (0.40)	**83.20** (1.40)	52.60 (0.50)	59.20 (0.80)	51.37 (3.73)	60.96 (4.46)	54.44 (2.33)
dirtyworkgloves	**70.40** (1.70)	54.50 (0.90)	53.90 (0.80)	**71.90** (2.40)	54.20 (0.90)	51.30 (0.30)	50.50 (1.87)	57.33 (3.66)	55.31 (2.79)
fabricsoftenerbox	88.20 (0.90)	57.90 (1.90)	61.00 (1.10)	**92.80** (0.50)	62.90 (0.80)	64.80 (0.70)	56.17 (8.16)	68.08 (4.05)	64.20 (2.84)
feltflowerrug	86.30 (1.40)	64.60 (1.50)	57.80 (0.80)	**89.90** (0.90)	61.10 (1.00)	68.10 (0.80)	58.24 (11.67)	75.79 (7.12)	67.65 (4.98)
glazedwoodpot	73.50 (2.50)	54.10 (1.10)	59.00 (1.40)	**77.10** (2.00)	52.50 (0.40)	50.00 (0.00)	50.26 (2.06)	57.94 (3.20)	52.50 (4.59)
goldmedal	76.50 (1.20)	55.50 (1.30)	58.40 (0.80)	**82.40** (1.40)	57.50 (1.00)	50.00 (0.00)	54.29 (7.54)	63.68 (4.10)	62.93 (4.03)
greenteabox	88.10 (1.30)	58.70 (0.80)	53.60 (0.40)	**90.90** (0.50)	55.40 (0.60)	70.30 (0.80)	55.58 (7.89)	60.77 (2.76)	56.61 (2.46)
juliespot	**81.00** (1.50)	59.80 (1.10)	55.60 (1.10)	78.90 (1.60)	54.40 (0.80)	51.30 (0.60)	51.90 (6.19)	66.53 (6.10)	60.34 (5.17)
largespoon	**66.10** (2.50)	50.90 (0.70)	49.90 (0.50)	61.60 (1.30)	51.50 (0.40)	50.00 (0.00)	50.47 (2.52)	53.22 (3.06)	54.37 (3.64)
rapbook	**74.50** (1.10)	53.70 (1.00)	54.80 (1.00)	71.90 (1.70)	52.40 (0.40)	50.00 (0.00)	50.20 (1.27)	55.72 (2.28)	52.34 (2.64)
smileyfacedoll	82.20 (1.70)	57.00 (1.10)	58.70 (0.70)	**85.20** (1.70)	56.30 (0.70)	50.00 (0.00)	54.87 (8.75)	66.71 (6.48)	60.87 (4.40)
spritecan	79.20 (1.30)	52.20 (1.10)	54.40 (0.90)	**84.90** (1.60)	55.00 (0.80)	60.10 (0.50)	53.30 (6.47)	63.43 (5.16)	57.97 (3.55)
stripednotebook	85.90 (1.00)	63.20 (2.20)	57.30 (1.30)	**91.50** (0.90)	58.80 (1.20)	76.40 (2.30)	56.42 (13.13)	78.16 (9.05)	60.91 (6.81)
translucentbowl	81.60 (1.70)	59.80 (1.50)	58.80 (1.30)	**90.40** (0.70)	55.00 (0.60)	50.00 (0.00)	56.24 (13.54)	75.08 (10.11)	60.21 (7.20)
wd40can	89.00 (0.90)	58.60 (1.10)	55.40 (0.80)	**90.40** (0.90)	56.70 (1.00)	68.10 (2.50)	54.81 (6.47)	60.99 (6.52)	55.65 (1.87)
woodrollingpin	**70.00** (1.10)	50.20 (0.30)	55.20 (1.30)	68.00 (2.00)	53.80 (0.80)	50.00 (0.00)	50.55 (2.66)	62.75 (8.49)	54.57 (3.97)
SIVAL: F1	22	1	0	13	1	1	0	7	3
ajaxorange	**68.10** (1.90)	13.10 (2.00)	23.90 (2.40)	65.90 (1.90)	28.10 (1.80)	50.60 (3.70)	24.48 (10.50)	31.81 (8.70)	27.57 (5.62)
apple	22.20 (2.00)	**26.70** (5.50)	25.70 (4.10)	17.50 (1.30)	**27.70** (3.10)	0.00 (0.00)	8.71 (8.30)	**35.07** (15.03)	**35.10** (12.58)
banana	**43.60** (2.70)	17.60 (3.60)	19.50 (4.00)	**41.70** (2.40)	12.80 (2.60)	0.00 (0.00)	12.58 (11.92)	**37.14** (14.83)	19.00 (16.48)
bluescrunge	23.10 (2.10)	27.40 (6.00)	20.90 (7.10)	17.20 (1.90)	33.40 (1.80)	0.00 (0.00)	16.55 (17.85)	**48.77** (17.99)	**38.86** (8.11)
candlewithholder	**52.80** (2.20)	19.50 (4.70)	24.00 (1.80)	**52.60** (2.40)	21.10 (3.30)	38.00 (2.70)	19.80 (10.72)	37.99 (10.25)	34.16 (9.47)
cardboardbox	**42.30** (2.00)	10.80 (3.00)	17.00 (1.80)	**41.50** (3.90)	10.40 (2.20)	14.50 (3.70)	16.77 (6.58)	20.95 (6.13)	19.01 (8.74)
checkeredscarf	75.20 (1.70)	19.90 (1.70)	13.70 (1.10)	**78.20** (1.70)	16.00 (1.30)	31.90 (2.60)	37.33 (12.51)	49.00 (13.28)	29.64 (9.58)
cokecan	**72.30** (1.90)	25.20 (5.10)	24.90 (2.50)	**70.80** (1.80)	30.00 (2.00)	37.80 (2.30)	27.89 (13.78)	43.42 (7.39)	34.09 (7.86)
dataminingbook	**56.20** (4.10)	21.30 (3.10)	22.10 (5.20)	49.80 (1.40)	36.60 (1.30)	0.00 (0.00)	19.27 (13.76)	**45.99** (15.31)	21.48 (11.83)
dirtyrunningshoe	**63.10** (2.50)	12.00 (2.00)	9.80 (1.40)	**63.70** (2.40)	9.90 (1.70)	30.80 (2.30)	25.81 (4.99)	31.91 (8.13)	17.40 (6.73)
dirtyworkgloves	**36.50** (3.50)	16.00 (2.70)	13.90 (2.60)	33.20 (2.80)	15.20 (2.80)	4.90 (1.30)	15.44 (3.49)	20.80 (5.73)	17.87 (6.38)
fabricsoftenerbox	**79.90** (1.60)	25.50 (6.00)	35.50 (3.10)	**79.00** (0.60)	40.80 (2.20)	45.50 (1.60)	31.61 (12.72)	47.73 (8.12)	42.97 (6.99)
feltflowerrug	**73.70** (2.70)	44.10 (3.60)	26.70 (2.40)	**75.00** (1.50)	35.90 (2.70)	52.60 (1.80)	35.51 (19.26)	57.23 (14.40)	50.54 (11.00)
glazedwoodpot	**35.00** (3.10)	14.00 (3.20)	29.20 (4.10)	**32.40** (3.20)	9.30 (1.60)	0.00 (0.00)	10.07 (2.46)	25.04 (8.49)	6.43 (7.48)
goldmedal	**53.10** (1.40)	18.90 (4.00)	28.20 (2.20)	**53.50** (3.30)	25.40 (3.10)	0.00 (0.00)	21.33 (13.31)	41.38 (9.93)	38.50 (9.22)
greenteabox	**77.30** (1.50)	29.30 (2.40)	13.40 (1.40)	**77.90** (1.20)	19.30 (2.00)	56.90 (1.70)	32.00 (11.53)	35.07 (7.33)	23.19 (7.36)
juliespot	**49.00** (3.40)	31.90 (3.00)	19.50 (3.40)	42.50 (2.50)	15.90 (2.30)	4.60 (2.20)	19.53 (9.56)	40.37 (10.71)	31.07 (11.17)
largespoon	**19.60** (1.30)	3.70 (2.20)	1.50 (1.50)	15.00 (0.80)	5.90 (1.50)	0.00 (0.00)	9.73 (3.84)	11.31 (8.68)	**13.87** (8.98)
rapbook	**31.80** (2.40)	13.40 (3.20)	16.90 (3.10)	26.60 (1.40)	9.30 (1.40)	0.00 (0.00)	12.42 (2.57)	19.68 (6.64)	9.01 (7.52)
smileyfacedoll	**61.80** (2.30)	23.30 (3.30)	28.90 (2.00)	54.20 (3.20)	21.80 (2.20)	0.00 (0.00)	19.41 (15.04)	40.32 (12.73)	33.23 (11.02)
spritecan	**60.00** (3.10)	8.80 (3.60)	15.60 (2.90)	**61.00** (3.10)	17.80 (2.70)	33.30 (1.40)	22.38 (9.50)	35.83 (10.36)	25.93 (9.51)
stripednotebook	**69.90** (2.40)	39.60 (5.50)	24.60 (3.40)	67.10 (1.80)	29.10 (3.60)	**67.10** (4.50)	26.52 (21.18)	**58.44** (16.28)	32.70 (16.16)
translucentbowl	**61.60** (4.10)	31.10 (3.80)	28.10 (3.30)	56.50 (2.80)	18.00 (2.00)	0.00 (0.00)	20.43 (19.19)	**50.57** (16.94)	29.83 (16.49)
wd40can	**77.20** (1.60)	28.90 (3.10)	19.20 (2.30)	74.50 (2.30)	23.20 (3.00)	50.40 (5.30)	25.16 (10.72)	31.83 (13.26)	20.07 (5.63)
woodrollingpin	**24.90** (2.70)	2.10 (1.10)	17.30 (3.80)	**22.20** (3.10)	13.60 (2.70)	0.00 (0.00)	8.54 (2.79)	**17.86** (9.84)	13.91 (9.50)
SIVAL: AUC	20	0	0	16	0	0	4	7	1
ajaxorange	**93.70** (0.90)	77.20 (5.20)	56.30 (1.10)	**93.10** (0.90)	74.30 (1.30)	87.20 (1.10)	63.19 (9.27)	65.04 (6.85)	48.46 (8.50)
apple	87.20 (1.40)	70.20 (4.50)	73.00 (2.60)	82.30 (2.50)	81.40 (4.40)	63.40 (2.70)	**86.15** (12.23)	**91.40** (3.30)	**89.12** (4.52)
banana	**91.20** (1.10)	73.90 (4.50)	69.10 (3.60)	**91.60** (0.70)	77.50 (4.30)	76.70 (2.60)	81.52 (12.35)	**87.99** (6.05)	71.89 (12.79)
bluescrunge	85.00 (1.30)	76.70 (4.90)	74.90 (6.90)	81.10 (2.60)	80.80 (2.40)	48.40 (3.20)	**89.47** (7.41)	**89.61** (6.06)	78.49 (7.66)
candlewithholder	**90.40** (0.90)	74.20 (4.90)	55.30 (1.10)	87.60 (1.20)	69.20 (1.60)	76.00 (2.00)	60.08 (12.53)	64.43 (4.66)	54.19 (6.44)
cardboardbox	**85.00** (1.20)	54.80 (2.40)	43.20 (1.60)	**84.70** (1.80)	71.80 (1.80)	67.80 (1.80)	66.58 (9.18)	65.81 (5.99)	61.31 (7.75)
checkeredscarf	**95.20** (0.40)	85.00 (1.40)	75.50 (2.60)	**94.90** (0.60)	88.30 (1.30)	79.50 (1.30)	88.12 (6.13)	88.26 (3.98)	72.22 (7.33)
cokecan	**95.60** (0.50)	84.20 (2.60)	60.70 (0.90)	94.10 (0.70)	86.30 (0.70)	77.40 (1.50)	78.62 (11.92)	76.63 (5.41)	53.87 (10.05)
dataminingbook	**92.70** (1.10)	64.60 (4.80)	67.00 (4.30)	**92.40** (1.10)	82.90 (1.10)	60.40 (3.00)	84.00 (10.47)	**90.68** (3.42)	77.91 (7.98)
dirtyrunningshoe	**91.60** (1.10)	64.70 (4.00)	54.90 (1.20)	**91.10** (1.00)	73.40 (1.10)	77.20 (1.50)	68.83 (11.16)	68.29 (5.15)	51.86 (4.26)
dirtyworkgloves	**78.00** (2.60)	58.70 (2.00)	54.40 (1.80)	**78.60** (2.80)	67.20 (1.70)	63.60 (1.40)	55.64 (8.12)	60.35 (5.43)	54.46 (5.04)
fabricsoftenerbox	96.80 (0.60)	81.90 (1.20)	61.00 (1.20)	**97.50** (0.40)	74.50 (1.00)	76.90 (2.80)	74.76 (9.04)	82.23 (3.67)	68.24 (4.17)
feltflowerrug	**96.20** (0.50)	83.90 (2.90)	70.90 (2.20)	**96.40** (0.60)	86.70 (1.40)	76.10 (1.30)	82.67 (12.61)	89.61 (3.45)	79.28 (4.16)
glazedwoodpot	84.60 (2.30)	71.80 (2.00)	79.80 (2.00)	**86.80** (1.50)	73.20 (2.30)	73.90 (1.90)	70.49 (10.20)	78.54 (5.87)	70.22 (7.81)
goldmedal	**92.20** (1.10)	79.40 (3.10)	69.70 (3.50)	89.60 (1.10)	77.20 (1.90)	80.00 (2.40)	67.75 (10.54)	74.82 (5.00)	63.27 (6.87)
greenteabox	95.70 (0.50)	69.90 (0.90)	65.40 (6.70)	**96.30** (0.50)	77.30 (0.80)	87.00 (0.60)	79.44 (7.37)	78.29 (4.70)	65.96 (4.71)
juliespot	**88.60** (1.30)	78.00 (2.10)	66.50 (3.70)	**87.20** (2.10)	80.80 (2.40)	84.70 (1.70)	76.31 (12.87)	81.84 (3.72)	76.19 (5.42)
largespoon	**76.00** (2.50)	42.90 (3.00)	27.70 (3.20)	66.60 (2.20)	45.80 (2.50)	33.50 (2.40)	62.83 (11.59)	58.60 (9.27)	56.52 (10.79)
rapbook	**82.50** (1.00)	58.40 (2.30)	56.50 (2.40)	80.90 (1.50)	75.70 (1.10)	68.20 (1.80)	58.09 (8.64)	58.52 (5.99)	61.09 (8.89)
smileyfacedoll	**92.90** (0.90)	76.20 (4.10)	63.60 (3.30)	**92.00** (1.40)	80.80 (2.90)	66.00 (2.90)	81.24 (9.80)	81.74 (4.67)	66.85 (6.10)
spritecan	**92.70** (1.10)	64.70 (6.20)	57.80 (1.40)	90.30 (1.30)	81.30 (1.60)	72.30 (1.50)	74.83 (10.39)	71.88 (5.08)	58.46 (7.76)
stripednotebook	**96.00** (0.60)	86.70 (2.20)	63.10 (4.10)	**95.50** (0.70)	91.50 (0.60)	94.40 (1.40)	**89.80** (11.22)	**94.69** (2.72)	85.83 (6.21)
translucentbowl	**94.80** (1.10)	84.90 (2.40)	74.50 (1.70)	**94.90** (0.40)	85.50 (1.50)	73.00 (2.70)	**91.84** (8.36)	**94.05** (2.65)	84.99 (7.58)
wd40can	**96.70** (0.30)	76.30 (3.00)	62.60 (2.50)	95.00 (0.90)	75.60 (1.80)	90.40 (1.40)	68.37 (13.12)	72.26 (6.35)	51.65 (6.45)
woodrollingpin	**78.00** (1.60)	45.50 (2.90)	70.90 (2.20)	**77.30** (2.40)	70.70 (4.40)	55.60 (2.50)	62.83 (13.46)	**73.57** (10.76)	64.79 (10.91)

Table C.15: Detailed AUC, UAR and F1-score metrics results from the experiments on instance-level classification on **SIVAL dataset** [Rahmani et al. 2005]. Each reported result is the average of 10-folds along with the standard error. Results in bold are best results and results with no statistically significant difference from the best ($\alpha = 0.05$). They come with the number of wins over the different classes for each metrics.

D

Detailed Convolutional Neural Network Architectures

In this annex, we print the extremely detailed architecture of the three CNN used for feature viusalization (Chapter 4): InceptionV1, ResNet50 and VGG19 in Tables D.1 to D.3. The latter is also used for texture synthesis (Chapter 3).

The layer used are the following one:

- Conv2D: 2D convolution layer
- ReLU: Rectified Linear Unit
- Dense: fully connected layer
- MaxPooling2D: Max pooling operation for 2D spatial data
- Flatten: Collapse the spatial and kernel dimensions to one unique dimension
- BN: Batch Normalization [Ioffe 2017]
- ZeroPadding2D: 2D zero padding
- PoolHelper: remove the first row and column in image dimensions
- LRN: Local response normalization from [Szegedy et al. 2015]
- GlobalAveragePooling2D : Spatial average
- Dropout: this layer randomly sets input neurons to 0 with a given frequency of rate at each step during training time

Table D.1: InceptionV1 detailed Architecture [Szegedy et al. 2015]. The model contains 13461996 trainable parameters with the deep supervision and thus the two extra classification branches. The model contains 7125020 trainable parameters without the deep supervision branches.

Layer Name	Type	Output Shape	Param #	Connected to
input	Input Layer	(224, 224, 3)		
zero_padding2d_0	ZeroPadding2D	(230, 230, 3)	0	input_1
conv2d0_pre_relu	Conv2D	(112, 112, 64)	9472	zero_padding2d_0
conv2d0	ReLU	(112, 112, 64)	0	conv2d0_pre_relu
zero_padding2d_1	ZeroPadding2D	(114, 114, 64)	0	conv2d0
pool_helper_8	PoolHelper	(113, 113, 64)	0	zero_padding2d_1
maxpool0	MaxPooling2D	(56, 56, 64)	0	pool_helper_8
localresponsenorm0	LRN	(56, 56, 64)	0	maxpool0
conv2d1_pre_relu	Conv2D	(56, 56, 64)	4160	localresponsenorm0
conv2d1	ReLU	(56, 56, 64)	0	conv2d1_pre_relu
conv2d2_pre_relu	Conv2D	(56, 56, 192)	110784	conv2d1
conv2d2	ReLU	(56, 56, 192)	0	conv2d2_pre_relu
localresponsenorm1	LRN	(56, 56, 192)	0	conv2d2
zero_padding2d_2	ZeroPadding2D	(58, 58, 192)	0	localresponsenorm1
pool_helper_9	PoolHelper	(57, 57, 192)	0	zero_padding2d_2
maxpool1	MaxPooling2D	(28, 28, 192)	0	pool_helper_9
mixed3a_3x3_bottleneck_pre_relu	Conv2D	(28, 28, 96)	18528	maxpool1
mixed3a_5x5_bottleneck_pre_relu	Conv2D	(28, 28, 16)	3088	maxpool1
mixed3a_3x3_bottleneck	ReLU	(28, 28, 96)	0	mixed3a_3x3_bottleneck_pre_relu
mixed3a_5x5_bottleneck	ReLU	(28, 28, 16)	0	mixed3a_5x5_bottleneck_pre_relu
zero_padding2d_3	ZeroPadding2D	(30, 30, 96)	0	mixed3a_3x3_bottleneck
zero_padding2d_4	ZeroPadding2D	(32, 32, 16)	0	mixed3a_5x5_bottleneck
mixed3a_pool_pre_relu	MaxPooling2D	(28, 28, 192)	0	maxpool1
mixed3a_1x1_pre_relu	Conv2D	(28, 28, 64)	12352	maxpool1
mixed3a_3x3_pre_relu	Conv2D	(28, 28, 128)	110720	zero_padding2d_3
mixed3a_5x5_pre_relu	Conv2D	(28, 28, 32)	12832	zero_padding2d_4
mixed3a_pool_reduce_pre_relu	Conv2D	(28, 28, 32)	6176	mixed3a_pool_pre_relu
mixed3a_pre_relu	Concatenate	(28, 28, 256)	0	mixed3a_1x1_pre_relu mixed3a_3x3_pre_relu mixed3a_5x5_pre_relu mixed3a_pool_reduce_pre_relu
mixed3a	ReLU	(28, 28, 256)	0	mixed3a_pre_relu
mixed3b_3x3_bottleneck_pre_relu	Conv2D	(28, 28, 128)	32896	mixed3a
mixed3b_5x5_bottleneck_pre_relu	Conv2D	(28, 28, 32)	8224	mixed3a
mixed3b_3x3_bottleneck	ReLU	(28, 28, 128)	0	mixed3b_3x3_bottleneck_pre_relu
mixed3b_5x5_bottleneck	ReLU	(28, 28, 32)	0	mixed3b_5x5_bottleneck_pre_relu
zero_padding2d_5	ZeroPadding2D	(30, 30, 128)	0	mixed3b_3x3_bottleneck
zero_padding2d_6	ZeroPadding2D	(32, 32, 32)	0	mixed3b_5x5_bottleneck
mixed3b_pool	MaxPooling2D	(28, 28, 256)	0	mixed3a
mixed3b_1x1_pre_relu	Conv2D	(28, 28, 128)	32896	mixed3a
mixed3b_3x3_pre_relu	Conv2D	(28, 28, 192)	221376	zero_padding2d_5
mixed3b_5x5_pre_relu	Conv2D	(28, 28, 96)	76896	zero_padding2d_6
mixed3b_pool_reduce_pre_relu	Conv2D	(28, 28, 64)	16448	mixed3b_pool
mixed3b_pre_relu	Concatenate	(28, 28, 480)	0	mixed3b_1x1_pre_relu mixed3b_3x3_pre_relu mixed3b_5x5_pre_relu mixed3b_pool_reduce_pre_relu
mixed3b	ReLU	(28, 28, 480)	0	mixed3b_pre_relu
zero_padding2d_7	ZeroPadding2D	(30, 30, 480)	0	mixed3b
pool_helper_10	PoolHelper	(29, 29, 480)	0	zero_padding2d_7
maxpool2	MaxPooling2D	(14, 14, 480)	0	pool_helper_10
mixed4a_3x3_bottleneck_pre_relu	Conv2D	(14, 14, 96)	46176	maxpool2
mixed4a_5x5_bottleneck_pre_relu	Conv2D	(14, 14, 16)	7696	maxpool2
mixed4a_3x3_bottleneck	ReLU	(14, 14, 96)	0	mixed4a_3x3_bottleneck_pre_relu
mixed4a_5x5_bottleneck	ReLU	(14, 14, 16)	0	mixed4a_5x5_bottleneck_pre_relu
zero_padding2d_8	ZeroPadding2D	(16, 16, 96)	0	mixed4a_3x3_bottleneck
zero_padding2d_9	ZeroPadding2D	(18, 18, 16)	0	mixed4a_5x5_bottleneck
mixed4a_pool	MaxPooling2D	(14, 14, 480)	0	maxpool2
mixed4a_1x1_pre_relu	Conv2D	(14, 14, 192)	92352	maxpool2
mixed4a_3x3_pre_relu	Conv2D	(14, 14, 204)	176460	zero_padding2d_8
mixed4a_5x5_pre_relu	Conv2D	(14, 14, 48)	19248	zero_padding2d_9
mixed4a_pool_reduce_pre_relu	Conv2D	(14, 14, 64)	30784	mixed4a_pool
mixed4a_pre_relu	Concatenate	(14, 14, 508)	0	mixed4a_1x1_pre_relu mixed4a_3x3_pre_relu mixed4a_5x5_pre_relu mixed4a_pool_reduce_pre_relu
mixed4a	ReLU	(14, 14, 508)	0	mixed4a_pre_relu
mixed4b_3x3_bottleneck_pre_relu	Conv2D	(14, 14, 112)	57008	mixed4a

mixed4b_5x5_bottleneck_pre_relu	Conv2D	(14, 14, 24)	12216	mixed4a
mixed4b_3x3_bottleneck	ReLU	(14, 14, 112)	0	mixed4b_3x3_bottleneck_pre_relu
mixed4b_5x5_bottleneck	ReLU	(14, 14, 24)	0	mixed4b_5x5_bottleneck_pre_relu
zero_padding2d_10	ZeroPadding2D	(16, 16, 112)	0	mixed4b_3x3_bottleneck
zero_padding2d_11	ZeroPadding2D	(18, 18, 24)	0	mixed4b_5x5_bottleneck
mixed4b_pool	MaxPooling2D	(14, 14, 508)	0	mixed4a
mixed4b_1x1_pre_relu	Conv2D	(14, 14, 160)	81440	mixed4a
mixed4b_3x3_pre_relu	Conv2D	(14, 14, 224)	226016	zero_padding2d_10
mixed4b_5x5_pre_relu	Conv2D	(14, 14, 64)	38464	zero_padding2d_11
mixed4b_pool_reduce_pre_relu	Conv2D	(14, 14, 64)	32576	mixed4b_pool
mixed4b_pre_relu	Concatenate	(14, 14, 512)	0	mixed4b_1x1_pre_relu mixed4b_3x3_pre_relu mixed4b_5x5_pre_relu mixed4b_pool_reduce_pre_relu
mixed4b	ReLU	(14, 14, 512)	0	mixed4b_pre_relu
mixed4c_3x3_bottleneck_pre_relu	Conv2D	(14, 14, 128)	65664	mixed4b
mixed4c_5x5_bottleneck_pre_relu	Conv2D	(14, 14, 24)	12312	mixed4b
mixed4c_3x3_bottleneck	ReLU	(14, 14, 128)	0	mixed4c_3x3_bottleneck_pre_relu
mixed4c_5x5_bottleneck	ReLU	(14, 14, 24)	0	mixed4c_5x5_bottleneck_pre_relu
zero_padding2d_12	ZeroPadding2D	(16, 16, 128)	0	mixed4c_3x3_bottleneck
zero_padding2d_13	ZeroPadding2D	(18, 18, 24)	0	mixed4c_5x5_bottleneck
mixed4c_pool	MaxPooling2D	(14, 14, 512)	0	mixed4b
mixed4c_1x1_pre_relu	Conv2D	(14, 14, 128)	65664	mixed4b
mixed4c_3x3_pre_relu	Conv2D	(14, 14, 256)	295168	zero_padding2d_12
mixed4c_5x5_pre_relu	Conv2D	(14, 14, 64)	38464	zero_padding2d_13
mixed4c_pool_reduce_pre_relu	Conv2D	(14, 14, 64)	32832	mixed4c_pool
mixed4c_pre_relu	Concatenate	(14, 14, 512)	0	mixed4c_1x1_pre_relu mixed4c_3x3_pre_relu mixed4c_5x5_pre_relu mixed4c_pool_reduce_pre_relu
mixed4c	ReLU	(14, 14, 512)	0	mixed4c_pre_relu
mixed4d_3x3_bottleneck_pre_relu	Conv2D	(14, 14, 144)	73872	mixed4c
mixed4d_5x5_bottleneck_pre_relu	Conv2D	(14, 14, 32)	16416	mixed4c
mixed4d_3x3_bottleneck	ReLU	(14, 14, 144)	0	mixed4d_3x3_bottleneck_pre_relu
mixed4d_5x5_bottleneck	ReLU	(14, 14, 32)	0	mixed4d_5x5_bottleneck_pre_relu
zero_padding2d_14	ZeroPadding2D	(16, 16, 144)	0	mixed4d_3x3_bottleneck
zero_padding2d_15	ZeroPadding2D	(18, 18, 32)	0	mixed4d_5x5_bottleneck
mixed4d_pool	MaxPooling2D	(14, 14, 512)	0	mixed4c
mixed4d_1x1_pre_relu	Conv2D	(14, 14, 112)	57456	mixed4c
mixed4d_3x3_pre_relu	Conv2D	(14, 14, 288)	373536	zero_padding2d_14
mixed4d_5x5_pre_relu	Conv2D	(14, 14, 64)	51264	zero_padding2d_15
mixed4d_pool_reduce_pre_relu	Conv2D	(14, 14, 64)	32832	mixed4d_pool
mixed4d_pre_relu	Concatenate	(14, 14, 528)	0	mixed4d_1x1_pre_relu mixed4d_3x3_pre_relu mixed4d_5x5_pre_relu mixed4d_pool_reduce_pre_relu
mixed4d	ReLU	(14, 14, 528)	0	mixed4d_pre_relu
mixed4e_3x3_bottleneck_pre_relu	Conv2D	(14, 14, 160)	84640	mixed4d
mixed4e_5x5_bottleneck_pre_relu	Conv2D	(14, 14, 32)	16928	mixed4d
mixed4e_3x3_bottleneck	ReLU	(14, 14, 160)	0	mixed4e_3x3_bottleneck_pre_relu
mixed4e_5x5_bottleneck	ReLU	(14, 14, 32)	0	mixed4e_5x5_bottleneck_pre_relu
zero_padding2d_16	ZeroPadding2D	(16, 16, 160)	0	mixed4e_3x3_bottleneck
zero_padding2d_17	ZeroPadding2D	(18, 18, 32)	0	mixed4e_5x5_bottleneck
mixed4e_pool	MaxPooling2D	(14, 14, 528)	0	mixed4d
mixed4e_1x1_pre_relu	Conv2D	(14, 14, 256)	135424	mixed4d
mixed4e_3x3_pre_relu	Conv2D	(14, 14, 320)	461120	zero_padding2d_16
mixed4e_5x5_pre_relu	Conv2D	(14, 14, 128)	102528	zero_padding2d_17
mixed4e_pool_reduce_pre_relu	Conv2D	(14, 14, 128)	67712	mixed4e_pool
mixed4e_pre_relu	Concatenate	(14, 14, 832)	0	mixed4e_1x1_pre_relu mixed4e_3x3_pre_relu mixed4e_5x5_pre_relu mixed4e_pool_reduce_pre_relu
mixed4e	ReLU	(14, 14, 832)	0	mixed4e_pre_relu
zero_padding2d_18	ZeroPadding2D	(16, 16, 832)	0	mixed4e
pool_helper_11	PoolHelper	(15, 15, 832)	0	zero_padding2d_18
maxpool3	MaxPooling2D	(7, 7, 832)	0	pool_helper_11
mixed5a_3x3_bottleneck_pre_relu	Conv2D	(7, 7, 160)	133280	maxpool3
mixed5a_5x5_bottleneck_pre_relu	Conv2D	(7, 7, 48)	39984	maxpool3
mixed5a_3x3_bottleneck	ReLU	(7, 7, 160)	0	mixed5a_3x3_bottleneck_pre_relu
mixed5a_5x5_bottleneck	ReLU	(7, 7, 48)	0	mixed5a_5x5_bottleneck_pre_relu
zero_padding2d_19	ZeroPadding2D	(9, 9, 160)	0	mixed5a_3x3_bottleneck
zero_padding2d_20	ZeroPadding2D	(11, 11, 48)	0	mixed5a_5x5_bottleneck
mixed5a_pool	MaxPooling2D	(7, 7, 832)	0	maxpool3

mixed5a_1x1_pre_relu	Conv2D	(7, 7, 256)	213248	maxpool3
mixed5a_3x3_pre_relu	Conv2D	(7, 7, 320)	461120	zero_padding2d_19
mixed5a_5x5_pre_relu	Conv2D	(7, 7, 128)	153728	zero_padding2d_20
mixed5a_pool_reduce_pre_relu	Conv2D	(7, 7, 128)	106624	mixed5a_pool
mixed5a_pre_relu	Concatenate	(7, 7, 832)	0	mixed5a_1x1_pre_relu mixed5a_3x3_pre_relu mixed5a_5x5_pre_relu mixed5a_pool_reduce_pre_relu
mixed5a	ReLU	(7, 7, 832)	0	mixed5a_pre_relu
mixed5b_3x3_bottleneck_pre_relu	Conv2D	(7, 7, 192)	159936	mixed5a
mixed5b_5x5_bottleneck_pre_relu	Conv2D	(7, 7, 48)	39984	mixed5a
mixed5b_3x3_bottleneck	ReLU	(7, 7, 192)	0	mixed5b_3x3_bottleneck_pre_relu
mixed5b_5x5_bottleneck	ReLU	(7, 7, 48)	0	mixed5b_5x5_bottleneck_pre_relu
head0_pool	AveragePooling2D	(4, 4, 508)	0	mixed4a
head1_pool	AveragePooling2D	(4, 4, 528)	0	mixed4d
zero_padding2d_21	ZeroPadding2D	(9, 9, 192)	0	mixed5b_3x3_bottleneck
zero_padding2d_22	ZeroPadding2D	(11, 11, 48)	0	mixed5b_5x5_bottleneck
mixed5b_pool	MaxPooling2D	(7, 7, 832)	0	mixed5a
head0_bottleneck_pre_relu	Conv2D	(4, 4, 128)	65152	head0_pool
head1_bottleneck_pre_relu	Conv2D	(4, 4, 128)	67712	head1_pool
mixed5b_1x1_pre_relu	Conv2D	(7, 7, 384)	319872	mixed5a
mixed5b_3x3_pre_relu	Conv2D	(7, 7, 384)	663936	zero_padding2d_21
mixed5b_5x5_pre_relu	Conv2D	(7, 7, 128)	153728	zero_padding2d_22
mixed5b_pool_reduce_pre_relu	Conv2D	(7, 7, 128)	106624	mixed5b_pool
head0_bottleneck	ReLU	(4, 4, 128)	0	head0_bottleneck_pre_relu
head1_bottleneck	ReLU	(4, 4, 128)	0	head1_bottleneck_pre_relu
mixed5b_pre_relu	Concatenate	(7, 7, 1024)	0	mixed5b_1x1_pre_relu mixed5b_3x3_pre_relu mixed5b_5x5_pre_relu mixed5b_pool_reduce_pre_relu
flatten	Flatten	(2048)	0	head0_bottleneck
flatten_1	Flatten	(2048)	0	head1_bottleneck
mixed5b	ReLU	(7, 7, 1024)	0	mixed5b_pre_relu
nn0_pre_relu	Dense	(1024)	2098176	flatten
nn1_pre_relu	Dense	(1024)	2098176	flatten_1
avgpool	AveragePooling2D	(1, 1, 1024)	0	mixed5b
nn0	ReLU	(1024)	0	nn0_pre_relu
nn1	ReLU	(1024)	0	nn1_pre_relu
flatten_2	Flatten	(1024)	0	avgpool
dropout	Dropout	(1024)	0	nn0
dropout_1	Dropout	(1024)	0	nn1
dropout_2	Dropout	(1024)	0	flatten_2
softmax0_pre_activation	Dense	(1000)	1033200	dropout
softmax1_pre_activation	Dense	(1000)	1033200	dropout_1
softmax2_pre_activation	Dense	(1000)	1033200	dropout_2
softname0	Softmax	(1000)	0	softmax0_pre_activation
softname1	Softmax	(1000)	0	softmax1_pre_activation
softmax2	Softmax	(1000)	0	softmax2_pre_activation

Table D.2: ResNet50 (50 layers) detailed Architecture [He et al. 2015]. The model contains 25583592 trainable parameters.

Layer Name	Type	Output Shape	Param #	Connected to
input	InputLayer	(224, 224, 3)		
conv1_pad	ZeroPadding2D	(230, 230, 3)	0	input
conv1_conv	Conv2D	(112, 112, 64)	9472	conv1_pad
conv1_bn	BN	zation) (112, 112, 64)	256	conv1_conv
conv1_relu	ReLU	(112, 112, 64)	0	conv1_bn
pool1_pad	ZeroPadding2D	(114, 114, 64)	0	conv1_relu
pool1_pool	MaxPooling2D	(56, 56, 64)	0	pool1_pad
conv2_block1_1_conv	Conv2D	(56, 56, 64)	4160	pool1_pool
conv2_block1_1_bn	BN	(56, 56, 64)	256	conv2_block1_1_conv
conv2_block1_1_relu	ReLU	(56, 56, 64)	0	conv2_block1_1_bn
conv2_block1_2_conv	Conv2D	(56, 56, 64)	36928	conv2_block1_1_relu
conv2_block1_2_bn	BN	(56, 56, 64)	256	conv2_block1_2_conv
conv2_block1_2_relu	ReLU	(56, 56, 64)	0	conv2_block1_2_bn
conv2_block1_0_conv	Conv2D	(56, 56, 256)	16640	pool1_pool
conv2_block1_3_conv	Conv2D	(56, 56, 256)	16640	conv2_block1_2_relu
conv2_block1_0_bn	BN	(56, 56, 256)	1024	conv2_block1_0_conv
conv2_block1_3_bn	BN	(56, 56, 256)	1024	conv2_block1_3_conv
conv2_block1_add	Add	(56, 56, 256)	0	conv2_block1_0_bn conv2_block1_3_bn

conv2_block1_out	ReLU	(56, 56, 256)	0	conv2_block1_add
conv2_block2_1_conv	Conv2D	(56, 56, 64)	16448	conv2_block1_out
conv2_block2_1_bn	BN	(56, 56, 64)	256	conv2_block2_1_conv
conv2_block2_1_relu	ReLU	(56, 56, 64)	0	conv2_block2_1_bn
conv2_block2_2_conv	Conv2D	(56, 56, 64)	36928	conv2_block2_1_relu
conv2_block2_2_bn	BN	(56, 56, 64)	256	conv2_block2_2_conv
conv2_block2_2_relu	ReLU	(56, 56, 64)	0	conv2_block2_2_bn
conv2_block2_3_conv	Conv2D	(56, 56, 256)	16640	conv2_block2_2_relu
conv2_block2_3_bn	BN	(56, 56, 256)	1024	conv2_block2_3_conv
conv2_block2_add	Add	(56, 56, 256)	0	conv2_block1_out conv2_block2_3_bn
conv2_block2_out	ReLU	(56, 56, 256)	0	conv2_block2_add
conv2_block3_1_conv	Conv2D	(56, 56, 64)	16448	conv2_block2_out
conv2_block3_1_bn	BN	(56, 56, 64)	256	conv2_block3_1_conv
conv2_block3_1_relu	ReLU	(56, 56, 64)	0	conv2_block3_1_bn
conv2_block3_2_conv	Conv2D	(56, 56, 64)	36928	conv2_block3_1_relu
conv2_block3_2_bn	BN	(56, 56, 64)	256	conv2_block3_2_conv
conv2_block3_2_relu	ReLU	(56, 56, 64)	0	conv2_block3_2_bn
conv2_block3_3_conv	Conv2D	(56, 56, 256)	16640	conv2_block3_2_relu
conv2_block3_3_bn	BN	(56, 56, 256)	1024	conv2_block3_3_conv
conv2_block3_add	Add	(56, 56, 256)	0	conv2_block2_out conv2_block3_3_bn
conv2_block3_out	ReLU	(56, 56, 256)	0	conv2_block3_add
conv3_block1_1_conv	Conv2D	(28, 28, 128)	32896	conv2_block3_out
conv3_block1_1_bn	BN	(28, 28, 128)	512	conv3_block1_1_conv
conv3_block1_1_relu	ReLU	(28, 28, 128)	0	conv3_block1_1_bn
conv3_block1_2_conv	Conv2D	(28, 28, 128)	147584	conv3_block1_1_relu
conv3_block1_2_bn	BN	(28, 28, 128)	512	conv3_block1_2_conv
conv3_block1_2_relu	ReLU	(28, 28, 128)	0	conv3_block1_2_bn
conv3_block1_0_conv	Conv2D	(28, 28, 512)	131584	conv2_block3_out
conv3_block1_3_conv	Conv2D	(28, 28, 512)	66048	conv3_block1_2_relu
conv3_block1_0_bn	BN	(28, 28, 512)	2048	conv3_block1_0_conv
conv3_block1_3_bn	BN	(28, 28, 512)	2048	conv3_block1_3_conv
conv3_block1_add	Add	(28, 28, 512)	0	conv3_block1_0_bn conv3_block1_3_bn
conv3_block1_out	ReLU	(28, 28, 512)	0	conv3_block1_add
conv3_block2_1_conv	Conv2D	(28, 28, 128)	65664	conv3_block1_out
conv3_block2_1_bn	BN	(28, 28, 128)	512	conv3_block2_1_conv
conv3_block2_1_relu	ReLU	(28, 28, 128)	0	conv3_block2_1_bn
conv3_block2_2_conv	Conv2D	(28, 28, 128)	147584	conv3_block2_1_relu
conv3_block2_2_bn	BN	(28, 28, 128)	512	conv3_block2_2_conv
conv3_block2_2_relu	ReLU	(28, 28, 128)	0	conv3_block2_2_bn
conv3_block2_3_conv	Conv2D	(28, 28, 512)	66048	conv3_block2_2_relu
conv3_block2_3_bn	BN	(28, 28, 512)	2048	conv3_block2_3_conv
conv3_block2_add	Add	(28, 28, 512)	0	conv3_block1_out conv3_block2_3_bn
conv3_block2_out	ReLU	(28, 28, 512)	0	conv3_block2_add
conv3_block3_1_conv	Conv2D	(28, 28, 128)	65664	conv3_block2_out
conv3_block3_1_bn	BN	(28, 28, 128)	512	conv3_block3_1_conv
conv3_block3_1_relu	ReLU	(28, 28, 128)	0	conv3_block3_1_bn
conv3_block3_2_conv	Conv2D	(28, 28, 128)	147584	conv3_block3_1_relu
conv3_block3_2_bn	BN	(28, 28, 128)	512	conv3_block3_2_conv
conv3_block3_2_relu	ReLU	(28, 28, 128)	0	conv3_block3_2_bn
conv3_block3_3_conv	Conv2D	(28, 28, 512)	66048	conv3_block3_2_relu
conv3_block3_3_bn	BN	(28, 28, 512)	2048	conv3_block3_3_conv
conv3_block3_add	Add	(28, 28, 512)	0	conv3_block2_out conv3_block3_3_bn
conv3_block3_out	ReLU	(28, 28, 512)	0	conv3_block3_add
conv3_block4_1_conv	Conv2D	(28, 28, 128)	65664	conv3_block3_out
conv3_block4_1_bn	BN	(28, 28, 128)	512	conv3_block4_1_conv
conv3_block4_1_relu	ReLU	(28, 28, 128)	0	conv3_block4_1_bn
conv3_block4_2_conv	Conv2D	(28, 28, 128)	147584	conv3_block4_1_relu
conv3_block4_2_bn	BN	(28, 28, 128)	512	conv3_block4_2_conv
conv3_block4_2_relu	ReLU	(28, 28, 128)	0	conv3_block4_2_bn
conv3_block4_3_conv	Conv2D	(28, 28, 512)	66048	conv3_block4_2_relu
conv3_block4_3_bn	BN	(28, 28, 512)	2048	conv3_block4_3_conv
conv3_block4_add	Add	(28, 28, 512)	0	conv3_block3_out conv3_block4_3_bn
conv3_block4_out	ReLU	(28, 28, 512)	0	conv3_block4_add
conv4_block1_1_conv	Conv2D	(14, 14, 256)	131328	conv3_block4_out
conv4_block1_1_bn	BN	(14, 14, 256)	1024	conv4_block1_1_conv
conv4_block1_1_relu	ReLU	(14, 14, 256)	0	conv4_block1_1_bn
conv4_block1_2_conv	Conv2D	(14, 14, 256)	590080	conv4_block1_1_relu

conv4_block1_2_bn	BN	(14, 14, 256)	1024	conv4_block1_2_conv
conv4_block1_2_relu	ReLU	(14, 14, 256)	0	conv4_block1_2_bn
conv4_block1_0_conv	Conv2D	(14, 14, 1024)	525312	conv3_block4_out
conv4_block1_3_conv	Conv2D	(14, 14, 1024)	263168	conv4_block1_2_relu
conv4_block1_0_bn	BN	(14, 14, 1024)	4096	conv4_block1_0_conv
conv4_block1_3_bn	BN	(14, 14, 1024)	4096	conv4_block1_3_conv
conv4_block1_add	Add	(14, 14, 1024)	0	conv4_block1_0_bn conv4_block1_3_bn
conv4_block1_out	ReLU	(14, 14, 1024)	0	conv4_block1_add
conv4_block2_1_conv	Conv2D	(14, 14, 256)	262400	conv4_block1_out
conv4_block2_1_bn	BN	(14, 14, 256)	1024	conv4_block2_1_conv
conv4_block2_1_relu	ReLU	(14, 14, 256)	0	conv4_block2_1_bn
conv4_block2_2_conv	Conv2D	(14, 14, 256)	590080	conv4_block2_1_relu
conv4_block2_2_bn	BN	(14, 14, 256)	1024	conv4_block2_2_conv
conv4_block2_2_relu	ReLU	(14, 14, 256)	0	conv4_block2_2_bn
conv4_block2_3_conv	Conv2D	(14, 14, 1024)	263168	conv4_block2_2_relu
conv4_block2_3_bn	BN	(14, 14, 1024)	4096	conv4_block2_3_conv
conv4_block2_add	Add	(14, 14, 1024)	0	conv4_block1_out conv4_block2_3_bn
conv4_block2_out	ReLU	(14, 14, 1024)	0	conv4_block2_add
conv4_block3_1_conv	Conv2D	(14, 14, 256)	262400	conv4_block2_out
conv4_block3_1_bn	BN	(14, 14, 256)	1024	conv4_block3_1_conv
conv4_block3_1_relu	ReLU	(14, 14, 256)	0	conv4_block3_1_bn
conv4_block3_2_conv	Conv2D	(14, 14, 256)	590080	conv4_block3_1_relu
conv4_block3_2_bn	BN	(14, 14, 256)	1024	conv4_block3_2_conv
conv4_block3_2_relu	ReLU	(14, 14, 256)	0	conv4_block3_2_bn
conv4_block3_3_conv	Conv2D	(14, 14, 1024)	263168	conv4_block3_2_relu
conv4_block3_3_bn	BN	(14, 14, 1024)	4096	conv4_block3_3_conv
conv4_block3_add	Add	(14, 14, 1024)	0	conv4_block2_out conv4_block3_3_bn
conv4_block3_out	ReLU	(14, 14, 1024)	0	conv4_block3_add
conv4_block4_1_conv	Conv2D	(14, 14, 256)	262400	conv4_block3_out
conv4_block4_1_bn	BN	(14, 14, 256)	1024	conv4_block4_1_conv
conv4_block4_1_relu	ReLU	(14, 14, 256)	0	conv4_block4_1_bn
conv4_block4_2_conv	Conv2D	(14, 14, 256)	590080	conv4_block4_1_relu
conv4_block4_2_bn	BN	(14, 14, 256)	1024	conv4_block4_2_conv
conv4_block4_2_relu	ReLU	(14, 14, 256)	0	conv4_block4_2_bn
conv4_block4_3_conv	Conv2D	(14, 14, 1024)	263168	conv4_block4_2_relu
conv4_block4_3_bn	BN	(14, 14, 1024)	4096	conv4_block4_3_conv
conv4_block4_add	Add	(14, 14, 1024)	0	conv4_block3_out conv4_block4_3_bn
conv4_block4_out	ReLU	(14, 14, 1024)	0	conv4_block4_add
conv4_block5_1_conv	Conv2D	(14, 14, 256)	262400	conv4_block4_out
conv4_block5_1_bn	BN	(14, 14, 256)	1024	conv4_block5_1_conv
conv4_block5_1_relu	ReLU	(14, 14, 256)	0	conv4_block5_1_bn
conv4_block5_2_conv	Conv2D	(14, 14, 256)	590080	conv4_block5_1_relu
conv4_block5_2_bn	BN	(14, 14, 256)	1024	conv4_block5_2_conv
conv4_block5_2_relu	ReLU	(14, 14, 256)	0	conv4_block5_2_bn
conv4_block5_3_conv	Conv2D	(14, 14, 1024)	263168	conv4_block5_2_relu
conv4_block5_3_bn	BN	(14, 14, 1024)	4096	conv4_block5_3_conv
conv4_block5_add	Add	(14, 14, 1024)	0	conv4_block4_out conv4_block5_3_bn
conv4_block5_out	ReLU	(14, 14, 1024)	0	conv4_block5_add
conv4_block6_1_conv	Conv2D	(14, 14, 256)	262400	conv4_block5_out
conv4_block6_1_bn	BN	(14, 14, 256)	1024	conv4_block6_1_conv
conv4_block6_1_relu	ReLU	(14, 14, 256)	0	conv4_block6_1_bn
conv4_block6_2_conv	Conv2D	(14, 14, 256)	590080	conv4_block6_1_relu
conv4_block6_2_bn	BN	(14, 14, 256)	1024	conv4_block6_2_conv
conv4_block6_2_relu	ReLU	(14, 14, 256)	0	conv4_block6_2_bn
conv4_block6_3_conv	Conv2D	(14, 14, 1024)	263168	conv4_block6_2_relu
conv4_block6_3_bn	BN	(14, 14, 1024)	4096	conv4_block6_3_conv
conv4_block6_add	Add	(14, 14, 1024)	0	conv4_block5_out conv4_block6_3_bn
conv4_block6_out	ReLU	(14, 14, 1024)	0	conv4_block6_add
conv5_block1_1_conv	Conv2D	(7, 7, 512)	524800	conv4_block6_out
conv5_block1_1_bn	BN	(7, 7, 512)	2048	conv5_block1_1_conv
conv5_block1_1_relu	ReLU	(7, 7, 512)	0	conv5_block1_1_bn
conv5_block1_2_conv	Conv2D	(7, 7, 512)	2359808	conv5_block1_1_relu
conv5_block1_2_bn	BN	(7, 7, 512)	2048	conv5_block1_2_conv
conv5_block1_2_relu	ReLU	(7, 7, 512)	0	conv5_block1_2_bn
conv5_block1_0_conv	Conv2D	(7, 7, 2048)	2099200	conv4_block6_out
conv5_block1_3_conv	Conv2D	(7, 7, 2048)	1050624	conv5_block1_2_relu
conv5_block1_0_bn	BN	(7, 7, 2048)	8192	conv5_block1_0_conv

conv5_block1_3_bn	BN	(7, 7, 2048)	8192	conv5_block1_3_conv
conv5_block1_add	Add	(7, 7, 2048)	0	conv5_block1_0_bn conv5_block1_3_bn
conv5_block1_out	ReLU	(7, 7, 2048)	0	conv5_block1_add
conv5_block2_1_conv	Conv2D	(7, 7, 512)	1049088	conv5_block1_out
conv5_block2_1_bn	BN	(7, 7, 512)	2048	conv5_block2_1_conv
conv5_block2_1_relu	ReLU	(7, 7, 512)	0	conv5_block2_1_bn
conv5_block2_2_conv	Conv2D	(7, 7, 512)	2359808	conv5_block2_1_relu
conv5_block2_2_bn	BN	(7, 7, 512)	2048	conv5_block2_2_conv
conv5_block2_2_relu	ReLU	(7, 7, 512)	0	conv5_block2_2_bn
conv5_block2_3_conv	Conv2D	(7, 7, 2048)	1050624	conv5_block2_2_relu
conv5_block2_3_bn	BN	(7, 7, 2048)	8192	conv5_block2_3_conv
conv5_block2_add	Add	(7, 7, 2048)	0	conv5_block1_out conv5_block2_3_bn
conv5_block2_out	ReLU	(7, 7, 2048)	0	conv5_block2_add
conv5_block3_1_conv	Conv2D	(7, 7, 512)	1049088	conv5_block2_out
conv5_block3_1_bn	BN	(7, 7, 512)	2048	conv5_block3_1_conv
conv5_block3_1_relu	ReLU	(7, 7, 512)	0	conv5_block3_1_bn
conv5_block3_2_conv	Conv2D	(7, 7, 512)	2359808	conv5_block3_1_relu
conv5_block3_2_bn	BN	(7, 7, 512)	2048	conv5_block3_2_conv
conv5_block3_2_relu	ReLU	(7, 7, 512)	0	conv5_block3_2_bn
conv5_block3_3_conv	Conv2D	(7, 7, 2048)	1050624	conv5_block3_2_relu
conv5_block3_3_bn	BN	(7, 7, 2048)	8192	conv5_block3_3_conv
conv5_block3_add	Add	(7, 7, 2048)	0	conv5_block2_out conv5_block3_3_bn
conv5_block3_out	ReLU	(7, 7, 2048)	0	conv5_block3_add
avg_pool	GlobalAveragePooling2D	(2048)	0	conv5_block3_out
probs	Dense and Softmax	(1000)	2049000	avg_pool

Table D.3: VGG19 detailed Architecture [Simonyan et al. 2015]. The model contains 143667240 trainable parameters.

Layer Name for feature visualization	Layer Name for texture synthesis	Type	Output Shape	Param #
input	input	Input Layer	(224, 224, 3)	
block1_conv1	conv1_1	Conv2D	(224, 224, 64)	1792
block1_conv2	conv1_2	Conv2D	(224, 224, 64)	36928
block1_pool	pool1	MaxPooling2D	(112, 112, 64)	0
block2_conv1	conv2_1	Conv2D	(112, 112, 128)	73856
block2_conv2	conv2_2	Conv2D	(112, 112, 128)	147584
block2_pool	pool2	MaxPooling2D	(56, 56, 128)	0
block3_conv1	conv3_1	Conv2D	(56, 56, 256)	295168
block3_conv2	conv3_2	Conv2D	(56, 56, 256)	590080
block3_conv3	conv3_3	Conv2D	(56, 56, 256)	590080
block3_conv4	conv3_4	Conv2D	(56, 56, 256)	590080
block3_pool	pool3	MaxPooling2D	(28, 28, 256)	0
block4_conv1	conv4_1	Conv2D	(28, 28, 512)	1180160
block4_conv2	conv4_2	Conv2D	(28, 28, 512)	2359808
block4_conv3	conv4_3	Conv2D	(28, 28, 512)	2359808
block4_conv4	conv4_4	Conv2D	(28, 28, 512)	2359808
block4_pool	pool4	MaxPooling2D	(14, 14, 512)	0
block5_conv1	conv5_1	Conv2D	(14, 14, 512)	2359808
block5_conv2	conv5_2	Conv2D	(14, 14, 512)	2359808
block5_conv3	conv5_3	Conv2D	(14, 14, 512)	2359808
block5_conv4	conv5_4	Conv2D	(14, 14, 512)	2359808
block5_pool	pool5	MaxPooling2D	(7, 7, 512)	0
flatten	flatten	Flatten	(25088)	0
fc1	fc1	Dense	(4096)	102764544
fc2	fc2	Dense	(4096)	16781312
predictions	predictions	Dense and softmax	(1000)	4097000

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Transfert d'Apprentissage de Réseaux de Neurones à Convolution pour la Synthèse de Texture et la Reconnaissance Visuelle d'Images Artistiques

Mots clés: Réseaux de neurones à convolution; Transfert d'apprentissage; Synthèse de texture; Images artistiques; Apprentissage Faiblement Supervisé; Multi-résolution

Résumé: Dans cette thèse, nous étudions le transfert de réseaux de neurones à convolution (abrégés CNN en anglais) pré-entrainés sur des images naturelles, vers des tâches différentes de celles pour lesquelles ils ont été entraînés. Nous avons travaillé sur deux axes de recherche : la synthèse de texture et la reconnaissance visuelle dans les images d'œuvres d'art. Le premier axe consiste à synthétiser une nouvelle image à partir d'une image de référence. La plupart des méthodes récentes sont basées sur l'utilisation des matrices Gram des cartes de caractéristiques issues de CNNs entrainés sur ImageNet. Nous avons développé une stratégie multirésolution pour prendre en compte les structures à grande échelle. Cette stratégie peut être couplée à des contraintes à grande distance, soit par une contrainte basée sur le spectre de Fourier, soit par l'utilisation de l'autocorrélation des cartes de caractéristiques. Elle permet d'obtenir d'excellentes synthèses en haute résolution, tout particulièrement pour les textures régulières. Ces méthodes ont été évaluées de manière quantitatives et perceptuelles. Dans un second temps, nous nous sommes intéressés au transfert d'apprentissage pour la classification des images d'art. Les CNNs peuvent être utilisés comme des extracteurs de caractéristiques ou comme initialisation pour un nouvel entrainement. Nous avons mis en avant la supériorité de cette seconde solution. De plus, nous avons étudié le processus d'apprentissage à l'aide de visualisation de caractéristiques, d'indices de similarité ainsi que des métriques quantitatives. Nous avons aussi étudié le transfert de CNN entrainé pour de la détection d'objets. Nous avons proposé une méthode simple de détection faiblement supervisée (cad uniquement des annotations au niveau de l'image). Elle est basée sur un apprentissage à instances multiples, l'utilisation de caractéristiques figées et de propositions de boîtes issues d'un CNN pré-entrainés. Nous avons expérimentalement montré l'intérêt de nos modèles sur six jeux de données non photoréalistes.

Title: Transfer Learning of Convolutional Neural Networks for Texture Synthesis and Visual Recognition in Artistic Images

Keywords: Convolutional Neural Networks; Transfer Learning; Texture Synthesis; Art Images; Weakly-Supervised Learning; Multi-resolution

Abstract: In this thesis, we study the transfer of Convolutional Neural Networks (CNN) trained on natural images to related tasks. We follow two axes: texture synthesis and visual recognition in artworks. The first one consists in synthesizing a new image given a reference sample. Most methods are based on enforcing the Gram matrices of ImageNet-trained CNN features. We develop a multi-resolution strategy to take into account large scale structures. This strategy can be coupled with long-range constraints either through a Fourier frequency constraint, or the use of feature maps autocorrelation. This scheme allows excellent high-resolution synthesis especially for regular textures. We compare our methods to alternatives ones with quantitative and perceptual evaluations. In a second axis, we focus on transfer learning of CNN for artistic image classification. CNNs can be used as off-the-shelf feature extractors or fine-tuned. We illustrate the advantage of the last solution. Second, we use feature visualization techniques, CNNs similarity indexes and quantitative metrics to highlight some characteristics of the fine-tuning process. Another possibility is to transfer a CNN trained for object detection. We propose a simple multiple instance method using off-the-shelf deep features and box proposals, for weakly supervised object detection. At training time, only image-level annotations are needed. We experimentally show the interest of our models on six non-photorealistic.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France