Deep representation spaces

Micael Carvalho

Deep Representation Spaces

Espaces Profonds de Répresentation

Thèse de doctorat de
Sorbonne Université

Présentée par
Micael CARVALHO

Spécialité
Informatique
École Doctorale Informatique, Télécommunications et Électronique

devant le jury composé de :

Sébastien Lefèvre	Rapporteur
Frédéric Precioso	Rapporteur
Eric Gaussier	Examinateur
Hervé Le Borgne	Examinateur
Laure Soulier	Examinatrice
Nicolas Thome	Examinateur
Matthieu Cord	Directeur de thèse

Machine Learning & Deep Learning for Information Access

ABSTRACT

In recent years, Deep Learning techniques have swept the state-of-the-art of many applications of Machine Learning, becoming the new standard approach for them. The architectures issued from these techniques have been used for transfer learning, which extended the power of deep models to tasks that did not have enough data to fully train them from scratch. This thesis' subject of study is the representation spaces created by deep architectures.

First, we study properties inherent to them, with particular interest in dimensionality redundancy and precision of their features. Our findings reveal a strong degree of robustness, pointing the path to simple and powerful compression schemes.

Then, we focus on refining these representations. We choose to adopt a cross-modal multi-task problem, and design a loss function capable of taking advantage of data coming from multiple modalities, while also taking into account different tasks associated to the same dataset. In order to correctly balance these losses, we also we develop a new sampling scheme that only takes into account examples contributing to the learning phase, i.e. those having a positive loss.

Finally, we test our approach in a large-scale dataset of cooking recipes and associated pictures. Our method achieves a 5-fold improvement over the state-of-the-art, and we show that the multi-task aspect of our approach promotes a semantically meaningful organization of the representation space, allowing it to perform subtasks never seen during training, like ingredient exclusion and selection.

The results we present in this thesis open many possibilities, including feature compression for remote applications, robust multi-modal and multi-task learning, and feature space refinement. For the cooking application, in particular, many of our findings are directly applicable in a real-world context, especially for the detection of allergens, finding alternative recipes due to dietary restrictions, and menu planning.

RÉSUMÉ

Ces dernières années, les techniques d'apprentissage profond ont fondamentalement transformé l'état de l'art de nombreuses applications de l'apprentissage automatique, devenant la nouvelle approche standard pour plusieurs d'entre elles. Les architectures provenant de ces techniques ont été utilisées pour l'apprentissage par transfert, ce qui a élargi la puissance des modèles profonds à des tâches qui ne disposaient pas de suffisamment de données pour les entraîner à partir de zéro. Le sujet d'étude de cette thèse couvre les espaces de représentation créés par les architectures profondes.

Dans un premier temps, nous étudions les propriétés de leurs espaces, en prêtant un intérêt particulier à la redondance des dimensions et la précision numérique de leurs représentations. Nos résultats démontrent un fort degré de robustesse, pointant vers des schémas de compression simples et puissants.

Ensuite, nous nous concentrons sur le l'affinement de ces représentations. Nous choisissons d'adopter un problème multi-tâches intermodal et de concevoir une fonction de coût capable de tirer parti des données de plusieurs modalités, tout en tenant compte des différentes tâches associées au même ensemble de données. Afin d'équilibrer correctement ces coûts, nous développons également un nouveau processus d'échantillonnage qui ne prend en compte que des exemples contribuant à la phase d'apprentissage, c'est-à-dire ceux ayant un coût positif.

Enfin, nous testons notre approche sur un ensemble de données à grande échelle de recettes de cuisine et d'images associées. Notre méthode améliore de 5 fois l'état de l'art sur cette tâche, et nous montrons que l'aspect multitâche de notre approche favorise l'organisation sémantique de l'espace de représentation, lui permettant d'effectuer des sous-tâches jamais vues pendant l'entraînement, comme l'exclusion et la sélection d'ingrédients.

Les résultats que nous présentons dans cette thèse ouvrent de nombreuses possibilités, y compris la compression de caractéristiques pour les applications distantes, l'apprentissage multi-modal et multitâche robuste et l'affinement de l'espace des caractéristiques. Pour l'application dans le contexte de la cuisine, beaucoup de nos résultats sont directement applicables dans une situation réelle, en particulier pour la détection d'allergènes, la recherche de recettes alternatives en raison de restrictions alimentaires et la planification de menus.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Context

Machine Learning (ML), under the broader domain of Artificial Intelligence (AI), has recently gained the attention of the scientific community and of the private sector due to its newly-acquired capability of solving complex problems. Advances in computer hardware and software allowed techniques that estimate a huge number of parameters to be exploited in a feasible time frame, contributing to the expansion of this area of study.

The development of large-scale public datasets was an important step to this change in panorama. Notably, ImageNet [1], which contains over 14 million images, had a big impact on the development of Computer Vision (CV) algorithms. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al. 2015), containing a subset of the initial dataset, has been promoted yearly since 2010, with its last version taking place in 2017, and served as a showcase for state-of-the-art CV approaches.

After A. Krizhevsky et al. 2012's victory on the ILSVRC 2012 classification track, opening a big gap between them and the second best entry, as shown in Figure 1.1, a family of techniques gained the attention of these communities. These techniques are now commonly known as Deep Learning (DL) — a term coined by Dechter 1986 that gained popularity. On subsequent years of this competition, all the top entries were composed of deep architectures.

1. http://www.image-net.org

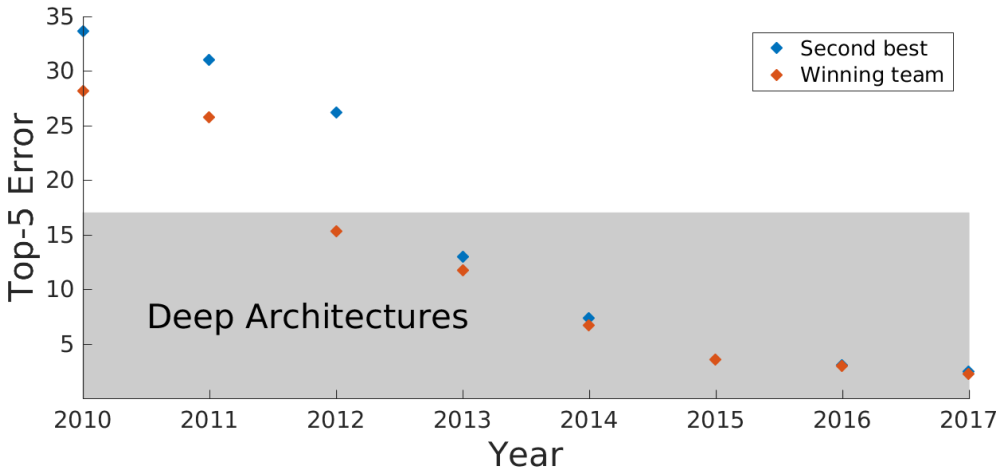


Figure 1.1 – **Best and second-to-best teams on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification** track. The big gap in performance between the winning entry and the second one observed in the year of 2012 mark the emergence of deep architectures.

CV is possibly the field that perceived the biggest advances in this recent DL sprout, with drastic changes in the diversity of techniques being employed. This phenomena drew the attention not only of the academic sphere, but also of the private sector that saw an opportunity to make use of the massive amounts of data at their disposal.

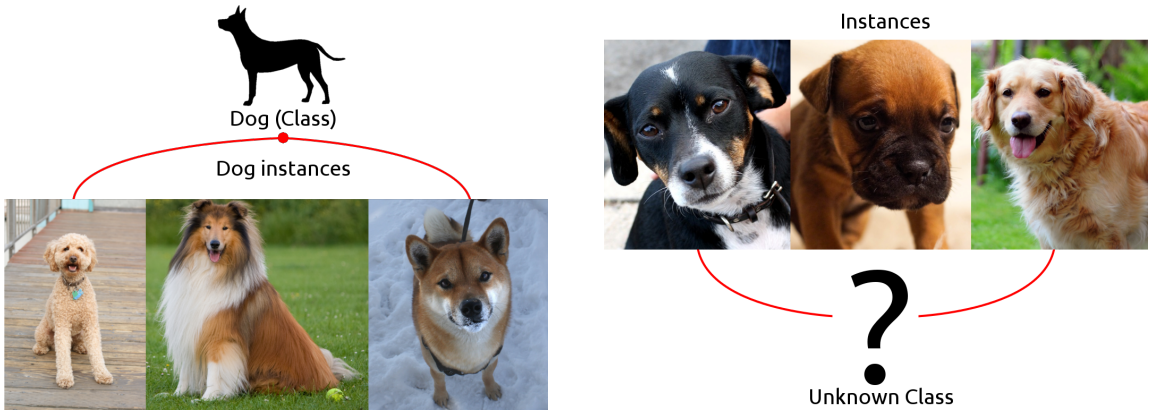
Facebook, for example, reported back in 2013 to receive 350 million new photos each day, with more than 250 billion photos already uploaded by that time [2]. Most of these images are processed to be correctly tagged, searched, and identified, and powerful algorithms are required to accomplish it.

With both the academic and the private spheres interested in the potential of these methods, the ML community was able to establish an interactive partnership between public and private sectors, both researching the subject and publicly publishing new findings.

Many of the challenges being explored by these researchers involve images, and they have been known and tackled by the CV community for the past years. Two of the most prominent ones are the classification and the retrieval tasks, both of them facing similar difficulties related to: occlusion, deformation, clutter, intra-class variation, illumination conditions, viewpoint and scale variation.

The classification task, possibly the most traditional one in the field, aims at finding corresponding labels (e.g human, train, computer) for a given image, as exemplified in Figure 1.2. In order to do so, one must be able to generalize knowledge about a specific label. For example, there are multiple breeds of dogs,

2. http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9?IR=T



(a) The dog class and some of its instances

(b) Instances from an unknown class

Figure 1.2 – **Challenges in the classification task.** The construction of a general model for identifying a given class is a complex task, demanding the ability to identify objects of interest while ignoring background and noise in the image. Figure 1.2a shows the dog class, and some of its instances. Figure 1.2b exemplifies the task of learning *what is a dog*, from a set of images limited by camera perspective and other factors.

and they look different, but they are all dogs — then *how can an AI system learn the macro concept of what is a dog from a limited set of samples?* Until recently, the standard way of achieving this was by creating a numerical description of each image with a *feature extractor*—and many of these have been proposed in the scientific literature—, then applying a *classifier* to discriminate them.

The problem then is to find such a feature extractor that is capable of capturing all the important traits we need. For example, in classical CV, the Bag-of-Words (BoW) representation was used to numerically describe an image; as well as the Object Bank representation (Li et al. 2013), which uses object detectors to describe an image. These descriptors are given to a classifier which makes the final decision. In this schema, the object detectors do not have to be necessarily the same of the classes being detected, since complementary features between those classes and the ones from the detectors may be used to identify the correct label. This scheme is very similar to modern transfer learning approaches, the difference being that the latter can take advantage of deep networks as feature extractors.

For modern classification tasks, instead of using hand-crafted feature extractors for creating a numerical description of the images, a DL model is adopted (for image, usually a Convolutional Neural Network (CNN)), unifying the feature extraction and classifier into a single framework capable of doing it all at once. However, the transfer learning approach we previously mentioned challenges this concept of all-in-one architecture by showing that a trained CNN can generate features that are not exclusive to its known classes.

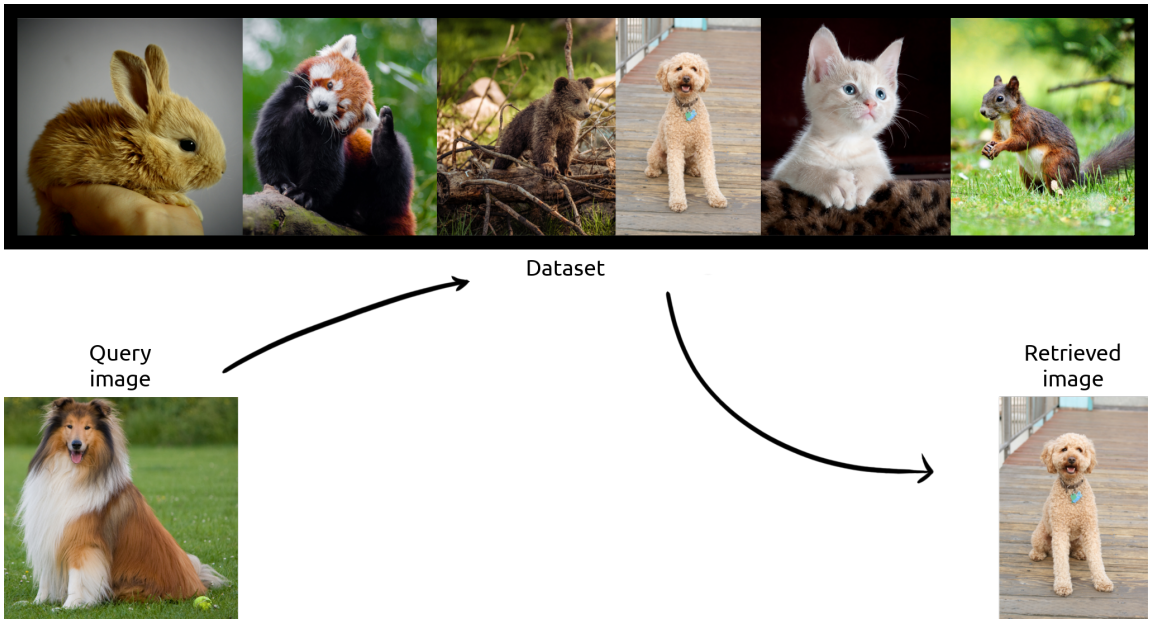


Figure 1.3 – **The image retrieval task.** The goal in the image retrieval task is to find, among a set of candidates, the closest image from a given query. In this example, we are looking for a dog among the images. Note that the query and the retrieved images need not be the same.

The classification task has many applications in the real world. One of the first known cases for the image classification is associated to networks like LeNet5 (LeCun et al. 1998), able to identify handwritten digits, and widely used in post office for automatically routing packages and letters. Another example is present in modern phones and cameras that include a smart photo function capable of detecting the type of scene being shot and then adjusting the image acquisition parameters properly.

As for the retrieval task, and in particular content-based image retrieval, the goal is to find an image—traditionally, the most similar one among a set of images—, given another one. The retrieval task can, however, take many different forms, as we will see throughout this thesis, but it always follows the pattern of "find *(something)* based on *(something else)* using *(this criterion)*". Figure 1.3 depicts a simple image retrieval task, where the objective is to retrieve an image based on the class similarity with the query.

More advanced flavors of the retrieval task deal with multi-modal data, creating common numerical representation spaces where the modalities can be compared. Inside of these spaces, each modality (e.g. image, text, sound) is represented as a point, and the vector containing the coordinates of these points are the *embeddings* of the items they represent.

Just like the classification, the retrieval is also present in many of our daily activities, such as: singing part of the lyrics of a song in order to find its name (e.g.

Shazam [3], SoundHound [4]), searching for visually similar images in search engines (e.g. Google [5], Bing [6]), or even taking pictures of a dish and looking for visually similar recipes (e.g. VISIIR [7]). The multi-modal retrieval also has its place in this list, as most of the image searches through search engines and cloud-storage services are based on textual description, and the system must be able to make the connection between what the user typed and the images of interest—in this case, the query is a text, and the result is an image.

The idea behind transfer learning is to reuse parameters from a pre-trained network. Training such networks from scratch can be very costly in terms of data and processing power, therefore reusing part of the knowledge from a pre-trained network can reduce the data greediness for a new task. For this particular reason, transfer learning unleashed the power of deep networks for small and mid-sized tasks.

In this context, transfer learning is usually done by adopting a network trained on a large-scale dataset, and then taking feature vectors from output of one of its intermediate layers. Later, these vectors can be used in different ways: (1) for classification, a new classifier is trained with them. This classifier can take the form of another layer in the network, specialized on the new task, or of something else, like Support Vector Machines (SVMs). (2) however, for the retrieval task, if one assumes these vectors already encode several invariances learned by the network, it can be directly adopted as the embedding of the image.

These new tasks are trained using *cost functions*, also known as *loss functions*. They are responsible for measuring how much the result obtained differs from the one expected, and usually these functions are derivable—and in this case, this *error* can be propagated throughout the network to find out how much each parameter contributed to the mistake. This information is present in the *gradient*, which is the first order derivatives of the loss function with respect to the weights of the network.

The transfer strategy allows the gradient to correct the weights in the part of the network specialized to the new task, while keeping the pre-trained model untouched. But to improve even more their performance, fine-tuning strategies can be adopted. These consist in allowing the gradient of the cost function to propagate back through the pre-trained network, usually with a lower weighting factor, instead of being constrained exclusively to the new task. This allows the whole architecture to adapt to the new task, removing uninformative features and possibly creating new ones.

3. https://www.shazam.com/
4. https://soundhound.com/
5. https://www.google.com/
6. https://www.bing.com/
7. http://visiir.lip6.fr/

1.2 Motivations

One concern that raised from the adoption of these new strategies is the explainability and understanding of the inner workings of these CNNs. Some researchers tried to tackle this problem by analyzing which part of the image the network was paying more attention to, while others tried to enforce a meaning while designing the architecture of the network, making sure it was interpretable.

The power of these architectures have also been exploited with transfer learning, which consists in reusing part of the knowledge learned by the model in another task. However, important aspects of the representation constructed by these networks did not receive much attention, in particular their redundancy, compactness, and robustness.

By adopting a transfer learning scheme we are able to measure the performance of a deep network in different tasks, which leads us to study, in the first part of this manuscript, the three properties we mentioned with the help of something we call a *stress framework*, responsible for interfering with the knowledge being transferred.

Then, we focus on learning these representations, with particular interest in ways of refining the knowledge they hold. Many strategies were developed in attempt to improve the way they are learned, but they usually rely on intuitions that are either very specific to a dataset, or that are based on assumptions that are not always true to all tasks.

We study, on a second part of this manuscript, ways to better construct theses spaces, allowing them to be rich in information. For this purpose, we propose a metric learning approach that can take advantage of instance-wise labeling, commonly used for retrieval, as well as class-based labeling, commonly used for classification.

Using two different types of data in a learning strategy is usually done with multi-task learning. In this perspective, our approach is, indeed, multi-task. However, contrary to standard strategies, we are would like to directly introduce the constraints for each task into the space created by the network, instead of adding specific parts responsible for each of them.

We are also interested in measuring the impacts of our strategies in a specific scenario with multi-modal data. In particular, we want to know if the usage of a multi-task strategy can improve results for applications from the information retrieval task. Therefore, in a third part of this manuscript, we test if these spaces exhibit the properties we sought — the instance-based and the class-based information, while also studying the general structure generated by the way the representations were constructed.

1.3 Contributions and Outline

The contributions of this work are three-fold:

1. Chapter 2: EXPLORING REPRESENTATION SPACES
   In the first part of this manuscript, we introduce our stress framework, capable of accessing properties of the feature spaces generated by a Deep Convolutional Neural Network (DCNN). It consistently interfere in the network in order to selectively destroy information.
   Although recent studies reevaluate deep architectures with respect to the size and precision of their representations, their primary focus are practical impacts upon the original tasks. Our framework is designed for transfer learning tasks and, as we try to shed light on general properties of the networks, we observe a strong degree of resiliency and redundancy in their features, opening the opportunity to create powerful compact descriptors.
   The stress framework is designed to provide a better understanding of deep feature spaces. We develop tools to measure their redundancy and resiliency to deformations, our findings open new possibilities for powerful compression schemes.
   It works by first extracting features using a pre-trained CNN, and then applying different transformations to these features, measuring how the classification performance is affected by them. This approach allows us to access specific characteristics inherent to the representation spaces constructed by these networks, while also providing us with insights about its inner workings.
2. Chapter 3: LEARNING MULTIMODAL LATENT SPACES
   For the second part, we present a new constrained learning scheme called AdaMine.
   When tackling multi-modal problems, a common approach is to have an architecture with parts specialized to different tasks associated with the modalities of the problem. However, we are interested in having an unified framework, capable dealing with modalities without adding extra parameters to the model, nor learning specific subtasks.
   Departing from a pre-trained model, AdaMine's goal is to refine the feature space with a fine-tuning based approach. It aligns images and texts descriptors for applications related to cross-modal retrieval directly in the representation space, without adding parts to the model.
   Its first part involves the combination of semantic- and instance-based information in an unified loss, and its second part encompasses solutions for selecting informative triplets in a large scale optimization context.
   The resulting method is an adaptive strategy for negative mining, which provides faster and more stable convergence when compared to other ap-

proaches. We evaluate it on the large and challenging Recipe1M cross-modal dataset, outperforming the state-of-the-art models by 5-fold improvement.

3. Chapter 4: RETRIEVAL IN THE COOKING CONTEXT

   Finally, in a third part, we perform an extensive qualitative study of the spaces created by AdaMine.

   We show that with our new constraints, the semantic space gain properties that can be exploited in order to obtain specific information inside of it. We provide a broader analysis of the feature space, experimentally showing that because our loss introduces semantic information into it, we are able, for example, to retrieve recipes related to a specific ingredient, or to search recipes that do not contain specific ingredients. This is a strong indicator that the learning strategies we adopted are powerful and flexible enough to extrapolate to broader tasks.

   This discussion led to a better understand of the feature spaces constructed with the additional semantic information, which acts as a global regularizer to these spaces. Many applications in the domain of the computational cuisine can be derived from this kind of approach, such as finding recipes with alternatives to ingredients that provoke allergy or intolerance, as well as general recipe retrieval from pictures.

1.4 Related publications

The following publications are included in parts or in an extended version in this thesis:

— Micael Carvalho, Matthieu Cord, Sandra Avila, Nicolas Thome, and Eduardo Valle (2016). "Deep Neural Networks Under Stress". In: *IEEE International Conference on Image Processing (ICIP)*

— Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord (2018). "Cross-Modal Retrieval in the Cooking Context: Learning Semantic Text-Image Embeddings". In: *The ACM conference on Research and Development in Information Retrieval (SIGIR)*

— Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, and Matthieu Cord (2018). "Images & Recipes: Retrieval in the cooking context". In: *IEEE International Conference on Data Engineering (ICDE), Data Engineering meets Intelligent Food and Cooking Recipe (DECOR) workshop*

CHAPTER 2

EXPLORING REPRESENTATION SPACES

Contents

Chapter abstract

In recent years, deep architectures have been used for transfer learning with state-of-the-art performance in many datasets. The properties of their features remain, however, largely unstudied under the transfer perspective. In this chapter, we present an extensive analysis of the resiliency of feature vectors extracted from deep models, with special focus on the trade-off between performance and compression rate. We show that deep features are more robust to disturbances than classical approaches, and our findings are a strong starting point to develop techniques related to network compression and mimic learning.

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

— Micael Carvalho, Matthieu Cord, Sandra Avila, Nicolas Thome, and Eduardo Valle (2016). "Deep Neural Networks Under Stress". In: *IEEE International Conference on Image Processing (ICIP)*

2.1 Introduction

As discussed in Chapter 1, Deep Convolutional Neural Networks (CNNs) have swept the Computer Vision (CV) community, with state-of-the-art performance for many tasks (A. Krizhevsky et al. 2012; He et al. 2015; Durand et al. 2016), going as far as being applied to other domains, including linguistic analysis (e.g. Vanni et al. 2018). However, an analytical understanding of their models is still lacking, shrouding their use under a cloud of ad hoc procedures — tricks of the trade — without which they simply fail to work. Therefore, a full understanding of deep representations became the new Holy Grail of research in Machine Learning and Computer Vision (Bruna et al. 2013; Y. LeCun et al. 2015).

We explore in this chapter the properties of Deep Networks, measuring to which extent they preserve discriminative information about the input, i.e., measuring the robustness of the feature vectors they generate. Indeed, we may understand a deep model as one that first learns to extract a good representation (feature extraction step) and then uses that representation to make a decision (classification or regression step). Most of the challenge in understanding deep models is due to the unknown nature of the learned features.

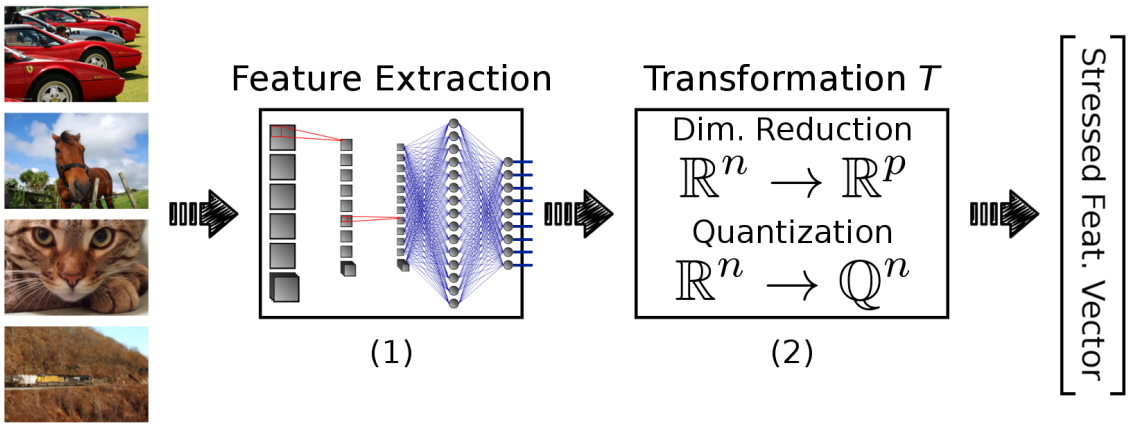


Figure 2.1 – **Overview of our stress framework.** Input images are converted to stressed feature vectors by: (1) extracting descriptions using a pre-trained deep network, (2) transforming/stressing the feature vectors by reducing their precision or their number of dimensions.

Our main objective is to investigate the VGG-M deep convolutional model (Chatfield et al. 2014), originally trained on ImageNet Large Scale Visual Recognition Challenge (ILSVRC), in a transfer scheme for the classification task of the PASCAL Visual Object Classes (PASCAL VOC) 2007 dataset (Everingham et al. 2010). Our proposal is a stress framework, represented in Figure 2.1, which consistently interfere in the network to selectively destroy information.

We explore two important aspects of deep architectures: dimensionality and numerical precision of their representations. Dimensionality stress tests reduce the dimension of the representations generated by the network, giving us insights about the redundancy and co-adaptation of features, and quantization stress tests quantize them by limiting the set of their possible values, showing if there are unnecessary details in them. We also combine the two stresses to measure their overlap and complementarity, and perform extensive experiments to study the robustness of this architecture.

This chapter is organized as follows:

— In Section 2.2 we discuss classical transfer schemes for Deep Convolutional Neural Networks (CNNs). The vanilla strategy is adopted in the proposals presented in this chapter, and the fine-tuning approach is used in all chapters of this thesis;

— We then present in Section 2.3 the stress framework used to probe the network we study, with its two main parts that include the dimensionality reduction and the quantization. Those are responsible for measuring the resiliency and redundancy in deep representations used for transfer learning;

— The experimental setup, as well as the results we obtain are presented in Section 2.4. We show that due to the properties we measure, it is possible to obtain a high level of compression while keeping good performance [1];

— In light of the results we obtained, in Section 2.5 we discuss network compression schemes, and devise a way of taking advantage of our findings with the stress framework in order to improve mimic learning [1];

— Finally, in Section 2.6 we discuss results obtained in this chapter, and point perspectives for future work related to feature and network compression schemes.

2.2 Transfer Strategies

Although recent studies reevaluate deep architectures with respect to the size and precision of their representations (e.g. Vanhoucke et al. 2011; M. Courbariaux, Bengio, and J.-P. David 2015; M. Courbariaux, Bengio, and J. David 2015; Judd et al. 2015), their primary focus are practical impacts upon the original tasks. Our framework is designed for transfer learning tasks and, as we try to shed light on general properties of the networks we study, we will see that they show a strong degree of redundancy, opening the opportunity to create powerful compact descriptors and to explore efficient network compression strategies.

Transfer learning consists in recycling knowledge from one model to another, in the form of model weights, initialization, or architecture, saving both com-

1. The source code the experiments presented in this part, as well as high resolution figures, is available online at https://github.com/MicaelCarvalho/DNNsUnderStress

putational resources and training data. Although it is a classical approach that can be adapted to most learning systems, recently it has been used, with great success, on deep models that are very greedy in terms of data and processing power (Azizpour et al. 2016). We are particularly interested in modern strategies, which usually adopt a pre-trained CNNs as the initial network.

The number of evidence for the efficacy of these approaches piles up, as many small tasks have their state-of-the-art scythed by deep networks. This was particularly evident when Razavian et al. 2014 showed how their CNNs achieved better or competing results in many tasks, going as far as saying their conclusions are that DL with CNNs should be considered the primary candidate in *essentially any visual recognition task*.

One of the biggest reasons for adopting transfer learning is to reduce the data-greediness of the model. In CV, it is common to adopt networks trained on the ILSVRC dataset, which was financially expensive and took years to fully annotate. However, in a simple classification task, one could just reuse such a model and re-train a classification layer using a much smaller dataset.

Transfer learning is a fundamental concept for this chapter, and the transferability of features issued from deep networks has been studied before (see Yosinski et al. 2014). Therefore we present in the following two of the main strategies for performing it. For a larger number of variations and techniques, we refer the reader to the works of Pan et al. 2010 and Weiss et al. 2016, who conduct a broader study on the topic.

Vanilla transfer learning A straightforward scheme to perform transfer learning is to choose a pre-trained network, freeze its weights up to a certain layer, and to introduce and train new layers for the new task, this setup is depicted in Figure 2.2. By picking different layers from the original network, one controls the degree of transfer between the models. Conceptually, the output of the frozen transferred layers for any image may be seen as a feature vector, thus any classifier may be used for classification on a target dataset.

This kind of approach is adopted by Chevalier et al. 2015, who use a multi-class linear SVM on two fine-grained datasets: FGVC Aircraft (Maji et al. 2013), for airplane classification, and the challenging PPMI (Yao et al. 2010), in which the task is to distinguish between people playing instruments and people merely holding them. Similarly, Durand et al. 2015 adopt a deep network for extracting visual features, this time from different datasets, classifying with their modified SVM called MANTRA.

Fine-tuning More advanced strategies perform, however, the fine-tuning of features. This method, represented in Figure 2.3, consists in allowing the gradient to be propagated through the copied network, often with a smaller learning rate,

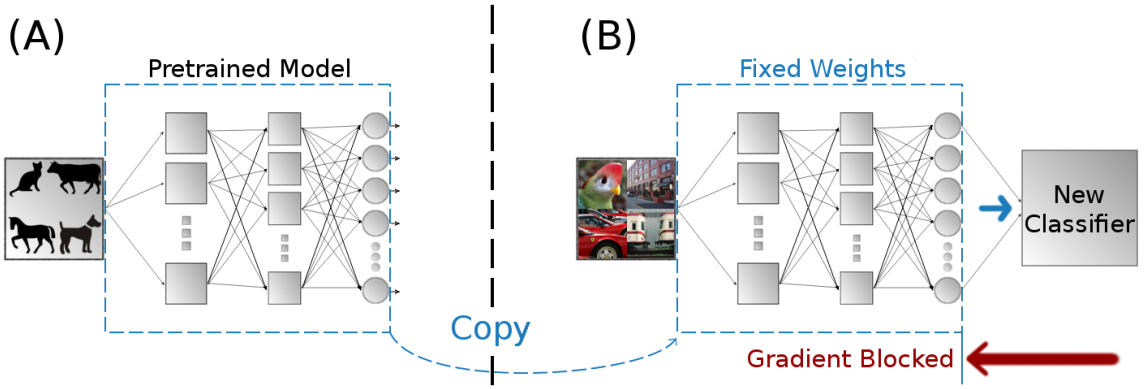


Figure 2.2 – **The vanilla transfer learning scheme.** Part of the weights of a model (A) that has been pretrained for a specific task are used, and a classifier is trained for a new task (B). No gradient is backpropagated through the weights of the pretrained model, therefore only the new classifier is learned.

allowing the weights to slightly adapt to the new task, and to remove information no longer useful for the new objective.

An interesting study conducted by Yosinski et al. 2014 question the transferability of features from deep Artificial Neural Networks (ANNs). They split the 1000 ILSVRC classes into two groups, each containing 500 classes and approximately half of the data. Two networks are then trained, one for each half. Later, a network is picked and part of its layers are retrained for both tasks, yielding two new networks. If they perform similarly, there is evidence that the features from the chosen layers are general.

Contrary to Razavian et al. 2014, one of Yosinski et al. 2014's goals is to test the effect of fine-tuning on the transfer. In order to do so, they recreate the same networks, with the same architecture, but by adopting fine-tuning instead of vanilla transfer learning. Their results show that fine-tuning can recover co-adapted interactions between layers, reducing drastically the damages of the transfer process. Furthermore, their architectures that were fine tuned achieved better performance than the ones originally trained exclusively on one half of the data, indicating that the transfer learning associated to fine-tuning can improve generalization, since the network is able to *see* more images.

In our pursuit of a better understanding of the unknown nature of features extracted from deep architectures, we use transfer learning and "stress" tests to probe deep ANNs, presented in the following section.

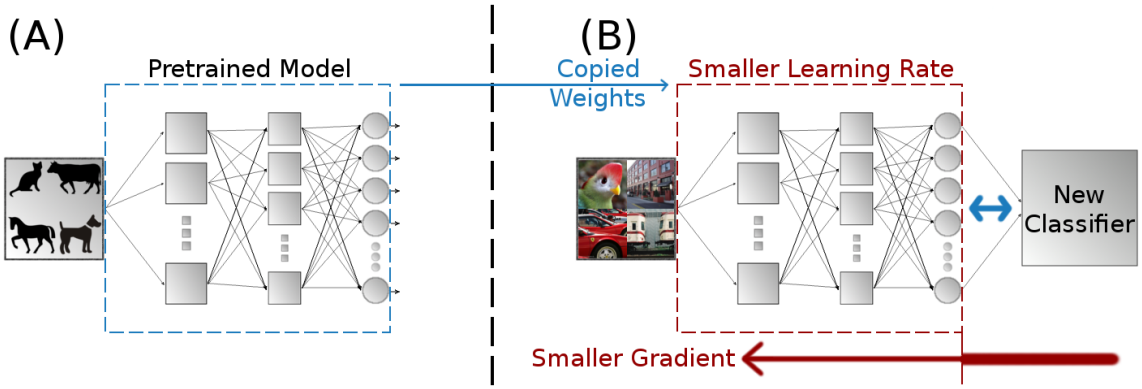


Figure 2.3 – **Transfer learning with fine-tuning.** Part of the weights of a model (A) that has been pretrained for a specific task are used, and a classifier is trained for a new task (B). The gradients of the new classifier are propagated through the weights of the pretrained model. Contrary to the vanilla transfer learning scheme, fine-tuning allows the network to adjust all of its weights, including the copied ones.

2.3 Stress Framework

Our proposition is complementary to, and built upon the studies conducted by Razavian et al. 2014, Yosinski et al. 2014. We adopt VGG-M (Chatfield et al. 2014) as base architecture, but to better highlight inherent properties of deep models, instead of specific characteristics of this model, we also evaluate part of our experiments with GoogLeNet (Szegedy et al. 2015). Furthermore, in order to differentiate these deep models from classical approaches, we also report comparative results with the Bag-of-Words (BoW)'s model BossaNova (Avila et al. 2013). In all cases, we pre-process the images according to each model's recommended protocol.

Let us formalize the pre-trained deep model as a series of functions $\phi_i : \mathbb{R}^{m_i} \rightarrow \mathbb{R}^{n_i}$, where ϕ_i is the i^{th} layer of the network, m_1 is equal to the dimensionality of the input data and $n_i = m_{i+1}$ is the output of such layer. In our stress tests, we choose a layer i up to which we freeze the network (i.e., we keep layers $\phi_1 ... \phi_i$ untouched). At first, we use the output of layer ϕ_i to train an SVM. Then, we pick a stressing function T and retrain the model using $T(\phi_i)$ as input. Comparing the two scores, we can infer the network's resiliency to the chosen stress.

Our stressing functions are divided in two groups: dimensionality reduction and quantization, described in the following subsections. For the first, we are trying to answer the question: "Instead of having feature vectors of 4096 dimensions, can we have, for example, 800 while keeping the same performance?". As for the second group, we focus on the question: "Can we reduce the precision of the

values in the feature vector and keep the same scores? For example, by only using 4 bits instead of 32."

2.3.1 Dimensionality Reduction (DR)

In order to understand how redundant is the deep representation, the first stress tests drop dimensions from the feature vector. The number of dimensions p_i preserved at each test step $1 \leq i \leq 20$ is proportional to the initial size n of the feature vector, following $p_i = \left\lfloor \frac{n * (21 - i)}{20} \right\rfloor$.

We contrast two strategies for selecting which $p_{i-1} - p_i$ dimensions should be dropped at each step i: $\mathbf{T_{DR\text{-}1}}$ randomly drops them, from the ones left; and $\mathbf{T_{DR\text{-}2}}$ uses a PCA-based strategy, that discards the dimensions encoding less variance and therefore being less informative for a simple classifier. To take in consideration the random choice in DR-1, we repeat the experiment 10 times.

2.3.2 Quantization (Q)

Algorithm 2.1 Creating a new language for the quantization process. j is the minimum and k is the maximum value observed in the training set, and n is the number of entries in the dictionary.

```
1: function CREATE_LANGUAGE(j, k, n)
2:     step ← (k − j) / n
3:     value ← j + step / 2
4:     new_language ← [ ]
5:     while value < k do
6:         append value to new_language
7:         value ← value + step
8:     end while
9:     return new_language
10: end function
```

The other stressor diminishes the numerical precision of the representation, quantizing the feature vectors. Our objective is not to explore advanced quantization strategies here, but to consider 2 fast and simple scalar quantizations and to analyze their effect on a classification task. In our first one, **Q-1**, all dimensions are quantized in the same $h \in [1, 30]$ regular intervals, using the minimum (min)

Algorithm 2.2 Quantization of the feature vector for the Q-1 strategy

```
1: function QUANTIZE_FEATURE_VECTOR(feature_vector, new_language)
2:     fsize ← size(feature_vector)
3:     lsize ← size(new_language)
4:     quantized_features ← feature_vector
5:     for i = 1 to fsize do
6:         winner ← 1
7:         distw ← feature_vector[i] − new_language[winner]
8:         for j = 2 to lsize do
9:             distj ← feature_vector[i] − new_language[j]
10:            if |distj| < |distw| then
11:                winner ← j
12:                distw ← distj
13:            end if
14:        end for
15:        quantized_features[i] ← new_language[winner]
16:    end for
17:    return quantized_features
18: end function
```

and maximum (max) scalar values observed in the training set for all dimensions. In our second one, **Q-2**, we adapt the limits for each dimension individually, according, again, to values observed in the training set.

Let $\mathbf{x}$ be the feature matrix of the training feature vectors. Formally, Q-1, using the global step $st = \frac{max(x)-min(x)}{h}$, has a single dictionary $\mathcal{H}$, generated by

$$\mathcal{H} = \left\{ \left(min(x) + \frac{st}{2} \right) + st * i \mid \ 0 \leq i < h \right\}$$

For Q-2, let $\mathbf{x}_t$ the t^{th} element from all the vectors in $\mathbf{x}$. Using one step $st_t = \frac{max(\mathbf{x}_t)-min(\mathbf{x}_t)}{h}$ per dimension, Q-2 has n (number of dimensions) dictionaries, generated by

$$\mathcal{H}_t = \left\{ \left(min\left(\mathbf{x}_t\right) + \frac{st_t}{2} \right) + st_t * i \mid \ 0 \leq i < h \right\}$$

Finally, in the quantization step, we assign to each element the value of the closest point in the dictionary. For Q-1 and Q-2, respectively, this is defined by:

$$T_{Q\text{-}1}(\mathbf{x}_{ij}) = \arg\min_y \ \left\{ abs\left(\mathbf{x}_{ij} - y\right) \mid y \in \mathcal{H} \right\}$$

$$T_{Q\text{-}2}(\mathbf{x}_{ij}) = \arg\min_y \ \left\{ abs\left(\mathbf{x}_{ij} - y\right) \mid y \in \mathcal{H}_j \right\}$$

Algorithm 2.3 Quantization of the feature vector for the Q-2 strategy

```
1:  function QUANTIZE_FEATURE_VECTOR(feature_vector, new_language)
2:      fsize ← size(feature_vector)
3:      quantized_features ← feature_vector
4:      for i = 1 to fsize do
5:          winner ← 1
6:          distw ← feature_vector[i] − new_language[i][winner]
7:          lsize ← size(new_language[i])
8:          for j = 2 to lsize do
9:              distj ← feature_vector[i] − new_language[i][j]
10:             if |distj| < |distw| then
11:                 winner ← j
12:                 distw ← distj
13:             end if
14:         end for
15:         quantized_features[i] ← new_language[i][winner]
16:     end for
17:     return quantized_features
18: end function
```

We offer, in Algorithm 2.1, an algorithm for the process of creating these dictionaries. For Q-1, this algorithm is called once, with j being the minimum and k the maximum observed values throughout all feature vectors in the training set; for Q-2, this algorithm is called once for each position of the feature vector, with j being the minimum and k the maximum values observed for that position.

The quantization is then performed using the dictionaries we just calculated. Because Q-1 and Q-2 have different strategies, the first acting globally on the feature vector, and the second acting locally, at each position of them, the process for quantizing the features is different for them. We detail the quantization procedure in Algorithm 2.2 for Q-1, and in Algorithm 2.3 for Q-2.

2.3.3 Feature Compression (FC)

The final experiment **FC**, applies both stressors $\mathbf{T_{DR-2}}$ and $\mathbf{T_{Q-2}}$ simultaneously, dropping dimensions of the feature vector and quantizing the values of the remaining elements. Our goal is to measure any cross-effects between **DR-2** and **Q-2**.

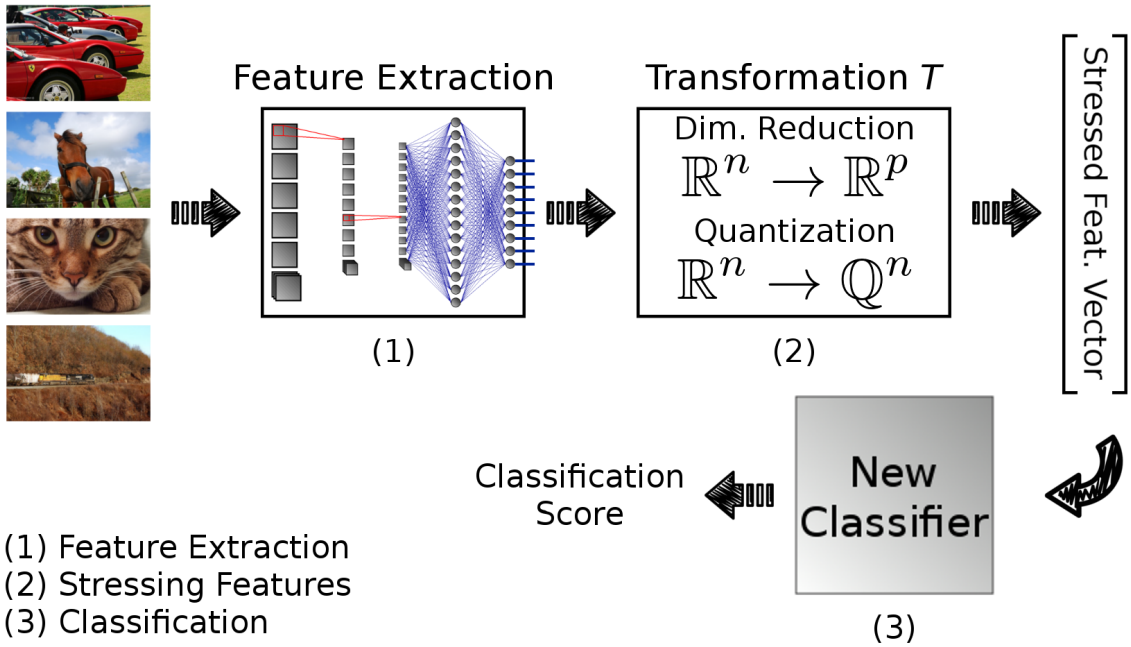


Figure 2.4 – **Overview of our experimental setup for the stress framework.** Input images are converted to stressed feature vectors by: (1) extracting descriptions using a pre-trained deep network, (2) transforming/stressing the feature vectors by reducing their precision or their number of dimensions. These vectors are fed to a classifier. The performance of the classifier is measured with and without our stressor, and the difference is used to quantify the impact of the transformations.

2.4 Experiments

As explained, for a given experimental point, we freeze a pre-trained network at layer ϕ_i, discarding all upper layers. We then pick a stressing function T, and use the output of $T(\phi_i)$ as a feature vector in a transfer learning classification task. We ℓ_2-normalize those feature vectors, and feed them to a linear SVM model [2] (Fan et al. 2008), measuring the model's scores for different choices of T. This full pipeline is represented in Figure 2.4. By picking stressing functions of different kinds and intensities (including the identity $T(x) = x$) we gain insight on the resiliency of deep models to those stresses.

Our base setup is composed of the VGG-M network (Chatfield et al. 2014), detailed in Table 2.1, and the PASCAL Visual Object Classes (PASCAL VOC) 2007

2. For all setups, we use a regularization parameter $C = 1$; preliminary experiments shown very little variation when the C was cross-validated.

	G 1	G 2	G 3	G 4	G 5	G 6	G 7	G 8
Conv.	L 1	L 5	L 9	L 11	L 13	–	–	–
Fully	–	–	–	–	–	L 16	L 18	L 20
ReLU	L 2	L 6	L 10	L 12	L 14	L 17	L 19	–
LRN	L 3	L 7	–	–	–	–	–	–
Pooling	L 4	L 8	–	–	L 15	–	–	–
Softmax	–	–	–	–	–	–	–	L 21

Table 2.1 – **The layers of the VGG-M model.** Description of layers (L) and groups (G) of the VGG-M model (Chatfield et al. 2014), from the MatConvNet toolbox (Vedaldi et al. 2015). *Conv.* indicates a convolutional layer, *Fully* a fully connected layer, *ReLU* a Rectified Linear Unit layer, *LRN* a Local Response Normalization layer, *Pooling* a Max Pooling layer and *Softmax* the activation of the Softmax function.

dataset (Everingham et al. 2010). We explore how to extend the results obtained for this dataset by comparing part of the experiments with two other: MIT-67 – Indoor (Quattoni et al. 2009) and UPMC Food-101 (X. Wang et al. 2015), containing 67 and 101 classes, respectively. The classification scores are reported in mean Average Precision (mAP) for PASCAL VOC 2007, and Accuracy (ACC) for Food-101 and MIT-67, following literature's tradition on those datasets. We adopt a simplified BossaNova's pipeline, without the concatenation with the classic Bag of Visual Words, and a linear SVM. Furthermore, we also evaluate part of our experiments with GoogLeNet (Szegedy et al. 2015) and, in order to differentiate these deep models from classical approaches, we also report comparative results with the Bag-of-Words (BoW)'s model BossaNova (Avila et al. 2013). In all cases, we pre-process the images according to each model's recommended protocol.

Table 2.2 shows the baseline scores for our experiments, using setups without perturbating the feature vectors (i.e., $T(x) = x$). Because our objective is to measure the individual impacts of our strategies, all of our analyses are performed with respect to these scores — for a given experiment, we report how much of the *original score* is achieved. For example, if, for a given stress test, we report 100% for a VGG-M model with MIT-67, this represents a real score of 63.35%. In other words, the stressor had no effect on the baseline score. Unless otherwise stated, for all experiments we report results obtained using the penultimate fully connected layer for each network, as this is the standard transfer learning protocol.

	VGG-M	GoogLeNet	BossaNova
PASCAL VOC 2007 (mAP)	76.95%	80.58%	51.02%
MIT-67 Indoor (ACC)	63.35%	–	–
UPMC Food-101 (ACC)	46.22%	–	–
Feature Dimensionality	$4 * 10^3$	$5 * 10^4$	$6 * 10^4$

Table 2.2 – **Classification scores for different architectures (VGG-M, GoogLeNet, and BossaNova) and datasets (PASCAL VOC 2007, MIT-67, and UPMC Food-101).** We show scores for deep and BoW strategies in a vanilla transfer scheme, with a linear SVM as classifier.

2.4.1 Dimensionality Reduction (DR)

The objective of this first round of experiments is to access the amount of redundant information in the descriptors extracted from the tested models. In order to do so, we directly measure their performance under a dimensionality reduction stressor. If we obtain similar scores after removing many dimensions of the feature vector, the SVM classifier was able to recover most of the important traits for the classification task from the remaining dimensions, revealing the redundancy in the initial features.

Base dataset To begin, we apply this stressor to all architectures using our base dataset PASCAL VOC 2007. We present the results for this test in Figure 2.5, where we observe strong redundancy on all the tested representations. This is indicated by the thin shadowed areas around each curve, which represent the variation across multiple runs using the same setup.

In this test, GoogLeNet was the most robust against the random dimensionality perturbation (DR-1), with an average mAP drop of 4.74% for 95% of the dimensions removed. However, GoogLeNet with its 2508 dimensions when 95%-compressed is, from start, 12-times bigger than CNN-M, with 204 dimensions under the same compression rate. Considering a direct comparison of descriptions of approximately the same size, the scores of the two models were equivalent.

Although BossaNova has shown similar resiliency to dimensionality reduction with respect to VGG-M, the latter held better scores for every test point, despite having feature vectors 15-times smaller — 3172 dimensions for BossaNova with a 95% compression rate, versus 204 for VGG-M. The 95%-reduced dimensionality for each model is indicated on the right side of Figure 2.5.

The PCA-based strategy (DR-2) was very effective for preserving information while dropping dimensions, as shown in Figure 2.6. Despite having the same dimensionality at every test point, DR-2 held 97.95% of the original mAP when

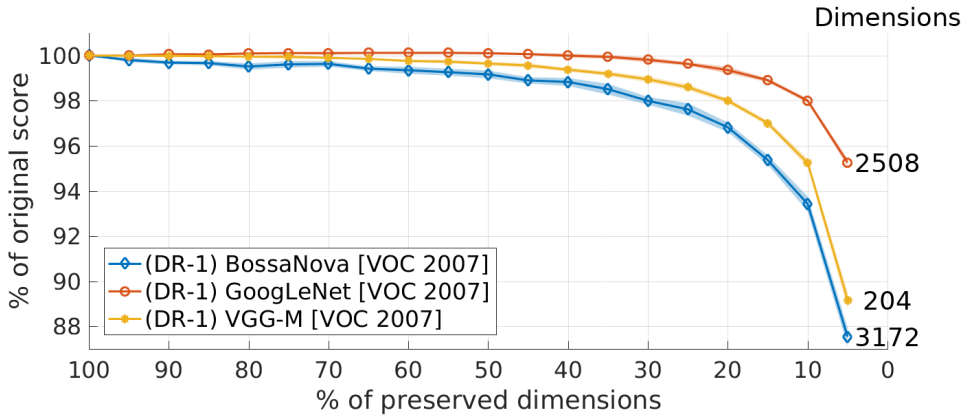


Figure 2.5 – Results for **dimensionality reduction (DR) on PASCAL VOC 2007** with standard deviation (shaded regions around the lines). In the horizontal axis, each value indicates the percentage of the original dimensions that is kept, while the corresponding score, with the respect to the initial one, is shown vertically. On the right side of the figure, we show the number of dimensions for each model, when only 5% of their initial size is preserved.

removing 95% of the original dimensions, while DR-1 could only keep 89.16% of the original mAP. We conclude that choosing the right dimensions to drop improves the robustness of the feature vectors to dimensionality perturbations.

Base network We proceed, then, to fix the base network, and apply the stressor to different datasets. The results observed in Figure 2.7 indicate that the number of classes in the target dataset also seems to play an important role on performance resiliency.

For correctly classifying the data, diverse datasets may need complementary feature points, which can be lost with dimensionality reduction. We observe that larger datasets like Food-101, which contains 101 classes, have stronger responses to the dimensionality perturbations, and seem to need more of the original dimensions in order to be correctly classified. On the other hand, MIT-67, with 67 categories, suffered less from the reduction, and PASCAL VOC 2007, with 20 classes, was the one which better resisted to this stressor.

All results We show, in Figure 2.8, all of the results discussed in this sub-chapter in a single figure, as well as results for VGG-M-128 — a reduced version of VGG-M, with 128 dimensions in its feature space.

The best resiliency was observed when using DR-2, that uses a PCA-based strategy. We also observed an intuitive property: as the feature space gets more compact from the beginning, the dimensionality reduction causes more damage

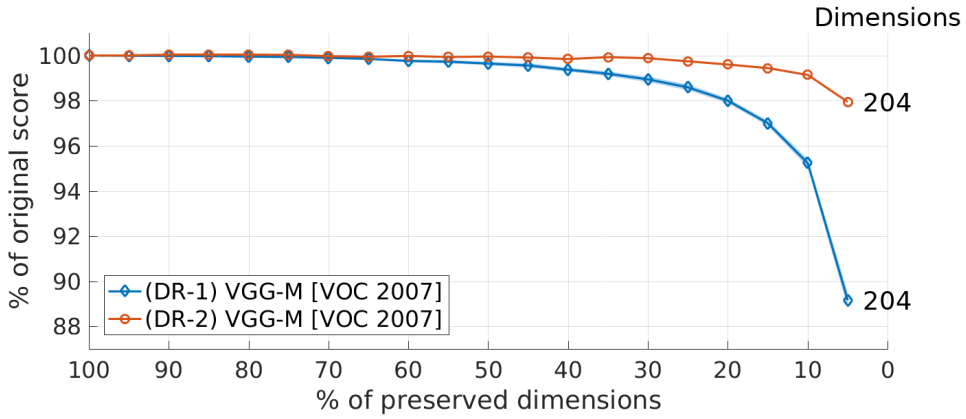


Figure 2.6 – Results for **different dimensionality reduction strategies** for PASCAL VOC 2007. The PCA-based strategy DR-2 enables the compression scheme to keep more information, yielding better results.

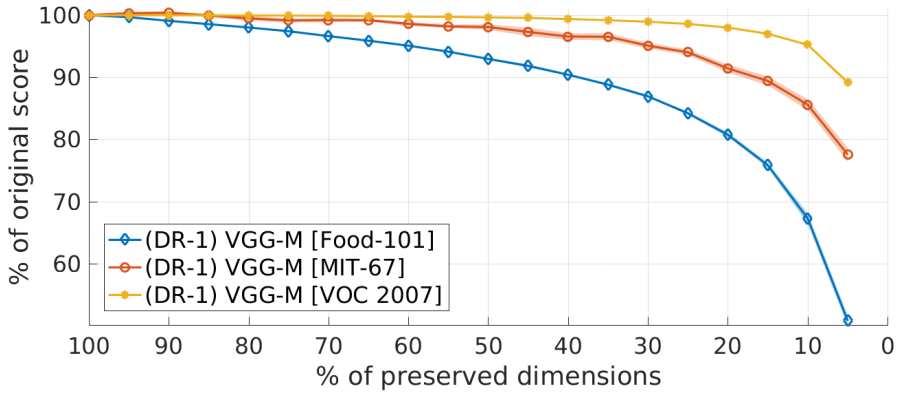


Figure 2.7 – Results for **dimensionality reduction (DR) with VGG-M** for Food-101, MIT-67 and PASCAL VOC 2007. The datasets have 101, 67 and 20 classes, respectively.

to the performance. This is evident when comparing VGG-M with VGG-M-128, the latter being trained with less dimensions on its original feature vector.

Layer choice In the last study for this stressor, we measure the impact of the layer choice for our base setup (VGG-M and PASCAL VOC). Traditionally, transfer learning is performed with the features from the penultimate layer of the network. For VGG-M, this represents Layer 19, as shown in Table 2.1. We then chose to test all layers from the 16th up, since lower layers have high dimensionality and are difficult to deal with in a setup that adopts an SVM as a classifier. The results for this experiment are shown in Figure 2.9.

We remark that the output of layer 21 is the most sensitive to the dimensionality perturbations. This is intuitive, since this feature vector corresponds to the output

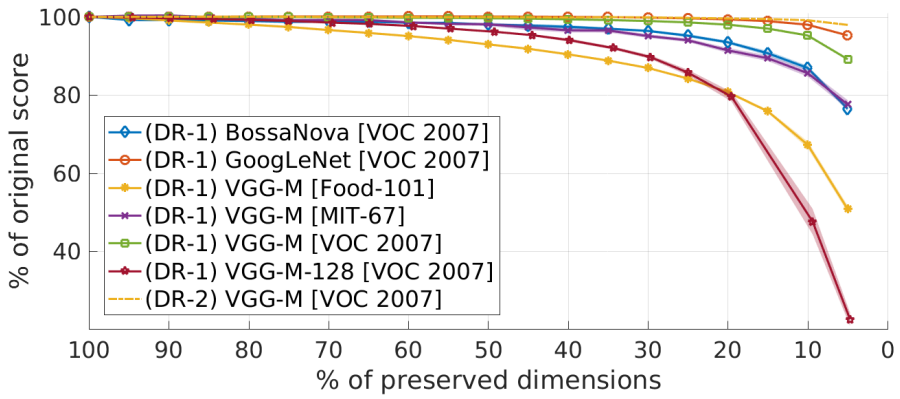


Figure 2.8 – Results for **all dimensionality reduction (DR) experiments** with GoogLeNet, BossaNova, and VGG-M for Food-101, MIT-67, and PASCAL VOC 2007.

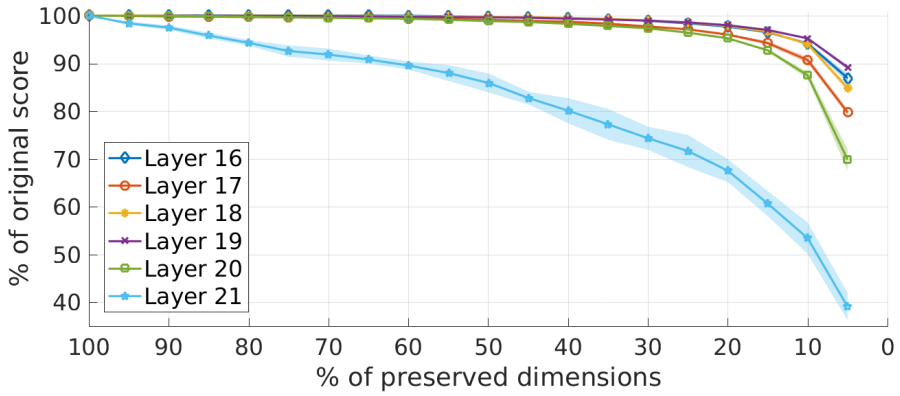


Figure 2.9 – Results for **dimensionality reduction (DR-1) with different layers of VGG-M**, described in Table 2.1, in the PASCAL VOC 2007 dataset.

of the final Softmax function, and thus to the probabilities for each class from ImageNet. Reducing the dimensionality of this specific output directly affects the classification performance, because we are removing high level information.

For all other layers, we observe similar degradation due to the dimensionality reduction. The standard layer for transfer learning (L19) is, however, slightly more robust than the others. This layer not only presents the best overall score for our experiment, also the best relative resiliency to the perturbations, corroborating the standard practice of taking it for transfer tasks.

2.4.2 Quantization (Q)

The second round of experiments we propose measure the importance of precision and the internal redundancy in the directions of the feature space. To

measure this, we reduce the size of the feature vectors, from initial $32 * m_i$ bits [3], by aggressively limiting their values. If we remove precision from the representations while observing small losses in the classification score, we can infer that the most important details to this task are still encoded by the remaining precision.

Base setup To begin, we apply the quantization stressor to the base setup (VGG-M and PASCAL VOC). The results for this test are presented in Figure 2.10, where we observe that most of the precision in the representations is useless to the classification task. With both Q-1 and Q-2 setups we were able to limit the number of values to 7 without losing performance, while the original size is 2^{32}. This means we can encode them in 3 bits (instead of 32) without changing the original score.

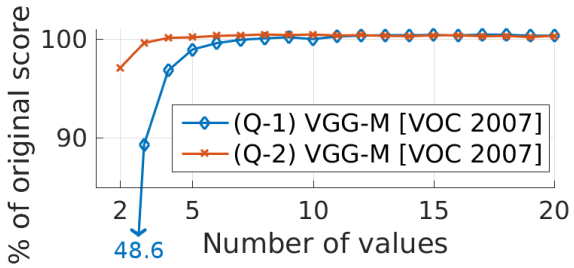


Figure 2.10 – Results for **quantization of features on the base setup (VGG-M and PASCAL VOC 2007)**. We can keep vanilla performance while reducing the feature vectors from $32 * m_i$ to $\lceil \log_2 7 \rceil * m_i$ and $\lceil \log_2 4 \rceil * m_i$ bits, using Q-1 and Q-2, respectively.

However, we also observe that Q-2 performed better than Q-1. While the latter (Q-1) defines a single set of values to all dimensions, the first (Q-2) tolerates different scales, defining a different set of values for each dimension. This result indicates that adaptiveness to scale plays an important role in compressing the representations. With Q-2 we can further reduce the representation to 4 values, which can be encoded in 2 bits.

Finally, Q-1 kept vanilla scores with 7 values, while Q-2 only needed 4. This represents a strong compression of the feature vectors, from $32 * m$ to $\lceil \log_2 4 \rceil * m = 2 * m$ bits.

Binary quantization Next, instead of simply studying the quantization strategy adopted in our experiments, we also propose to systematically erase rightmost bits from the representation, without actually using their values to guide this reduction. This is a more *rustic* quantization, which does not use the features to calculate the set of values to be used, and can check (1) the importance of a

3. For 32-bit single-precision floating-point numbers.

data-aware strategy, and (2) the effectiveness of a simple and straightforward approach.

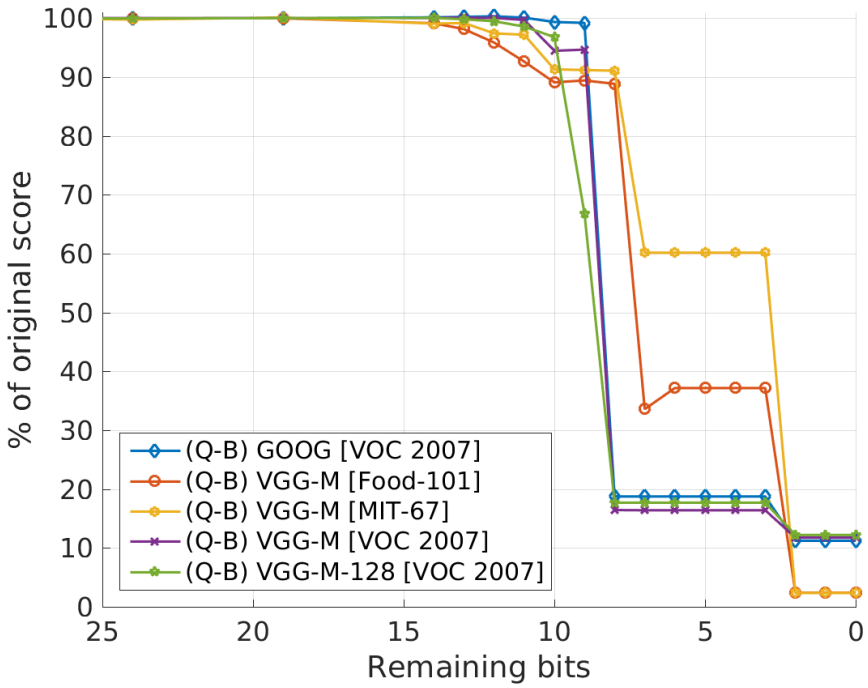


Figure 2.11 – Results for **erasing rightmost bits on multiple representations**. We can keep vanilla performance for all configurations when reducing the feature vectors from $32 * m_i$ to $14 * m_i$ bits. GOOG indicates the GoogLeNet network.

The results for the binary quantization are presented in Figure 2.11. For the base setup (VGG-M and PASCAL VOC), we keep original scores while erasing 63% of the binary representation of the features (from 32 bits to 12 bits).

Similar performance was observed throughout datasets. However, we remark that GoogLeNet was more robust to this perturbation, keeping the original scores with 11 bits, and losing a very small amount of it with 9 bits left.

We also remark that having a representation space of a different dimensionality does not seem to play an important role with this stressor. VGG-M-128 had results very similar to the ones of VGG-M with the same dataset. And GoogLeNet, having feature vectors that are 12-times bigger than the ones from VGG-M, and about 390-times bigger than the ones from VGG-M-128, also performed similarly.

Finally, we highlight that a data-drive approach, presented with Q-1 and Q-2, can compress these representations in a more efficient manner. The best scores with our binary compression could only reduce the representations from 32 bits

to 9, in the best scenario, without losing most of the classification performance. With only 2 bits, Q-2 was able to keep almost the same score.

Layer choice Following the same idea behind the dimensionality reduction experiments, we measure the impact of the layer choice for our base setup (VGG-M and PASCAL VOC) with the quantization (Q-2) strategy. The results for these experiments are shown in Figure 2.12.

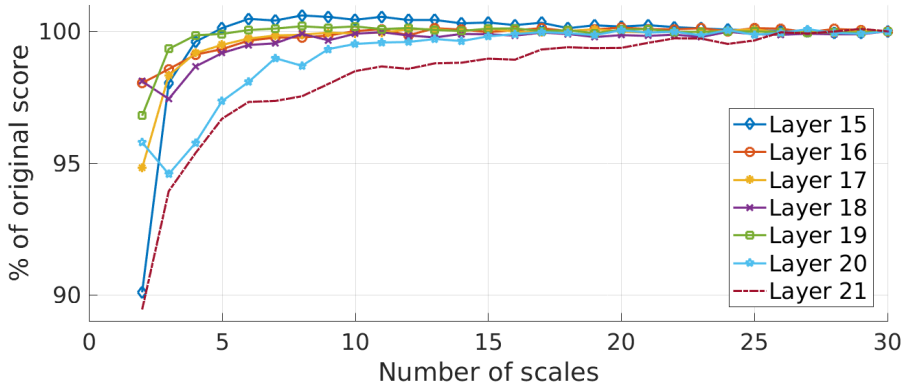


Figure 2.12 – Results for **quantization (Q-2) with different layers of VGG-M**, described in Table 2.1, in the PASCAL VOC 2007 dataset.

Similarly to our last analysis for the layer choice, we remark that the output of layer 21 is, again, the most sensitive to the dimensionality perturbations. It is important to highlight, however, that although we reduce the precision of the feature vectors, their dimensionality remains the same, therefore layers containing less dimensions can still keep more information.

With VGG-M, layers 20 and 21 are the smallest ones, which may help justify their poor performance. We also observe the impact of the Softmax function on this stressor, since layer 21 performed much worse than layer 20. The standard transfer layer (L19) remains fairly robust to the transformation we are applying. Although it does not present the best scores for all data points of the experiment, it remains among the top ones, leading on some compression levels. This result, allied to the one obtained for the dimensionality reduction, helps to validate the choice of the penultimate layer for transfer learning. With the exception of the *Layer choice* experiments, all the results we report in this chapter are obtained using L19.

2.4.3 Feature Compression (FC)

For the third and last round of experiments related to the feature space exploration, we assess the complementarity of the dimensionality reduction (DR) and

the quantization (Q) experiments. By applying both at the same time, we can measure if the features are redundant both externally (redundant dimensions) and internally (excessive precision).

On Table 2.3, we summarize the main results for the dimensionality reduction (DR) experiments on PASCAL VOC 2007, where each column indicates the maximum desired loss with respect to the original score for an experiment, while the cells indicate the minimum value which satisfies such requirement. For example, the second line of the second column (*GoogLeNet* and *DR-1 – 2%*) reveals that with only 10% of the dimensions preserved, GoogLeNet score drops less than 2%.

	Original Score	**DR-1 – 2%**	**DR-1 – 5%**	**DR-2 – 1%**
VGG-M	76.95%	25%	10%	10%
GoogLeNet	80.58%	10%	5%	–
BossaNova	39.59%	50%	25%	–

Table 2.3 – **Minimum representation rate for dimensionality reduction** (DR) on PASCAL VOC 2007. Each column indicates a requirement, and each line represents a dataset. The cells reveal the minimum representation needed for losing at most the indicated percentage. For instance, (DR-1 – 2%) + GoogLeNet = 10% means that with only 10% of the dimensions, we lose at most 2% of the initial score with GoogLeNet.

Then, we perform the same experience for the quantization (Q) stresses on our base setup, composed of a VGG-M network and the PASCAL VOC 2007 dataset. We show these results on Table 2.4, where we see that high compression rates can be achieved with quantization as well: With only 3 values, that can be represented with 2 bits, we lose at most 1% of the initial score.

	Original Score	**Q-1 – 1%**	**Q-1 – 4%**	**Q-2 – 1%**	**Q-2 – 3%**
VGG-M	76.95%	6 values	4 values	3 values	2 values

Table 2.4 – **Minimum representation rate for quantization** on our base setup. Each column indicates a requirement. The cells reveal the minimum representation needed for losing at most the indicated percentage. For instance, (Q-2 – 1%) = 3 values means that with only 3 values, that can be represented with 2 bits, we lose at most 1% of the initial score with VGG-M.

Finally, we apply our Feature Compression (FC) strategy, which combines dimensionality reduction and quantization, to the base setup, and the results for

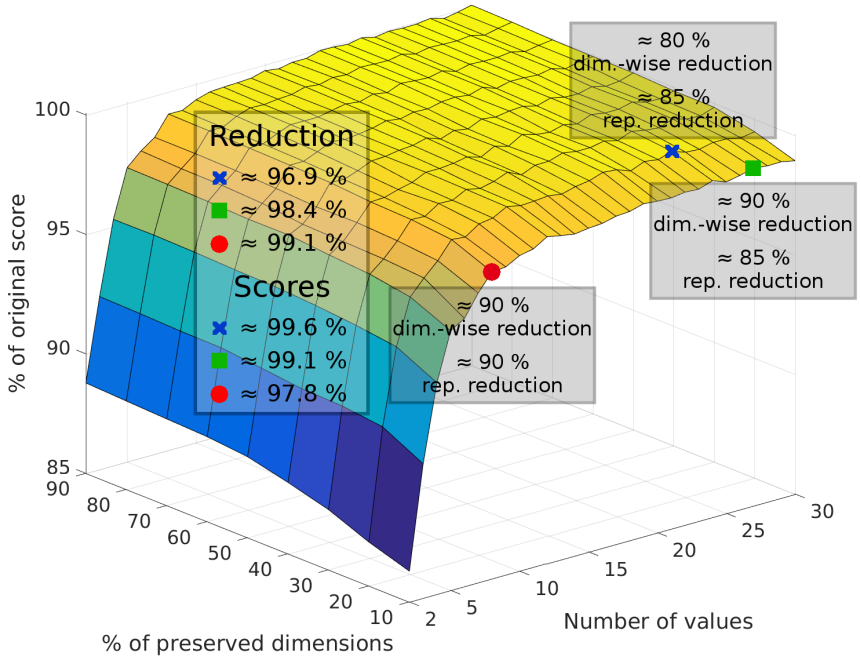


Figure 2.13 – **Results for feature compression (FC).** We reduce the number of dimensions and the precision of the feature vectors at the same time. The circle, square and cross, mark configurations with compression rates of 99.1%, 98.4% and 96.9%, respectively, while maintaining 97.8%, 99.1% and 99.6% of the original score.

this experiment are shown in Figure 2.13. The flat region on the top represents combinations of parameters from DR-2 and Q-2 with complementary characteristics, indicating that the features can be compressed in terms of dimension and precision at the same time. We point, with the circle, square and cross markers, specific combinations of DR-2 and Q-2 with compression rates of 99.1%, 98.4% and 96.9%, respectively, while maintaining 97.8%, 99.1% and 99.6% of the original score.

Summary In this section, we have seen that features extracted from deep CNNs are overrepresented for common transfer tasks. This over-representation is manifested both in terms of number of dimensions and numerical precision. We were able to greatly reduce their size with simple strategies—as simple as random goes—without falling behind on the classification score. This opened up the possibility for easy and powerful compression schemes, which were successfully

explored, demonstrating again that the properties we observed can be taken advantage of for creating practical applications.

2.5 Network Compression

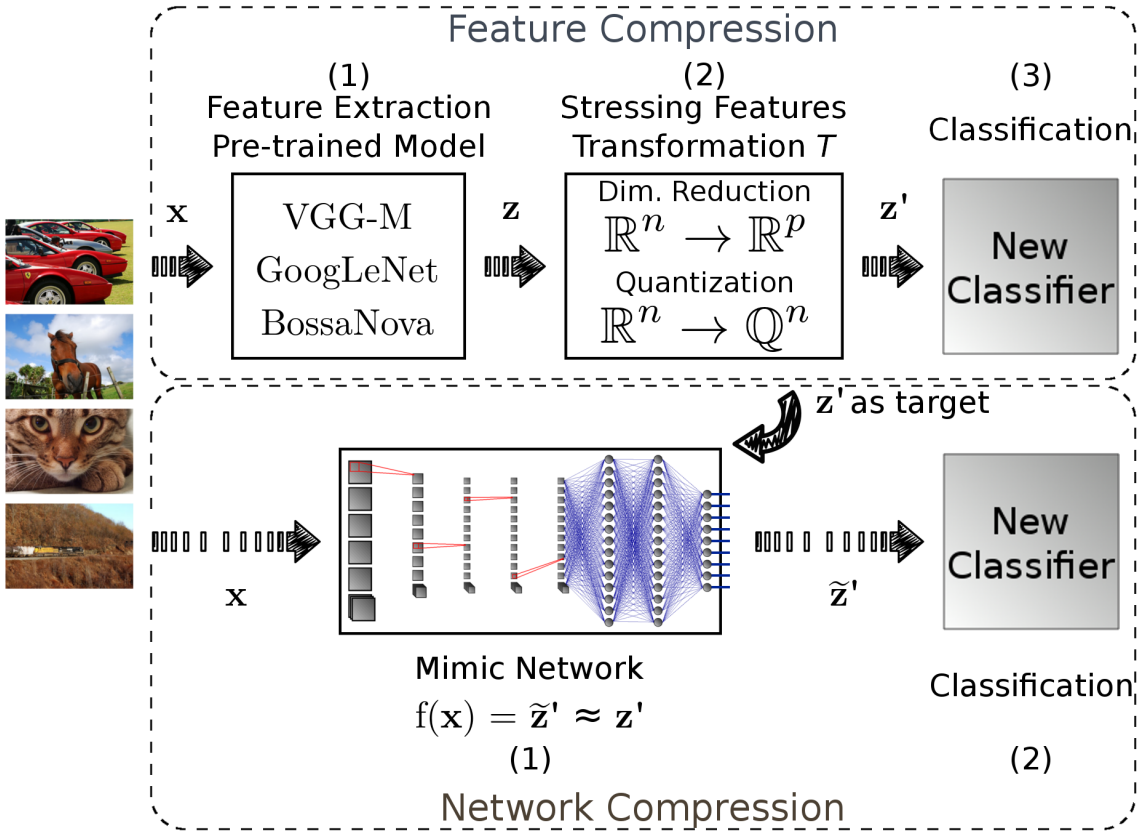


Figure 2.14 – **Representation of our mimic intuition.** A pre-trained network (top, 1) is used to extract image descriptors that are transformed using the stress framework (top, 2) from (Carvalho, Cord, et al. 2016), and a smaller network (down, 1) can be used to copy the stressed descriptors (top, 2).

Up to now, our strategy was able to compress the output of a deep network, however, it would be interesting to be able to compress the whole network. The results achieved in the last section opened many questions concerning the representations of deep networks, therefore we propose an exploratory study of network compression schemes, followed by a loss proposition that takes advantage of our previous results. In the following, we briefly describe three of the most adopted strategies for this end.

Binarization One idea to create compact networks is proposed by M. Kim et al. 2016, which consists in a neural network with binary parameters, bias terms, input and intermediate hidden layer output signals, requiring only bit logic for the feedforward pass. The experimental portfolio is, however, limited, and due to the nature of their proposal we assume their strategies are mainly applicable to easy tasks, like MNIST (Yann LeCun et al. n.d.).

Similarly, Matthieu Courbariaux et al. 2016 proposed a network with weights and activations constrained to +1 and -1. During inference time, most of the multiplications are replaced by 1-bit exclusive-not-or (XNOR) operations. Memory usage is drastically reduced during test, since we don't need the precision that was kept for the gradient during the training stage. This strategy eases the usage in more difficult tasks, since the learning has the full precision, necessary for small steps.

Changes in representation Different strategies change the representation of floats in the network. Judd et al. 2015, for example, propose replacing the floating points by fixed-point variables, changing also the precision of the numbers. Their approach can only be used for test time, meaning that the training must be conducted normally, and the results they achieve with many architectures (LeNet, Convnet, AlexNet, NiN and GoogLeNet) agree with ours, showing that deep networks have excessive precision.

With the same goal of reducing the network in inference time, Y.-D. Kim et al. 2015 propose one-shot whole network compression, which consists of three steps: (1) rank selection with variational Bayesian matrix factorization; (2) Tucker decomposition on kernel tensor; and (3) fine-tuning to recover accumulated loss of accuracy. They test their setup in mobile devices, and with a AlexNet network, they achieve a speedup of 7.61 on a smartphone Samsung Galaxy S6.

The idea of applying transformations to all layers in inference time of Han et al. 2015 is able to greatly reduce the size of the model without losing precision. Although they are able to compress the storage space occupied by the model, the memory requirements during runtime do not see the same benefit, since some values must be looked up before usage in order to reconstruct the weight matrix. They achieve a reduction of 35 for weight storage requirements for AlexNet and of 49 for VGG-16, besides presenting measures of speedup and energy efficiency for their proposals.

Mimic learning Another strategy for creating compressed networks is mimic learning. This technique consists in training a big network, and then training a student, usually a smaller network, to copy features from the big one. If correctly applied, in the end, a smaller network which produces features similar to the teacher's features is expected.

A primordial version of this idea is presented by Bromley et al. 1994. Their proposal is to use two siamese networks (i.e. identical networks, with shared weights), in order to approach the task of signature verification. Their intuition lies on the fact that by forcing the proximity, in the feature space, of two signatures of the same person, we are encouraging the model to build instance-level invariance, being able to recognize different signatures of the same person instead of only the exact same one.

Building an ensemble of models is another way of improving performance. An ensemble is composed of multiple instances that may not be computationally expensive individually, but their combined execution generally is. Modern architectures, like deep neural networks, have many parameters and dependencies on previous computation, and building an ensemble of these can be costly. These characteristics make it difficult to take advantage of the full potential of such systems in constrained environments, like mobile phones and smart devices, emerging from the Internet of Things' era.

Aiming to solve this problem, Buciluǎ et al. 2006 introduced a method for compressing ensembles into smaller, faster models, without considerable loss in performance. Their approach consists of synthetically augmenting the dataset by exploiting neighbor relations between samples, and using an ensemble to annotate the new data.

One of the most known recent techniques for mimic learning is model distillation, presented by Hinton et al. 2015. Their approach uses the probabilities of a teacher model to train a student network. Distillation's intuition is to optimize a weighted average of two different objective functions, the first one being the cross entropy of soft targets (outputs of Softmax), computed with high temperature, and the second one the cross entropy with the labels (classification task).

Other strategies have been built upon these propositions, improving and studying their properties. Because many of them are not sufficiently related to this thesis, we refer the reader to the work of Cheng et al. 2018 for a better (but not complete) coverage.

2.5.1 Relaxed mimic losses

Contrary to other approaches, mimic learning can take advantage of a teacher network that has better scores for a given task. Furthermore, it does not rely on hand-crafted strategies that need to be heavily tuned in order to be applied to a problem. The application of this kind of method to real-world problems is clear once we start considering the cost of deep CNNs, for example: it would not be feasible to have hundreds of Graphics Processing Units (GPUs) inside an automated vehicle in order to provide real-time recognition of its surroundings, nor having to wait for longer periods of time in order for a cellphone to correctly tag the contents of a picture.

In these contexts, a smaller network could be important, as common hardware would be capable to run it faster and it would be overall cheaper to deploy. When using mimic learning, one can arbitrarily choose the number of layers of the student network, being able to control the depth of the network and, consequentially, the number of steps that depend on previous calculations.

Therefore, in light of the results discussed in Section 2.4, we hypothesize that instead of directly copying the descriptors extracted by the network, it should be possible copy stressed descriptors by adopting one of our stressors as the transformation depicted in Figure 2.14. A standard mimic strategy would adopt the identity function instead. Should we succeed in this task, we will have a compression scheme capable of reducing the model and the size of the descriptors it generates, while maintaining good performance.

In particular, we are interested in testing if the quantization stressor could be used in a mimic scheme. Our observations show that fine precision is often not important for good performance on classification, therefore, when adopting a teacher-student learning scheme, it may be better for the student to accept small deviations from the teacher's output, instead of penalizing every small difference in their outputs.

Instead of simply applying the quantization stressor to the teacher network and mimicking its features, we devised a family of relaxed regression losses that are able to approach the scores of both models without forcing them to be the same. For the relaxed L1 loss, this is equivalent to two opposed Hinge functions, with a margin term, as depicted in Figure 2.15.

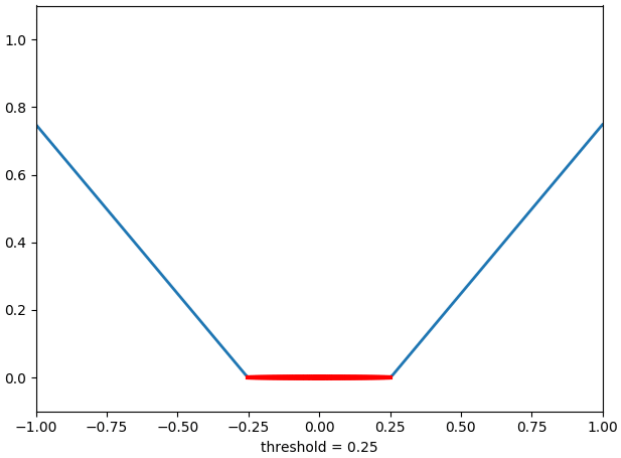


Figure 2.15 – **Our idea of a relaxed regression loss.** The relaxed L1 loss is equivalent to two opposed hinge functions spaced by a threshold. This threshold removes the penalization for feature vectors that are close to the original ones.

We provide the relaxed version for the L1 and L2 losses, shown in Equation 2.1, in Equation 2.2. This strategy could be extended to many other regression losses, since it consists in adding a margin centered at 0 in order to remove the sensibility of the gradient when small differences are detected.

$$\begin{aligned} \mathcal{L}_{L1} &= \sum_{i=1}^{N} |y_{i,teacher} - y_{i,student}| \\ \mathcal{L}_{L2} &= \sum_{i=1}^{N} |y_{i,teacher} - y_{i,student}|^2 \end{aligned} \tag{2.1}$$

$$\begin{aligned} \mathcal{L}_{RL1} &= \sum_{i=1}^{N} |y_{i,teacher} - y_{i,student}| - \alpha \\ \mathcal{L}_{RL2} &= \sum_{i=1}^{N} (|y_{i,teacher} - y_{i,student}| - \alpha)^2 \end{aligned} \tag{2.2}$$

2.5.2 Exploratory experiments

Teacher-student analysis In order to conduct our exploration, we propose a toy example with the setup described in Figure 2.14, but using simpler networks and datasets. We adopt the LeNet-5 network (LeCun et al. 1998), containing 145,000 parameters, and the CIFAR-10 dataset (Alex Krizhevsky 2009), containing 60,000 images.

Initially, an L2 regression loss is used, with the identity transformation instead of our stressors, forcing the student network to copy the probabilities generated by the teacher network. As for the student, we chose to use the same network, but training it from scratch and reducing the number of convolutional units in each layer, effectively reducing the number of parameters.

The results for this initial experiment, shown in Table 2.5, confirm that the student network can be much smaller than the teacher network, while being able to copy the teacher's performance in its almost totality. This result motivates further exploration of mimic learning.

Mimicking with relaxed losses We propose a simple experiment to verify the effectiveness of our strategy for mimic learning. This time, we adopt a VGG-16 network pre-trained on ImageNet as the teacher. Instead of creating smaller networks as students, we chose to adopt the same architecture, but randomly initialized, in order to measure the effects of the relaxation on the regression losses we test. In other words, we are following the scheme in Figure 2.14, but with an identity function for T.

#params for student	Relative score*
53.3k	97.8%
21.9k	97.7%
9.8k	95.6%
4.6k	73.0%

Table 2.5 – **Results for a simple mimic strategy**, using LeNet-5 and CIFAR-10, with an L2 regression loss. The teacher network has 145,00 parameters. * We present relative scores with respect to the teacher network.

Initially, we test the L1 and the L2 losses, described in Equation 2.1, where N indicate the number of dimensions of the feature vectors — the feature space sizes and number of parameters for the teacher and the student are the same, since we adopt the same network architecture for them.

Then, we test the RL1 and the RL2 losses, which are the relaxed versions we propose, described in Equation 2.2. To create them, we add a margin term to the original L1 and L2 losses, which will create a *safe margin* around the desired values, allowing small deviances to have zero loss.

For training our models, we adopted a simple setup: We train each student for 25 epochs on the ILSVRC 2012 training set, with $\alpha = 5 * 10^{-2}$ and the learning rate of 10^{-4} with the Adam optimizer (Kingma et al. 2015). Although these choices might not be not optimal for the standard task of ImageNet classification, we chose not to cross-validate them, since the goal is not to achieve top scores, but to demonstrate the effects of our losses in the mimic process.

In Table 2.6, we show the results for the relaxed loss experiment. Even with our limited protocol, and without cross-validating our hyper-parameters, we observe small but consistent gains with the relaxed version of the regression losses. We ran this experiment two times, and for both cost functions, our modifications were able to improve the results over the non-relaxed versions of the losses.

2.6 Conclusion

In Section 2.1 and Section 2.2, we presented approaches related to the usage of deep features in problems from the Computer Vision (CV) community. In modern applications, deep models have been used as a black box, without a clear understanding of its inner workings nor the nature of the representations they create. Allied to transfer learning, the power of deep learning was harvested to achieve state-of-the-art performance in many tasks, but few studies were dedicated to understanding their properties.

Loss	Score	Relative score*
L1	55.09%	100.00%
RL1	55.18%	100.16%
L2	55.61%	100.94%
RL2	55.68%	101.07%

Table 2.6 – **Results for a simple, large-scale mimic strategy**, using VGG-16 and ImageNet, with different losses. All experiments ran for 15 epochs with a fixed set of hyper-parameters that were not cross-validated. We present absolute classification scores on the ILSVRC 2012, as well as *relative scores with respect to the L1 loss mimic.

Our goal was to evaluate the robustness of deep representations by introducing perturbations to feature vectors extracted from upper layers of deep networks. We explored in depth the resiliency of features transferred from the VGG-M model to the Pascal VOC 2007 dataset. For this objective, we introduced, in Section 2.3, a stress framework capable of reducing the dimensionality and quantizing the vector space used during the transfer process.

The results shown in Section 2.4 revealed a high level of redundancy in deep representations, indicating they may be heavily compressed. In our experiments, we achieve a compression rate of 98.4%, while losing only 0.88% of the original score for Pascal VOC 2007. To ensure our conclusions are not dataset- nor model-specific, our two main approaches – dimensionality reduction and quantization – were extensively tested, with supplementary results for MIT-67, Food-101, GoogLeNet and BossaNova.

We observed that despite being more compact, deep architectures are also more robust to perturbations, when compared to approaches based on Bag-of-Words (BoW). Those findings are specially useful for image retrieval and metric learning (Le Barz et al. 2015), in which the size of the feature vector is crucial to achieve fast response times, and for applications involving portable devices or remote classification, in which data must be efficiently transferred over the network.

In Section 2.5 we have explored strategies for transferring the knowledge from a teacher network to a student network through mimic learning. We have shown in our initial toy experiments that smaller networks have the ability to distill the knowledge of a bigger teacher model without losing much of its performance. This results corroborates our observations in Section 2.4. In another experiment, we have tested an intuitive idea for relaxing regression losses in order to perform mimic learning, achieving promising results that can be further explored in future work.

Finally, the experiments we have shown in this chapter can easily see their way to many real-world applications. In the context of remote classification, for example, the compression of features is fundamental. Modern smartphones have enough computational power to run deep networks in inference mode without struggle, but implementing such a system can quickly become infeasible in terms of data transfer costs. This is especially true for remote context-based information retrieval, for which the embedding of one image has to be compared to a huge number of samples in the retrieval pool. For this particular situation, having compact descriptors not only reduce the cost of transferring them through the network, but also greatly reduce the search time.

CHAPTER 3

LEARNING MULTIMODAL LATENT SPACES

Contents

Chapter abstract

In this chapter, we tackle the cross-modal alignment problem for visual and textual data. Our target application is the large-scale retrieval task (image to text, and vice versa). Our proposed approach, AdaMine, brings 2 main contributions: (1) an auxiliary regularization loss capable of semantically organizing latent spaces via multitask learning; and (2) an adaptive triplet sampling strategy that provides consistent gradient information during training. The effectiveness of our method is demonstrated on the Recipe1M dataset, composed of one million image-recipe pairs and additional class information. We improve the state-of-the-art results on this dataset and empirically demonstrate the benefits of implicit multitask learning for the retrieval task.

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

— Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord (2018). "Cross-Modal Retrieval in the Cooking Context: Learning Semantic Text-Image Embeddings". In: *The ACM conference on Research and Development in Information Retrieval (SIGIR)*

3.1 Introduction

We have discussed, in Chapter 1, the image retrieval task. Its goal is to retrieve an image based on a specific criterion and a query (usually another image). An easy way to achieve this is to suppose the existence of a latent space in which a specific measure can be used to assess the distance between any two points. Each of these points, inside this latent space, can be the output of a CNN for a specific image, for example.

These spaces usually have interesting properties of redundancy and excessive precision (see Chapter 2). Therefore, the objective of this chapter is to present a way of refining deep models in order to enrich their feature space and obtain better descriptors.

Considering the image retrieval task, and the fact that we can use a CNN to obtain feature vectors from its feature space, the question to be asked is: *how to correctly organize it?* And the answer is not as simple as it sounds: changing the weights of the CNN in a way that it can perform the transformation from the image space to this latent one. In order to do so, a fine-tuning approach, discussed in Section 2.2, can be adopted.

However, the problem becomes more complex when we start considering a scenario with multi-modal data. Instead of dealing with images only, our target application contain images and texts that should be correctly aligned in this space. For this situation, it is no longer possible to train a single CNN that can learn the image-text correspondence, since this model will only be able to deal with one mode. A straightforward way of taking up this issue is to adopt alignment learning approaches, which introduce a loss into this space responsible for matching any two vectors.

One of the main candidates for this is metric learning, whose goal is to find a transformation that can map an input into a point inside a latent space. Inside this space, a pre-chosen metric (e.g. euclidean) can be used to compare any two points. The classical cost functions used for this purpose include the pairwise and the triplet, in which a pair and a trio of examples are required for each training iteration, respectively.

After projecting different modalities into a representation space, we directly align them using metric losses. Contrary to the standard multi-task approach of adding multiple heads, each specialized to a different objective, this strategy does not add cumbersome extra parts to the model, that might be heavy in number of parameters and are usually discarded after being trained.

The benefits of these approaches become more evident once we start considering the difficulty of data annotation since, in a metric learning scheme, one can take advantage of inequalities between images. In practice this means we only need to indicate if two images should be similar or not, or that one given pair of images

should be closer in the representation space than another one, instead of actually labeling the samples.

In this chapter, we will adopt a fine-tuning setup and explore metric learning approaches for the cross-modal alignment. This will allow us to refine not only the information circulating inside the chosen architectures, but also to align their feature vectors in a shared latent space. The rest of this chapter is organized as follows:

— In Section 3.2 we discuss the alignment learning issue, as well as metric learning approaches. There are two main groups of studies we are interested in: the first one adopts metric losses but focuses on improvements in the network architecture, and the second one proposes new cost functions to address shortcomings commonly found on the popular ones. In order to provide a broader understanding of the roles of both groups in this research, we will briefly describe one work from the literature for each topic, and focus on the two losses that are the most related to our contribution: the pairwise and the triplet;

— Then, in Section 3.3 we introduce a new constrained learning scheme to align image and text in a semantic space for applications to cross-modal retrieval. Our method, called AdaMine, encompasses the following contributions: A double loss composed of instance-based and semantic-based triplets, as well an adaptive sampling strategy that allows both losses to be correctly weighted and to work together in reorganizing the feature space;

— AdaMine is then tested in Section 3.4, achieving a 5-fold improvement over the state-of-the-art of the adopted large-scale cross-modal retrieval task. We perform an extensive set of analyses to show the role of each part of our contribution, revealing its flexibility, and showing it does not require in-depth tuning of hyper-parameters in order to obtain good results;

— Finally, in Section 3.5 we draw the conclusions of this chapter, as well as the perspectives to future work related to AdaMine.

3.2 Related work

Building cross-modal embeddings between text and images is a challenging task in Computer Vision. The resulting semantic representations are essential for various applications, as visual question answering (Antol et al. 2015), image-caption retrieval (Kiros, Zhu, et al. 2015), and image-caption generation (Xu et al. 2015; Karpathy et al. 2015). Some of these tasks require an optimal alignment between matching pairs of items in order to obtain relevant ranking and retrieval, others do not align the modalities, but adopt an in-between task.

To tackle this problem, learning-based approaches have been developed using cross correlation alignment, e.g. CCA-based (Hotelling 1936; Tran et al. 2016)

and its deep extensions (Andrew et al. 2013; Yan et al. 2015), or metric learning strategies based on pair or triplet constraints to learn cross-modal embeddings (Kiros, Salakhutdinov, et al. 2015).

As an example of ad-hoc approach, Joulin et al. 2015 experimented with 99.2 million Flickr photos with associated titles, hashtags, and captions. They trained GoogLeNet's and AlexNet's variants, adopting a different strategy to speed up the training phase: only the weights that correspond to classes present in a given training batch are updated when it's presented to the network. Their model was able to predict word similarity and some correspondence between different languages (FR-EN) only by learning from image similarity. However, they adopted a simple one-versus-all logistic loss. Their claim is that a pairwise ranking loss promotes sparse updates, which significantly slows down training.

Being able to deal with multi-modal data is important for many modern applications. In particular, for automated vehicles that need to understand their environment based on information coming from different sources and in different formats, such as Lidar sensors and RGB images. For example, works like the one of Audebert et al. 2018 investigate fusion strategies for using data from different modalities in order to perform semantic labeling.

For many tasks, semantic information is available and taken advantage of. El Mahdaouy et al. 2018, for example, use textual semantic information in order to perform the link prediction task in complex embeddings, and Salvador et al. 2017 exploit semantic information with the help of a classification head attached to the model.

Their application involves the tasks of ranking and retrieval and, in the literature, several surrogates for the inequality constraints presented below (Equation 3.1) have been proposed, like (Weinberger et al. 2009; Xing et al. 2003; Law et al. 2013), from which we highlight metric learning approaches, discussed in the this section.

Notations Let $x_v \in \mathcal{V}$ the visual input from space $\mathcal{V}$ and $x_t \in \mathcal{T}$ be the text input from space $\mathcal{T}$. Deep mappings $g_v : \mathcal{V} \to \mathcal{F}$ and $g_t : \mathcal{T} \to \mathcal{F}$ provide alignment between $\mathcal{V}$ and $\mathcal{T}$, that are compared using the distance function $d(x_v, x_t) = ||g_v(x_v) - g_t(x_t)||_2$ [1]. Assuming $\mathbb{Q} = \{x_q\}$ to be the set of queries, $\mathbb{P}_q = \{x_p\}$ the set of relevant (positive) samples with respect to query x_q, and $\mathbb{N}_q = \{x_n\}$ the set of irrelevant (negative) samples with respect to query x_q, the ranking constraints for all queries are

$$\forall x_q, \forall x_p, \forall x_n, \quad d(x_q, x_p) < d(x_q, x_n) \tag{3.1}$$

1. For simplicity, we will consider identical notations for distances on $\mathcal{T} \times \mathcal{V} \to \mathbb{R}^+$ and $\mathcal{V} \times \mathcal{T} \to \mathbb{R}^+$.

3.2.1 Pairwise alignment

Xing et al. 2003 uses an objective function minimizing all the distance between positive pairs while maintaining the sum over all negative pairs large enough. Initially proposed for linear embedding, this approach has been improved using margin scheme and deep nets embedding (Hadsell et al. 2006). The resulting pairwise loss function $\mathcal{L}_{pw}$ for θ including all the parameters of g_v and g_t is presented in Equation 3.2, with $y = 1$ (resp. $y = 0$) for pos. (resp. neg.) pairs, the hinge loss $[\psi]_+ = \max(0, \psi)$, and α the margin. The resulting cost is linearly proportional to the computed distance between two items, conditioned to α.

$$\mathcal{L}_{pw}(\theta, x_q, x) = y \cdot d(x_q, x) + (1 - y)\big[\alpha - d(x_q, x)\big]_+ \tag{3.2}$$

For large scale visual and textual embedding, Salvador et al. 2017 adopts the pairwise formulation. They also introduce a dataset, to which they add semantic information corresponding to classes. This additional information is used to learn a classifier, which is exploited to regularize the learning procedure of their cross-modal embedding. However, noticing that brute pairwise optimization yield poor results, they combine it with a classification strategy for which they have additional supervised data. The latter is responsible for regularizing their embedding framework.

A different optimization strategy for cross-modal embedding is provided by the Canonical Correlation Analysis (CCA) (Hotelling 1936) which aims at maximizing the correlation in $\mathcal{F}$ between positive pairs only. CCA and its variations like Kernel-CCA (Lai et al. 2000; Bach et al. 2002) and Deep-CCA (Andrew et al. 2013) have been successfully applied to align text and images (Yan et al. 2015).

We adopt a pairwise cost approach as baseline in Section 3.3, and we propose a more efficient learning strategy built on triplet-based approaches, described in the following.

3.2.2 Triplet-based learning and extensions

A more natural surrogate of the ranking constraints in Equation 3.1 is to consider triplets composed of a query, a relevant, and an irrelevant sample. This strategy is similar to the Large Margin Nearest Neighbor loss (Weinberger et al. 2009), but without explicitly penalizing large distances between positive samples, and has been successfully used to learn Visual Semantic Embeddings (VSE) (Kiros, Salakhutdinov, et al. 2015) with applications to captioning as a text retrieval task (Karpathy et al. 2015).

Contrarily to pairwise, the triplet loss (Equation 3.3) defines a margin between two opposing samples, forcing positive and negative pairs to cooperate. In other

words, no explicit constraint is imposed on the scale of the manifolds being created, but only on their interacting boundaries.

$$\mathcal{L}_{tri}(\theta, x_q, x_p, x_n) = \left[d(x_q, x_p) + \alpha - d(x_q, x_n)\right]_+ \tag{3.3}$$

One example of use case for the triplet loss is the work of Arandjelović et al. 2016. They initially propose an adaptation of the Vector of Locally Aggregated Descriptors (VLAD, Jégou et al. 2010), described in Equation 3.4, as a derivable layer in a neural network. For each cluster center c_k, VLAD will generate a vector aggregating the differences between c_k and all feature vectors x_i belonging to the Voronoi cell of c_k. This last condition is met by the hard assignment in Equation 3.5.

$$V(j,k) = \sum_{i=1}^{N} \sigma(\mathbf{x}_i, k)(x_i(j) - c_k(j)) \tag{3.4}$$

$$\sigma(\mathbf{x}_i, k) = \mathbb{1}_{k=\arg\min_l \|\mathbf{x}_i - \mathbf{c}_l\|_2^2} \tag{3.5}$$

For neural networks, the main problem with this approach is the non derivability of the hard assignment operation. The solution Arandjelović et al. 2016 found is to replace the hard assignment by a Softmax function, expanding and manipulating the distance to obtain the first part of Equation 3.6, where $\mathbf{w}_k = 2\alpha\mathbf{c}_k$ and $b_k = -\alpha \|\mathbf{c}_k\|^2$.

$$N(j,k) = \sum_{i=1}^{N} \frac{e^{\mathbf{w}_k^T \mathbf{x}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^T \mathbf{x}_i + b_{k'}}} (x_i(j) - c_k(j)) \tag{3.6}$$

Contrary to VLAD, NetVLAD decouples $\mathbf{w}_k$, b_k and $\mathbf{c}_k$ during training, treating $\mathbf{w}_k^T \mathbf{x}_i + b_k$ as a fully connected activation, and learning these parameters independently. To test this layer, Arandjelović et al. 2016 chose the task of place retrieval, with pictures extracted from Google Street View Time Machine. An important characteristic of their data must be considered, however: images are annotated with their approximate location on the map, but no information about the camera position is provided. This means that two pictures having the same geographic position may be facing different directions, capturing different buildings and different scenes. A consequence of this lack of information is that one can only have *potential* positive pairs (i.e. being in the same position, they may be depicting the same object, but maybe not), and *definite* negative pairs (i.e. the images come from distant locations, definitely not the same place).

To overcome this issue, they adapted a triplet loss to consider the weakly supervised scenario. In their version, shown in Equation 3.7, for an image query q, only the best matching potential positive pair is used, assuming that for each image at least one true positive pair is available. This is equivalent to applying

a *hard-negative-margin-like* strategy for the positive pairs, by only selecting the easiest positive sample each time. All the negative pairs n_j are then forced to be away by a margin m, and the hinge loss h is applied to ensure only the pairs violating this constraint are considered during the learning phase.

$$\mathcal{L}_{\text{netvlad}} = \sum_j [\min_i D^2_{q,p_i} + m - D^2_{q,n_j}]_+ \tag{3.7}$$

Another way of extending pairwise and triplet losses is to develop quadruplets, which may offer more flexibility for modeling relative constraints and hierarchies, as shown by (Law et al. 2013). For example, quadruplet-like approaches with independent pairwise distances are adopted by (W. Chen et al. 2017; Ustinova et al. 2016), while (Huang et al. 2017) construct them with symmetric triplet-like losses. Often, quadruplets are combined with either a pairwise or a triplet loss, for regularization or semi-supervision (W. Chen et al. 2017; Huang et al. 2017).

Other proposals, like (Hoffer et al. 2015; L. Wang et al. 2016), tackle alignment or multimodal tasks, in manners that can be seen as extensions of Siamese networks (Bromley et al. 1994), where the same parameters are used for forwarding corresponding inputs.

Many strategies can be adopted for improving the performance of triplet losses. N-pair (Sohn 2016)'s idea, for example, is to reduce the burden of forwarding triplets through the network and increase convergence with an in-batch negative sampling strategy.

Song et al. 2015, however, modify the triplet loss, proposing the use of many negative examples for a positive pair of images. They also crawled a dataset (Online Products) from eBay, which is public and maintained by them. Their method was tested on Online Products, CUB-200-2011 and CARS196, achieving state-of-the-art performance on all three datasets. Their technique enables one to use more information during training, and possibly speeds up the training, since many distances are used for one training step.

On the work of Rippel et al. 2015, a new *magnet loss* is proposed. The idea behind it is to act on whole clusters instead of pairs or triplets; this approach removes the constraint of having a single cluster per class, allowing the same class to be present in different parts of the representation space – a desirable property, since images from a specific class may be strongly related to images from another class (e.g. on the cat class, a cat with a human, and on the dog class, a dog with a human; these examples are semantically related, and should be close in the representation space). Formally, their stochastic loss is defined by

$$\mathcal{L}_{\text{magnet}} = \frac{1}{MD} \sum_{m=1}^{M} \sum_{d=1}^{D} \left\{ -\log \frac{e^{-\frac{1}{2\hat{\sigma}^2} \left\| \mathbf{r}_d^m - \hat{\boldsymbol{\mu}}_m \right\|_2^2 - \alpha}}{\sum_{\hat{\boldsymbol{\mu}} : C(\hat{\boldsymbol{\mu}}) \neq C(\mathbf{r}_d^m)} e^{-\frac{1}{2\hat{\sigma}^2} \left\| \mathbf{r}_d^m - \hat{\boldsymbol{\mu}} \right\|_2^2}} \right\}_+$$

where $\mathbf{r}_d$ represents the deep representation of the input, $\hat{\boldsymbol{\mu}}_m$ the cluster's means, $\hat{\sigma}$ its variance, and $C(\cdot)$ the function that returns the class of its input.

3.2.3 Discussion about metric strategies

Song et al. 2016 explain that both pairwise and triplet losses have shortcomings that are not unusual. They argue that in specific situations, as the ones shown in Figure 3.1, the loss may desorganize the feature space, making it less concise.

In particular, for the pairwise approach, a pair with negative relation may be pushed outside of its cluster, in attempt to minimize the loss (see Figure 3.1a). As for the triplet one, depending on the position of the positive and negative pairs, the direction of the update may also push the anchor towards a cluster of a different class (see Figure 3.1b).

This problem is mainly due to the fact that each positive pair is bound to an arbitrary negative pair that may not be adequate for learning. With this in mind, they propose an extension of the triplet loss capable of picking negative pairs close to the positive one, avoiding situations in which a single badly-chosen example could disturb the representation space.

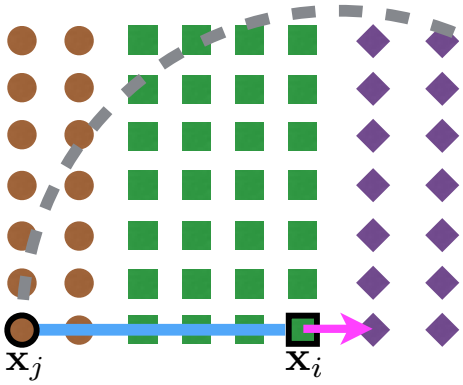
Contrary to the classical triplet loss (Equation 3.3), their approach (Equation 3.8) uses the entire training batch to mine negative examples. For each positive pair in the batch $(i,j) \in \mathcal{P}$, it takes advantage of all related negative pairs $(i,k),(j,l) \in \mathcal{P}$, penalizing the ones violating the margin α proportionally to their distance to it.

$$\begin{aligned} \hat{\mathcal{L}}_{\text{LSFE}}(i,j) &= log\Big(\sum_{(i,k)\in\mathcal{N}} exp\{\alpha - D^2_{i,k}\} + \sum_{(j,l)\in\mathcal{N}} exp\{\alpha - D^2_{j,l}\} \Big) + D^2_{i,j} \\ \mathcal{L}_{\text{LSFE}} &= \frac{1}{2\mathcal{P}} \sum_{(i,j)\in\mathcal{P}} max(0, \hat{\mathcal{L}}_{\text{LSFE}}(i,j))^2 \end{aligned} \tag{3.8}$$

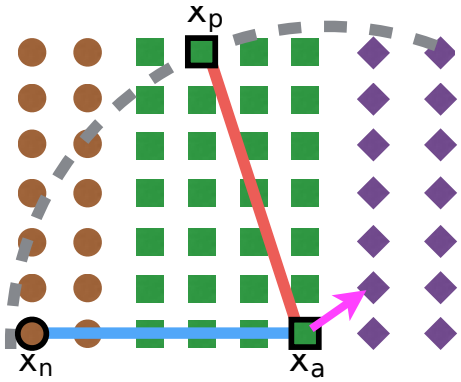
Although it is clear that such an approach can reduce these problems, it quickly becomes difficult to optimize, as it might take a high number of negative samples into account, and even if a batch-wise application can be envisaged to avoid calculating too many negative pairs, it will still increase the cost of back-propagating the error through the network. Our proposal, presented in the following section, includes an adaptive selection of triplets that allow us to also adapt the gradient during the training procedure, ignoring uninformative samples.

3.2.4 Multi-task approaches

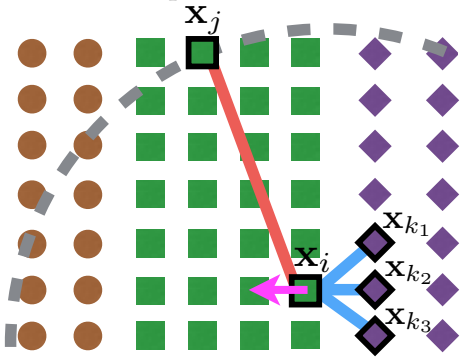
Recently, several deep learning schemes have been proposed to handle multi-task objectives. L. Wang et al. 2016 adopt triplet-based instance and structure-preserving losses, jointly optimizing retrieval and local neighborhood in the



(a) Pairwise corner case



(b) Triplet corner case



(c) LiftedSFE proposal

Figure 3.1 – **Shortcomings of pairwise (Figure 3.1a) and triplet (Figure 3.1b) losses.** For the pairwise approach, a pair with negative relation may be pushed outside of its cluster in attempt to minimize the loss (see Figure 3.1a). As for the triplet one, depending on the position of the positive and negative pairs, the direction of the update may also push the anchor towards a cluster of a different class (see Figure 3.1b). The Lifted SFE approach uses the closest negative pairs to avoid these problems (see Figure 3.1c). In this figure, blue arrows represent negative pairs, red arrows positive pairs and pink arrows the direction of movement of examples belonging to three classes: circle, square and diamond. (Adapted from Song et al. 2016)

feature space, while Kokkinos 2017 propose a multi-dataset, multi-task architecture, with a shared trunk and multiple branches that are specific to each task. The main idea is that related tasks can benefit from each other.

The approach of Salvador et al. 2017 to learn cross-modal embeddings is to add a semantic regularization objective, forcing the embeddings to solve a multi-task problem composed of the pairwise retrieval loss $\mathcal{L}_{pw}$ and a classification loss based on class information associated with each sample. This is accomplished by adding a new head to the model, specialized on the classification task, that tries to predict the class information from the joint space $\mathcal{F}$, similarly to Kokkinos 2017's proposal. The intuition behind the method is that the added objective will enforce some semantic structure into the joint embedding, which should improve generalization capabilities and lead to better retrieval performances.

However, one clear advantage of metric learning strategies, not explored by these works, is their independence. Losses yielded from them usually do not need specific parts to be added to the architecture in order to be used, and therefore integrating multi-task information into the model is as straightforward as designing a new loss for this task, without having parts that are task-specific.

3.3 AdaMine

We argue, then, that such *explicit* multitask strategies, as the one adopted by Salvador et al. 2017, may not be the best solution to improve the retrieval task. First, the extra head is cumbersome as it carries a possibly big number of parameters that will be discarded at the end of the training procedure as the classification is not the main goal. Moreover, we suspect the classification head to be able to perform its task without strongly modifying the joint space $\mathcal{F}$ due to the sheer complexity carried by its number of parameters. To avoid these drawbacks, we propose an implicit multitask objective which ensures a semantic structure on the joint space $\mathcal{F}$, as presented in the next section.

In our proposal, we adopt the triplet-based strategy as the main option for our cross-modal embedding learning. Similarly to (Sohn 2016), we carefully study how to build mini-batches composed of relevant matching pairs. Furthermore, we consider multitask learning as a way of exploiting additional training data about these pairs, as well as an adaptive mining strategy further detailed in Subsection 3.3.3.

Our global architecture is depicted in Figure 3.2. Based on two deep branches for textual and visual encoding, we introduce our learning scheme in the latent space. Detailed in Figure 3.3, we integrate both retrieval objective and semantic information in a single crossmodal metric learning problem, minimizing:

$$\mathcal{L}_{total}(\theta) = \mathcal{L}_{ins}(\theta) + \lambda \mathcal{L}_{sem}(\theta) \tag{3.9}$$

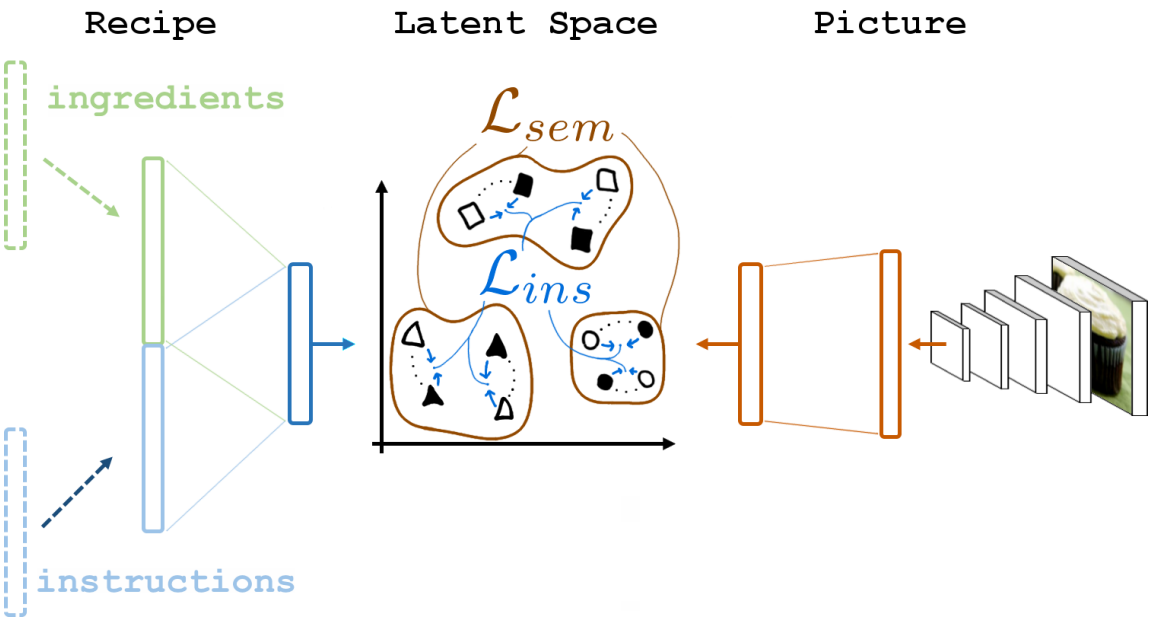


Figure 3.2 – **AdaMine overview.** Our model is based on deep textual (left) and visual (right) embedding branches that are mapped into a common latent representation space. A double-loss is used to train the whole architecture, mixing instance-level and semantic relations into a new regularization-based learning strategy. The main application relates to the large-scale Recipe1M crossmodal retrieval task, for finding pictures from recipes, and vice versa.

where $\mathcal{L}_{ins}$ is the loss associated with the retrieval task, and $\mathcal{L}_{sem}$ is the loss coming with the semantic information. In our framework, it is expressed as an implicit classification task. One can see this second term $\mathcal{L}_{sem}$ acting as a regularization over the $\mathcal{L}_{ins}$ optimization.

In the following, we first develop our choice for the retrieval loss. Next, we explain our proposition for the semantic loss $\mathcal{L}_{sem}$. Then, we explain our algorithm to manage efficient stochastic gradient descent over this double-triplet loss optimization.

3.3.1 Retrieval loss

$\mathcal{L}_{ins}(\theta)$ is defined using a surrogate of the ranking constraints. We propose two losses for the retrieval $\mathcal{L}_{ins}$. First, a pairwise baseline $\mathcal{L}_{pw++}$, for which we add a

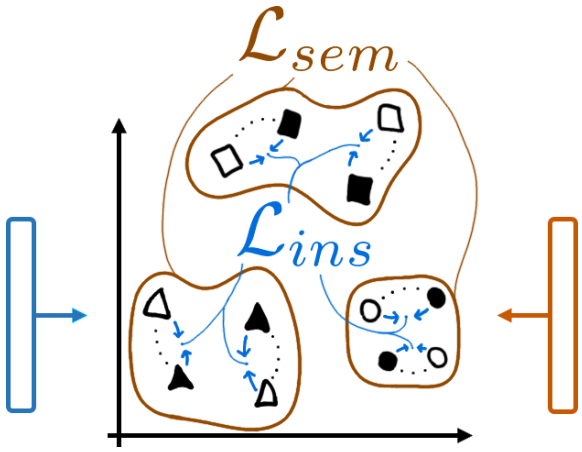


Figure 3.3 – **Close-up representation of the latent space** $\mathcal{F}$ in which text (coming from left) and images (coming from right) are embedded. Two cost functions work together to align them: $\mathcal{L}_{ins}$ is instance-aware and aligns corresponding samples, and $\mathcal{L}_{sem}$ is semantic-aware and provide class information for clustering.

positive margin to the pairwise loss adopted in (Salvador et al. 2017), as proposed by (Hu et al. 2014):

$$\begin{aligned}\mathcal{L}_{pw++}(\theta, x_q, x) = \; & y\left[d(x_q, x) - \alpha_{pos}\right]_+ \\ & + (1-y)\left[\alpha_{neg} - d(x_q, x)\right]_+\end{aligned} \tag{3.10}$$

with $y = 1$ (resp. $y = 0$) for pos. (resp. neg.) pairs. The positive margin α_{pos} allows matching pairs to have different representations, thus reducing the risk of overfitting. For carefully chosen α_{pos} and α_{neg}, we can still guarantee that a zero loss implies a perfect ranking. However, because we tackle the crossmodal retrieval problem, and the triplet loss is a more natural surrogate for the ranking function, when compared to the pairwise one, our second proposition is a bi-directional instance-based triplet loss $\mathcal{L}_{tri}$:

$$\mathcal{L}_{bitri}(\theta) = \mathcal{L}_{tri,\text{image,text,text}}(\theta, x_q, x_p, x_n) + \mathcal{L}_{tri,\text{text,image,image}}(\theta, x_q, x_p, x_n) \tag{3.11}$$

This approach is instance-based because relations between the anchor x_q and its triplet counterparts x_p and x_n are defined by whether they come from the exact same pair {image,recipe} or not, and bi-directional because both modalities are considered for $\mathbb{Q}$ [2].

2. Since the problem is completely symmetrical in modalities, we simply consider all triplets of the form $(\mathbb{Q}, \mathbb{P}_q, \mathbb{N}_q) \in (\mathcal{V}, \mathcal{T}, \mathcal{T})$ and $(\mathbb{Q}, \mathbb{P}_q, \mathbb{N}_q) \in (\mathcal{T}, \mathcal{V}, \mathcal{V})$.

3.3.2 Semantic loss

We consider that additional semantic information is available during the learning, here expressed as classification labels (following Salvador et al. 2017). As defined in Equation 3.9, our $\mathcal{L}_{sem}$ is acting as regularizer capable of taking advantage of semantic information in the crossmodal task, without adding extra parameters to the architecture nor graph dependencies. To leverage class information, we construct triplets that optimize a surrogate of the k-nearest neighbor classification task. Ideally, for a given query x_q, and its corresponding class $c(x_q)$, we want its associated closest sample $x_{\star,q}$ in the feature space $\mathcal{F}$ to respect $c(x_q) = c(x_{\star,q})\ \forall q \in \mathbb{Q}$.

A natural approach for enforcing this constraint into the space is to optimize a class-aware metric learning loss such as the one proposed in (Weinberger et al. 2009). We thus propose a triplet loss $\mathcal{L}_{sem}$ in order to have a semantic structure to the space. $\mathcal{L}_{sem}$ is similar to $\mathcal{L}_{tri}$ (Equation 3.3), but it has different interactions between samples, as defined by the following sets of semantic positives $\mathbb{P}_{q,s}$ and semantic negatives $\mathbb{N}_{q,s}$:

$$\begin{aligned} \mathbb{P}_{q,s} &= \{x : c(x) = c(x_q)\} \\ \mathbb{N}_{q,s} &= \{x : c(x) \neq c(x_q)\} \end{aligned} \tag{3.12}$$

Contrary to the classification machinery adopted by (Salvador et al. 2017), $\mathcal{L}_{sem}$ optimizes semantic relations directly in the latent space $\mathcal{F}$ without changing the architecture of the network, as shown on Figure 3.3. This promotes a smoothing effect on the space by encouraging instances of the same class to stay closer to each other, instead of heavily overfitting to match their instance-based counterpart.

3.3.3 Adapting SGD update over Mini-batches

As commonly used in Deep Learning, we use the stochastic gradient descent (SGD) algorithm which approximates the true gradient over mini-batches. Several strategies can be explored to compute the update step in the context of the triplet loss. We present here the two main strategies found in the literature (*average* and *maximum*) and discuss their limitations. This leads us to our proposed gradient update step which we call *adaptive*. In the following, δ refers to the update term that is applied to the set of parameters at each mini-batch update, *i.e.*, $\theta(t+1) = \theta(t) - \eta\delta$.

The *average* strategy computes the loss functions considering all of the negative samples available, averaging each one by the number of samples used:

$$\delta_{avg} = \sum_{x_q \in \mathbb{Q}} \left(\sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,v}} \sum_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,v}} \frac{\nabla \mathcal{L}_{tri}(\theta, x_q, x_p, x_n)}{\beta_r} + \sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,s}} \sum_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,s}} \lambda \frac{\nabla \mathcal{L}_{sem}(\theta, x_q, x_p, x_n)}{\beta_s} \right) \tag{3.13}$$

Where $\beta_r = |\mathbb{Q}| \cdot |\mathbb{N}^{\mathbb{B}}_{q,v}| \cdot |\mathbb{P}^{\mathbb{B}}_{q,v}|$ and $\beta_s = |\mathbb{Q}| \cdot |\mathbb{N}^{\mathbb{B}}_{q,s}| \cdot |\mathbb{P}^{\mathbb{B}}_{q,s}|$, represent the total number of triplets for each loss. This is the strategy used in (Karpathy et al. 2015; Kiros, Salakhutdinov, et al. 2015). However, as remarked in (Faghri et al. 2017), the *average* strategy can lead to vanishing gradients since, as the optimization progresses, most of the triplets tend to contribute less (or not contribute at all) to the error.

To circumvent the vanishing gradient problem, the *maximum* strategy replaces the sum over all negatives by the selection of the hardest negative sample for each positive pair, as described by (Faghri et al. 2017). It can be directly applied to our average triplet loss, as follows:

$$\delta_{max} = \sum_{x_q \in \mathbb{Q}} \left(\sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,v}} \max_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,v}} \frac{\nabla \mathcal{L}_{tri}(\theta, x_q, x_p, x_n)}{\beta_r} + \sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,s}} \max_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,s}} \lambda \frac{\nabla \mathcal{L}_{sem}(\theta, x_q, x_p, x_n)}{\beta_s} \right) \tag{3.14}$$

We then consider $|\mathbb{N}^{\mathbb{B}}_q| = 1$ for calculating β_r and β_s.

While $\mathcal{L}_{max}$ directly optimizes the Recall@1 metric, as explained in (Faghri et al. 2017), it takes a few epochs to "warm up" at the beginning of the learning process. This phenomenon is explained by the very limited amount of triplets contributing to the gradient, when many are still violating the constraints. We also believe this problem is amplified as the size of the training set grows.

Adaptive AdaMine weighting To tackle these issues, our proposed *adaptive* strategy modifies δ_{avg}, discarding triplets with uninformative information (*i.e.*, zero loss):

$$\delta_{adm} = \sum_{x_q \in \mathbb{Q}} \left(\sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,v}} \sum_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,v}} \frac{\nabla \mathcal{L}_{tri}(\theta, x_q, x_p, x_n)}{\beta'_r} + \sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,s}} \sum_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,s}} \lambda \frac{\nabla \mathcal{L}_{sem}(\theta, x_q, x_p, x_n)}{\beta'_s} \right) \tag{3.15}$$

with β'_r and β'_s compensating for triplets that are not contributing to the cost:

$$\begin{aligned}\beta'_r &= \sum_{x_q \in \mathbb{Q}} \sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,v}} \sum_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,v}} \mathbb{1}_{\mathcal{L}_{tri} \neq 0} \\ \beta'_s &= \sum_{x_q \in \mathbb{Q}} \sum_{x_p \in \mathbb{P}^{\mathbb{B}}_{q,s}} \sum_{x_n \in \mathbb{N}^{\mathbb{B}}_{q,s}} \mathbb{1}_{\mathcal{L}_{sem} \neq 0}\end{aligned} \tag{3.16}$$

This strategy combines the benefits of δ_{avg} and δ_{max}. At the very beginning of the optimization, all triplets contribute to the cost and, as constraints stop being violated, they are dropped. At the end of the training phase, one would expect δ_{adm} to behave similarly to δ_{max}, because most of the triplets will have no contribution to it, leaving the hardest negatives to be optimized.

Some methods propose to reinforce the influence of negative samples by changing the loss function (Song et al. 2016; Lin et al. 2017). We prefer, in our approach, to explicitly define the weights associated to the gradients of the negative samples.

An important problem of multitask learning, properly presented and addressed by (Kokkinos 2017), is the possibility of bias in parts of the architecture that are shared between tasks. When the amount of samples for different tasks is unevenly balanced, so are the gradients responsible for the updates, leading to a set of parameters that is prone to favour the most frequently updated tasks. In that regard, an added benefit of δ_{adm} is due to the fact that each loss is independently normalized by its number of active triplets, keeping the trade-off between $\mathcal{L}_{tri}$ and $\mathcal{L}_{sem}$ unaffected by the noise induced by the triplet sampling. This provides a natural solution to the gradient balancing problem.

We evaluate our proposed methods in the crossmodal retrieval task of Recipe1M (Salvador et al. 2017), a dataset of cooking recipes and images, adopting the symmetric formulation of our losses, which considers both image and text modalities as source of queries.

3.4 Experimental setup

All of our experiments were conducted using PyTorch [3], with a reimplementation of the experimental setup (i.e. preprocessing, architecture and evaluation procedures) described by (Salvador et al. 2017).

Dataset (Salvador et al. 2017) introduced Recipe1M, a large dataset of one million pairwise aligned image and text documents corresponding to cooking recipes with matching pictures. The scale and the diversity of this dataset challenges tractable deep learning strategies for this cross-modal retrieval task.

The raw Recipe1M dataset is large-scale, with about 1 million image and recipe pairs. It is currently the largest one in this domain, including twice as many

3. http://pytorch.org

recipes as (Kusmierczyk et al. 2016) and eight times as many images as (J. Chen and C.-W. Ngo 2016). Furthermore, the presence of semantic information makes it particularly suited for multitask learning: around half of the pairs are associated with a class, among 1048 classes parsed from the recipe titles.

Evaluation procedures The models are evaluated on 10 unique subsets of 1,000 (1k setup) or 5 unique subsets of 10,000 (10k setup) randomly selected image-recipe pairs, and the mean results are reported for the median retrieval rank (MedR), as well as the recall percentage at top K (R@K), which corresponds to the percentage of queries for which the match is ranked among the top K closest results.

For the final model selection, we evaluate the MedR on the validation set at the end of each training epoch. We train each model for 80 epochs, keeping only the one with the best MedR on validation.

Architecture To embed the images from the raw pixels to the retrieval space, we adopt a pre-trained ResNet-50 model[4]. As for the recipes, a first module embed the list of ingredients with a bi-LSTM, following a pretrained embedding matrix built with word2vec (Mikolov et al. 2013). A second module embeds the list of cooking instructions with a hierarchical LSTM. The first LSTM (word-level) is pretrained using the skip-thoughts technique (Kiros, Zhu, et al. 2015) and is not fine tuned. The second LSTM (sentence-level) is learned from scratch. A third module produces the recipe features $\mathcal{T}$ by merging both textual modalities with a concatenation. Finally, after mapping either the visual or the textual modality into the joint feature space $\mathcal{F}$, a hyperbolic tangent function is applied to the resulting representation, which is then L2 normalized. This normalization ensures that the squared Euclidean distance is proportional to the cosine distance.

Training scheme As adopted by (Salvador et al. 2017), we use the Adam (Kingma et al. 2015) optimizer with a learning rate of 0.0001. Besides, we propose a simpler training scheme: At the beginning of the training phase, we freeze the ResNet-50 weights, optimizing only the text-processing branch, as well as the weights of the mapping of the visual processing branch. After 20 epochs, the weights of the ResNet-50 are unfrozen and the whole architecture is fine-tuned for 60 more epochs.

Triplet sampling As is common with triplet based losses in deep learning, we adopt a per-batch sampling strategy for estimating $\mathcal{L}_{ins}$ and $\mathcal{L}_{sem}$ (see Subsection 3.3.3). For the instance-based losses, our mini-batch is balanced and composed of 100 crossmodal (image-recipe) matching pairs. We define the positive instance

4. We also evaluated ResNet-152, but the results were comparable in spit of being slower to train.

set $\mathbb{P}^{\mathbb{B}}_{q,v}$ of each item q by selecting its matching crossmodal sample, and the negative instance set $\mathbb{N}^{\mathbb{B}}_{q,v}$ with the rest of the crossmodal samples.

Half of the pairs in each mini-batch $\mathbb{B}$ are randomly sampled from the classless set, while the other half is sampled from the set with known classes. For any pair (x_q, x_p) inside the mini-batch, where $c(x_q) = c(x_p)$, we make sure there is at least one other pair (x'_q, x'_p) associated with the same class, *i.e.*, $c(x_q) = c(x'_q) = c(x_p) = c(x'_p)$. This is a necessary condition for the creation of the per-batch semantic ensembles of positive matches $\mathbb{P}^{\mathbb{B}}_{q,s}$.

For each item q we then randomly select a single crossmodal sample that belongs to the same class, but not to the same instance, ensuring $|\mathbb{P}^{\mathbb{B}}_{q,s}| = 1$ and $\mathbb{P}^{\mathbb{B}}_{q,s} \neq \mathbb{P}^{\mathbb{B}}_{q,v}$. Finally, in order to guarantee the same number of negative samples for each query q, we randomly select n negative items for each q, with n being the size of the smallest negative ensemble inside the batch.

3.4.1 State-of-the-art comparison

We provide a quantitative evaluation of our proposed methods, compared to different baseline models, in Table 3.1. **CCA** denotes the Canonical Correlation Analysis method applied to the same pretrained embeddings that we use for all the reported methods. **PWC** denotes the pairwise loss with the classification layer from (Salvador et al. 2017). We report their state-of-the-art results for the 1k and 10k setups when available. **PWC++** denotes the improved version of **PWC**, which uses the positive margin (set to 0.3) as well as the negative margin (set to 0.9). Finally, **AdaMine** is a combination of the adaptive bidirectional instance and semantic triplet losses. Its margin is set to 0.3, and the weight λ for the semantic crossmodal triplets ($\mathcal{TV}_s$ and $\mathcal{VT}_s$) is set to 0.3.

First, we observe that adding a positive margin to the pairwise approach **PWC** results in an important improvement over the baseline model. This strategy allowed **PWC++** to achieve median ranks (medR) of 3.3 and 3.5 for the 1k setup, representing gains of 1.9 and 1.6 points over the results reported by (Salvador et al. 2017): 5.2 and 5.1. Similarly prominent gains were obtained for the 10k setup, of 7.3 and 4.2 points. Without the positive margin in **PWC**, all the matching pairs of image and recipe are forced to be mapped to the same point in the feature space $\mathcal{F}$. This behavior can lead to overfitting and sub-optimal solutions. By adding a positive margin we relax this constraint, obtaining a better solution.

Our triplet-based approach (**AdaMine**), however, surpasses the current state-of-the-art results by a large margin. For the 1k setup, it reduces the medR score by a factor of 5—from 5.2 and 5.1 to 1.0 and 1.0—, and by a factor bigger than 3 for the 10k setup. Moreover, it has fewer parameters than **PWC++** and **PWC**, since the feature space is directly optimized with a semantic loss, without the addition a parameter-heavy head to the model.

		MedR	R@1	R@5	R@10
1k setup		Image to Textual recipe			
	CCA (Salvador et al. 2017)	15.7	14.0	32.0	43.0
	PWC (Salvador et al. 2017)	5.2	24.0	51.0	65.0
	PWC++	3.3 ± 0.4	25.8 ± 1.6	54.5 ± 1.3	67.1 ± 1.4
	AdaMine	$\mathbf{1.0} \pm 0.1$	$\mathbf{39.8} \pm 1.8$	$\mathbf{69.0} \pm 1.8$	$\mathbf{77.4} \pm 1.1$
		Textual recipe to Image			
	CCA (Salvador et al. 2017)	24.8	9.0	24.0	35.0
	PWC (Salvador et al. 2017)	5.1	25.0	52.0	65.0
	PWC++	3.5 ± 0.5	24.8 ± 1.1	55.0 ± 1.8	67.1 ± 1.2
	AdaMine	$\mathbf{1.0} \pm 0.1$	$\mathbf{40.2} \pm 1.6$	$\mathbf{68.1} \pm 1.2$	$\mathbf{78.7} \pm 1.3$
10k setup		Image to Textual recipe			
	PWC (Salvador et al. 2017)	41.9	-	-	-
	PWC++	34.6 ± 1.0	7.6 ± 0.2	19.8 ± 0.1	30.3 ± 0.4
	AdaMine	$\mathbf{13.2} \pm 0.4$	$\mathbf{14.9} \pm 0.3$	$\mathbf{35.3} \pm 0.2$	$\mathbf{45.2} \pm 0.2$
		Textual recipe to Image			
	PWC (Salvador et al. 2017)	39.2	-	-	-
	PWC++	35.0 ± 0.9	6.8 ± 0.2	21.5 ± 0.2	28.8 ± 0.3
	AdaMine	$\mathbf{12.2} \pm 0.4$	$\mathbf{14.8} \pm 0.3$	$\mathbf{34.6} \pm 0.3$	$\mathbf{46.1} \pm 0.3$

Table 3.1 – **Comparison of AdaMine and the SOTA.** MedR means Median Rank (lower is better). R@K means Recall at K (between 0% and 100%, higher is better). The mean and std values over 10 (resp. 5) bags of 1k (resp. 10k) pairs each are reported for the top (resp. bottom) table.

Choosing triplet over pairwise losses leads to better results. This is explained by the triplet's ability to better model the ranking constraints into the representation space, as discussed in Subsection 3.2.2. Contrarily to pairwise, it is able to apply a flexible margin, that takes into account the anchor's distance to both positive and negative samples, instead of a hard margin, directly applied to these distances.

Due to the combination of an instance-based cost and a semantic-based one, we are able to enrich the retrieval space without adding costly modules to the architecture. Furthermore, our adaptive mining strategy combines the benefits of the *average* and the *maximum* ones, described in Subsection 3.3.3, while also dealing with their shortcomings. The combination of these proposals allowed **AdaMine** to obtain the best score for all evaluated metrics.

With the intention of better understanding the individual influence of each contribution we presented, we develop, in the following section, extra experiments that test them.

3.4.2 Further analyses

				Image to Textual recipe			
	Retrieval loss	Semantic loss	Sampling	MedR	R@1	R@5	R@10
(1)	Random	-	-	5000	0.0	0.1	0.1
(2)	-	$\mathcal{L}_{bitri}$	adm	207.3	1.4	5.7	9.6
(3)	$\mathcal{L}_{bitri}$	-	avg	16.2	12.6	30.9	41.4
(4)	$\mathcal{L}_{bitri}$	-	adm	15.4	13.3	32.1	42.6
(5)	$\mathcal{L}_{bitri}$	Classification	adm	14.8	13.6	32.7	43.2
(6)	$\mathcal{L}_{bitri}$	$\mathcal{L}_{bitri}$	avg	14.2	14.1	33.3	43.9
(7)	$\mathcal{L}_{bitri}$	$\mathcal{L}_{bitri}$	adm	**13.2**	**14.9**	**35.3**	**45.2**
				Textual recipe to Image			
	Retrieval loss	Semantic loss	Sampling	MedR	R@1	R@5	R@10
(1)	Random	-	-	5000	0.0	0.1	0.1
(2)	-	$\mathcal{L}_{bitri}$	adm	205.4	1.4	5.4	9.1
(3)	$\mathcal{L}_{bitri}$	-	avg	17.2	11.7	29.8	40.4
(4)	$\mathcal{L}_{bitri}$	-	adm	15.8	12.3	31.1	41.7
(5)	$\mathcal{L}_{bitri}$	Classification	adm	15.2	12.9	31.8	42.5
(6)	$\mathcal{L}_{bitri}$	$\mathcal{L}_{bitri}$	avg	14.4	13.3	32.5	43.5
(7)	$\mathcal{L}_{bitri}$	$\mathcal{L}_{bitri}$	adm	**12.2**	**14.8**	**34.6**	**46.1**

Table 3.2 – **Extra analyses of the AdaMine components.** MedR means Median Rank (lower is better). R@K means Recall at K (between 0% and 100%, higher is better). The mean value over 5 bags of 10,000 pairs each is reported.

In Table 3.2, we conduct deeper analyses of our model, as to highlight the contribution of each of our proposals to the final results. In order to better show the nuances between different approaches, all experiments in this section are conducted with the 10k setup. To simplify the discussion, we consider the MedR in the im2recipe task for our analyses, but our arguments stand for the other evaluation metrics as well (reported in the Table 3.2).

Adaptive sampling validation We start by comparing our adaptive sampling strategy (*adm*) with the average one (*avg*), adopted by (Karpathy et al. 2015; Kiros, Zhu, et al. 2015). First, we disconsider the influence of the semantic loss, by taking the lines (3)-*avg* and (4)-*adm*. The (3)-*avg* achieves 16.2 MedR, while the adaptive scheme (4)-*adm* obtains an improvement of 0.8 over it, with the score of 15.4. The same experiments, this time considering the semantic loss, are the (6)-*avg* and the (7)-*adm*. They display the same behavior: (6)-*avg* achieves 14.2 MedR, and (7)-*adm* 13.2. In this scenario, the gain is of 1.0. It is worth mentioning that we also tried the *maximum* sampling strategy (Faghri et al. 2017), but it was very unstable and

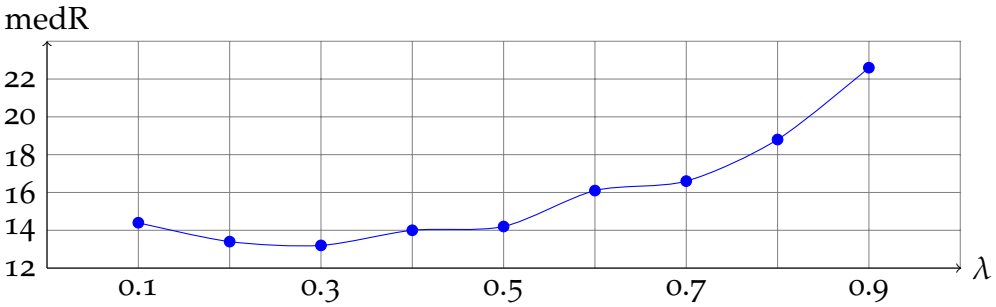


Figure 3.4 – Scores for **different values of the λ hyper-parameter**, responsible for weighting the semantic regularization cost $\mathcal{L}_{sem}$ of **AdaMine**, calculated over 5 bags of 10.000 validation samples.

did not succeeded in converging on this large-scale dataset. These experiments validate the efficiency of *adm* as the negative sampling procedure, therefore we will adopt it for the next analyses.

Semantic loss impact We then quantify how decisive is the use of the semantic information in our learning process. Starting from the random model (line 1), which achieves 5000 of MedR, we show that the semantic loss alone (line 2) achieves far better results: 207.3. This experiment highlights the capacity of the semantic loss to organize the feature space without using the instance information. Next, we point that the instance-based triplet model with additional semantic information (line 5) reaches better results than the one without it—respectively, 14.8 and 15.4 of MedR, a gain of 0.6 points. Furthermore, we show that our semantic loss, proposed in Subsection 3.3.2 (line 7), outperforms the method proposed in (Salvador et al. 2017) (line 5), which adds a classification head C to the model. (5) achieves 14.8 MedR, and (7) 13.2, a difference of 1.6 points. These experiments confirm the importance of additional semantic clues: despite having one million less parameters, our approach achieves better scores, when compared to the addition of the classification head.

Finally, we conduct an analysis of the hyperparameters of **AdaMine**. Figure 3.4 illustrates the impact of λ, responsible for weighting the semantic regularization cost $\mathcal{L}_{sem}$. Although we observe a fair level of robustness for lower values of λ, any value over 0.5 has a hindering effect on the retrieval task, since the semantic grouping starts to be of considerable importance. We also performed a similar test with the margin α, presented in Figure 3.5, where we observe the method works well for any sensibly-chosen low value of α.

Ablation studies In Table 3.3 and Table 3.4, we also consider several scenarios where we only use parts of our model. The suffix *_sem* indicates the semantic loss alone, while *_ins* indicates the instance-wise loss alone, *_ins+cls* the instance-wise

		Image to Textual recipe			
		MedR	R@1	R@5	R@10
SOTA	Random	499	0.0	0.0	0.0
	CCA [1]	15.7	14.0	32.0	43.0
	PWC [1]	5.2	24.0	51.0	65.0
	PWC [1]*	5.0 ± 0.4	22.8 ± 1.4	47.7 ± 1.4	60.1 ± 1.4
	PWC++	3.3 ± 0.4	25.8 ± 1.6	54.5 ± 1.3	67.1 ± 1.4
Model scenarios	AdaMine_sem	21.1 ± 2.0	8.7 ± 0.7	25.5 ± 0.9	36.5 ± 0.9
	AdaMine_ins	1.5 ± 0.5	37.5 ± 1.1	67.0 ± 1.3	76.8 ± 1.5
	AdaMine_ins+cls	1.1 ± 0.3	38.3 ± 1.6	67.5 ± 1.2	78.0 ± 0.9
	AdaMine_avg	2.3 ± 0.5	30.6 ± 1.1	60.3 ± 1.2	71.4 ± 1.3
	AdaMine_ingr	4.9 ± 0.5	22.6 ± 1.4	48.5 ± 1.6	59.8 ± 1.3
	AdaMine_instr	3.9 ± 0.5	24.4 ± 1.6	52.6 ± 2.0	65.4 ± 1.6
	AdaMine	$\mathbf{1.0 \pm 0.1}$	$\mathbf{39.8 \pm 1.8}$	$\mathbf{69.0 \pm 1.8}$	$\mathbf{77.4 \pm 1.1}$

		Textual recipe to Image			
		MedR	R@1	R@5	R@10
SOTA	Random	499	0.0	0.0	0.0
	CCA [1]	24.8	9.0	24.0	35.0
	PWC [1]	5.1	25.0	52.0	65.0
	PWC [1]*	5.3 ± 0.4	21.2 ± 1.2	48.0 ± 1.1	60.4 ± 1.4
	PWC++	3.5 ± 0.5	24.8 ± 1.1	55.0 ± 1.8	67.1 ± 1.2
Model scenarios	AdaMine_sem	21.1 ± 1.9	8.2 ± 0.9	25.5 ± 1.0	36.2 ± 0.9
	AdaMine_ins	1.6 ± 0.5	36.1 ± 1.6	66.6 ± 1.3	76.8 ± 1.5
	AdaMine_ins+cls	1.2 ± 0.4	37.5 ± 1.4	67.7 ± 1.2	77.3 ± 1.0
	AdaMine_avg	2.2 ± 0.3	30.6 ± 1.8	60.6 ± 1.1	71.9 ± 1.1
	AdaMine_ingr	5.0 ± 0.6	21.5 ± 1.4	47.7 ± 2.1	59.8 ± 1.8
	AdaMine_instr	3.7 ± 0.5	23.6 ± 1.7	52.7 ± 1.6	65.5 ± 1.5
	AdaMine	$\mathbf{1.0 \pm 0.1}$	$\mathbf{40.2 \pm 1.6}$	$\mathbf{68.1 \pm 1.2}$	$\mathbf{78.7 \pm 1.3}$

Table 3.3 – Detailed **comparison of AdaMine components on 1k samples.** MedR means Median Rank (lower is better). R@K means Recall at K (between 0% and 100%, higher is better). The mean and std values over 10 bags of 1,000 pairs each are reported. Items marked with a star (*) are our reimplementation of the cited methods. [1] results from (Salvador et al. 2017).

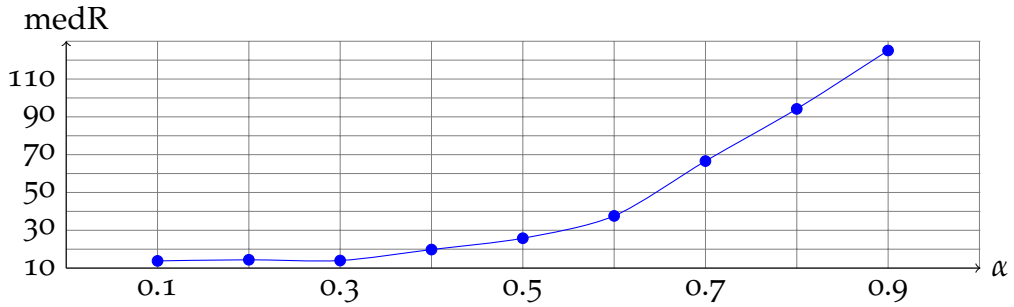


Figure 3.5 – Scores for **different values of the α hyper-parameter**, which defines the triplet margin for **AdaMine**, calculated over 5 bags of 10.000 validation samples.

loss with the classification branch proposed by (Salvador et al. 2017), *_avg* our model with the average triplet sampling (Equation 3.13), and *_ingr* and *_instr* the ingredient and the instruction branches alone. For this experiment, the max strategy (Equation 3.14) did not converge in any of the multiple runs.

We show that our approach is able to achieve results close to the State-Of-The-Art (SOTA) even with missing information, e.g. knowing the ingredients but not the instructions, and that replacing the *average* strategy by our *adaptive* one greatly improves the results, since the two losses we adopted can be correctly balanced during the training phase. We also highlight that the semantic loss is more efficient, since it achieves lower retrieval scores when compared to adding the classifier branch.

3.5 Conclusion

In this chapter, we performed the alignment of cross-modal feature spaces, tackling the retrieval task.

Our motivations came from the very roots of the retrieval's task, and its quest for answering the question of *how to correctly organize feature spaces?*

A number of options for performing this kind of alignment were proposed on the scientific literature, and some of them were discussed in this work. However, each of them also comes with shortcomings, either in the form of increasing the number of parameters of the model, or increasing the computational cost of learning.

We presented a new multi-task loss, which exploits both instance-based and semantic-based training information in a single learning framework. Our main contribution is a simple to implement and scalable regularization scheme, which incorporates semantic information via an auxiliary task using the metric learning framework. The strategy we call AdaMine, for adaptively-mined triplet loss,

		Image to Textual recipe			
		MedR	R@1	R@5	R@10
	PWC++ (best SOTA)	34.6 ± 1.0	7.6 ± 0.2	19.8 ± 0.1	30.3 ± 0.4
Model scenarios	AdaMine_sem	207.3 ± 3.9	1.4 ± 0.3	5.7 ± 0.3	9.6 ± 0.3
	AdaMine_ins	15.4 ± 0.5	13.3 ± 0.2	32.1 ± 0.7	42.6 ± 0.8
	AdaMine_ins+cls	14.8 ± 0.4	13.6 ± 0.2	32.7 ± 0.4	43.2 ± 0.3
	AdaMine_avg	24.6 ± 0.8	10.0 ± 0.2	25.9 ± 0.4	35.7 ± 0.5
	AdaMine_ingr	52.8 ± 1.2	6.5 ± 0.2	17.9 ± 0.2	25.8 ± 0.3
	AdaMine_instr	39.0 ± 0.9	6.4 ± 0.1	18.9 ± 0.4	27.6 ± 0.5
	AdaMine	$\mathbf{13.2 \pm 0.4}$	$\mathbf{14.9 \pm 0.3}$	$\mathbf{35.3 \pm 0.2}$	$\mathbf{45.2 \pm 0.2}$
		Textual recipe to Image			
		MedR	R@1	R@5	R@10
	PWC++ (best SOTA)	35.0 ± 0.9	6.8 ± 0.2	21.5 ± 0.2	28.8 ± 0.3
Model scenarios	AdaMine_sem	205.4 ± 3.2	1.4 ± 0.1	5.4 ± 0.2	9.1 ± 0.4
	AdaMine_ins	15.8 ± 0.7	12.3 ± 0.3	31.1 ± 0.5	41.7 ± 0.6
	AdaMine_ins+cls	15.2 ± 0.4	12.9 ± 0.3	31.8 ± 0.3	42.5 ± 0.2
	AdaMine_avg	24.0 ± 0.6	9.2 ± 0.4	25.4 ± 0.5	35.3 ± 0.4
	AdaMine_ingr	53.8 ± 0.7	5.8 ± 0.3	17.3 ± 0.2	25.0 ± 0.2
	AdaMine_instr	39.2 ± 0.7	5.7 ± 0.4	17.9 ± 0.6	26.6 ± 0.5
	AdaMine	$\mathbf{12.2 \pm 0.4}$	$\mathbf{14.8 \pm 0.3}$	$\mathbf{34.6 \pm 0.3}$	$\mathbf{46.1 \pm 0.3}$

Table 3.4 – Detailed **comparison of AdaMine components on 10k samples.** MedR means Median Rank (lower is better). R@K means Recall at K (between 0% and 100%, higher is better). The mean and std values over 5 bags of 10,000 pairs each are reported. Items marked with a star (*) are our reimplementation of the cited methods. [1] results from (Salvador et al. 2017).

samples triplet constraints from the classification labels, and these constraints are added as a regularization term to our objective function that deals with the cross-modal alignment.

A second contribution is the exploration of solutions for selecting informative triplets in our large scale optimization context and, in particular, an adaptive strategy for negative mining, which provides more stable and faster convergence when compared to other approaches. AdaMine is evaluated on the very large and challenging Recipe1M cross-modal dataset, outperforming the state-of-the-art models. As noticed in (Faghri et al. 2017), the classical gradient averaging over the mini-batches examples used in deep nets struggles to learn precise embeddings. However, (Faghri et al. 2017) also noticed that their hard negative mining strategy suffers from a slow start, a phenomenon amplified by our large scale context. To

overcome both difficulties, we introduced an adaptive strategy that efficiently tunes the gradient update by weighting all the (hard) negative constraints.

Finally, AdaMine achieved large improvements over the classical approaches one adopted by (Kiros, Salakhutdinov, et al. 2015; Faghri et al. 2017), proving itself as a useful addition for when several triplets are simultaneously optimized. We also show how the triplet approaches are superior to the pairwise approaches adopted in the literature, and compare two methods to incorporate semantic information. We present many experiments on a large multi-modal dataset, and our strategy outperforms the state-of-the-art by a large margin.

As a side note, to prove the flexibility of AdaMine, it was hot-tested on the Google Landmark Retrieval Challenge[5], and obtained a silver medal. It is important to notice, however, that no validation nor tuning of hyper-parameters was performed. The model was trained, and then submitted in the last hour of the competition, without searching for values for α, λ, nor choosing the number of training epochs. With more time and proper validation, we believe AdaMine could have ranked even better.

5. https://www.kaggle.com/c/landmark-retrieval-challenge

CHAPTER 4

RETRIEVAL IN THE COOKING CONTEXT

Contents

Chapter abstract

Recent advances in the machine learning community allowed a wide range of new applications to emerge. Its association to domains like cooking created the computational cuisine, smart cooking, and other related fields. We build on the work presented in Chapter 3, adopting the large-scale retrieval task of the Recipe1M dataset, composed of one million image-recipe pairs and additional class information, in which the goal is to find recipes based on picture or pictures based on recipes. We conduct an in-depth exploration of the representation space constructed by AdaMine, showing that subtasks never seen by the network can be performed thanks to the semantic structure imposed by our double loss scheme. Our results indicate that our method is able to structure the representation space in a way that it is possible to discriminate between specific ingredients inside the recipes or their visual content. Particularly to the food domain, the findings of this chapter reveal the potential of applications concerning dietary restrictions and menu planning.

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

— Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, and Matthieu Cord (2018). "Images & Recipes: Retrieval in the cooking context". In: *IEEE International Conference on Data Engineering (ICDE), Data Engineering meets Intelligent Food and Cooking Recipe (DECOR) workshop*

4.1 Introduction

Cooking is one of the most fundamental human activities connected to various aspects of human life such as food, health, dietary, culinary art, and so on. Data mining and machine learning techniques have been used to extract and clean large datasets of recipes from the Internet, and also to plan and analyze the recipe instructions.

Consequently, smart cooking has recently become the center of increased interest as shown by the development of related workshops such as the Workshop on Multimedia for Cooking and Eating Activities (CEA 2017).

One difficulty underlying computational cooking lies in the nature of the data, since recipes generally include images and text, structured or not (e.g., the list of ingredients or instructions in natural language). This opens several challenges in terms of indexing/storing and gives rise to numerous application tasks, such as recommendation or classification. Computational cooking has consequently emerged as a new research topic that also benefits from recent advances in machine learning based on deep neural approaches.

Recent strategies aim at projecting data elements into a latent semantic space in a way that similar elements are represented with similar low-dimensional features (Harris 1954; Mikolov et al. 2013). Beyond solving indexing issues, these latent representations, also called *embeddings*, allow machines to perceive texts and images in a meaningful way, similar to that of humans. This perception can be exploited in smart cooking-oriented tasks, such as ingredient identification (J. Chen and C.-w. Ngo 2016), recipe recognition (X. Wang et al. 2015), or recipe popularity prediction (Sanjo et al. 2017).

The increasing importance of food-related tasks in computer vision has motivated the creation of many datasets. M. Chen et al. 2009, for example, proposed the Pittsburgh fast-food image dataset, Kawano et al. 2014 release a dataset containing mainly-Japanese food categories, and Farinella et al. 2015 another one containing distinct plates. However, these datasets are small in size, containing hundreds to a few thousands images.

In order to solve complex tasks like these ones in a reliable manner, richer sets of images were required. Other initiatives provided larger datasets, with more diversity and a bigger number of samples, which were fundamental for the usage of deep networks. For example, Bossard et al. 2014 proposed the Food-101 dataset, containing around 101,000 images of 101 different categories. In the same spirit, X. Wang et al. 2015 introduced a twin dataset, containing the same categories, but with recipes associated to each picture. Finally, the dataset proposed by J. Chen and C.-w. Ngo 2016 is similar in the number of images, while also including relations to 353 ingredients and 65 thousand recipes. Other ideas, like the one of Beijbom et al. 2015, involve taking advantage of the GPS data, used to retrieve

nearby restaurants and to match a picture taken by the user to items from the menu, this allowed important information, like nutritional values, to be recovered.

However, a large-scale dataset containing images and recipes was not available until very recently, when Salvador et al. 2017 published Recipe1M. They collected nearly 1 million recipes, with about 800,000 images associated to them. To the best of our knowledge, up to date this is the biggest dataset of its kind, and it is adopted in our experimental apparatus on this chapter.

In Chapter 3, we presented AdaMine, a new learning strategy developed for an architecture with two deep neural networks. It was initially designed for the cross-modal retrieval problem, with a particular application to the computational cooking, as it was trained on the Recipe1M dataset.

After training the multi-modal architecture with our learning strategy, we are now interested in smart cooking, the retrieval task between recipe modalities (namely recipe texts and dish pictures), and in experimenting with AdaMine in this setup to better understand the representations refined by it. Therefore, we propose a broader study of the model we previously trained. We will analyze the base task, retrieving relevant pictures of a meal given its recipe or, conversely, a relevant recipe given an image query, as well as different usages of our framework, querying for specific ingredients and removing ingredients from a recipe.

This chapter is organized as follows:

— In Section 4.2 we present our test apparatus, discussing our experimental goals, with special focus on extra tasks like ingredient retrieval and ingredient exclusion;
— We then conduct in Section 4.3 an exploration of the representations created by our model, showing applications and properties, like the semantic-wise and instance-wise neighborhood relations;
— Finally, in Section 4.4 we discuss the experimental results we obtained, highlighting the potential of our model for solving traditional and promising computational cooking use-cases. We analyze several downstream tasks to exploit the proposed model (in terms of learned architecture and/or representations) and qualitatively demonstrate its effectiveness.

4.2 Apparatus

The global architecture adpted in our experiments is depicted in Table 4.1, as well as few excerpts from the dataset. This model is the same model we trained in Chapter 3, with the same learning scheme. It consists of two deep neural network branches that respectively map each modality (image or text recipe) into a common representation semantic space in which they can be compared. This architecture is further detailed in Section 3.3.

Recipe1M is designed to tackle the problem of cross-modal retrieval, in which the user wants to retrieve images of a dish by providing its recipe or, conversely, retrieving a recipe by providing a picture of the meal. These tasks are represented in Figure 4.2 and Figure 4.1, respectively. The presence of high level semantic information in the dataset, such as food classes, allows us to enforce a semantic structure on the learned representations (*i.e.*, items from the same class to have similar representations).

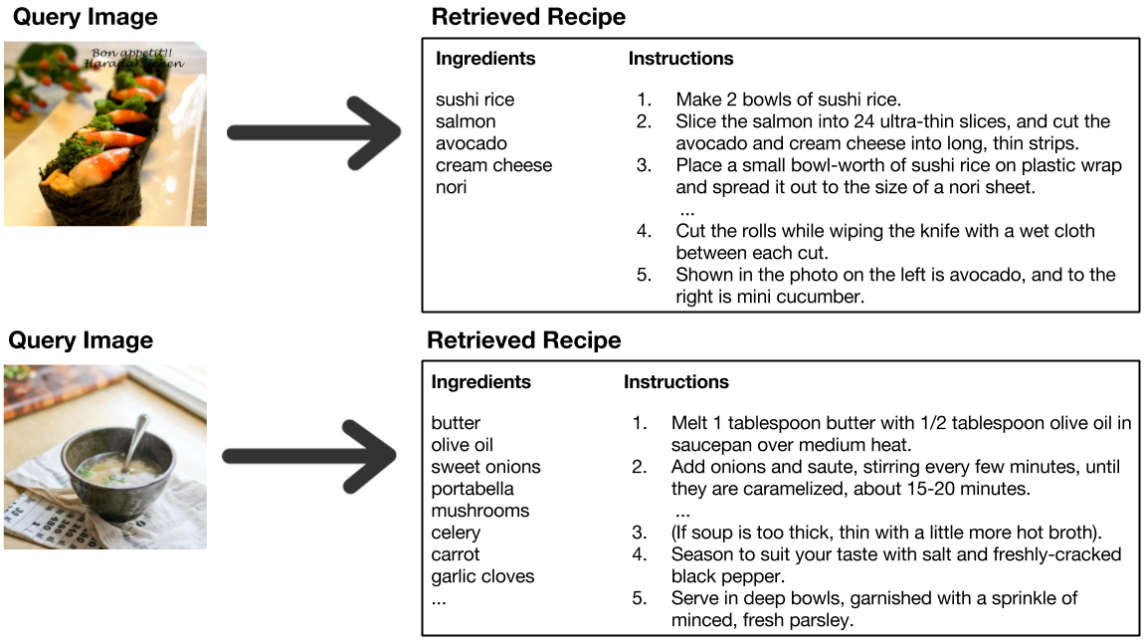
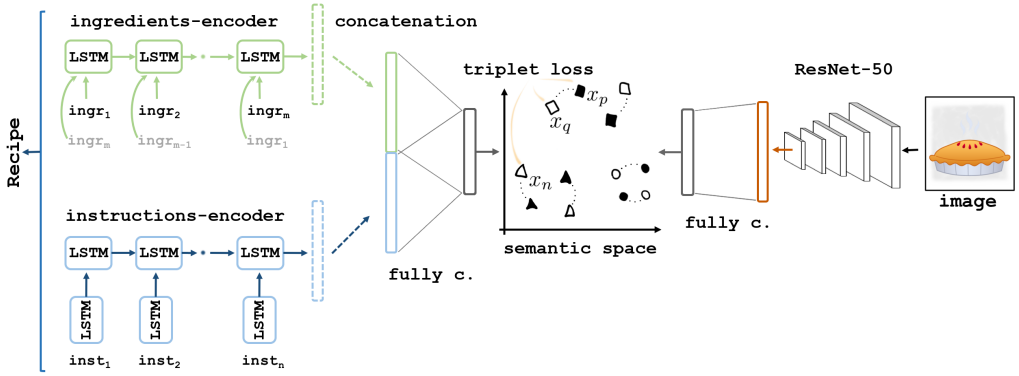


Figure 4.1 – **Recipe1M's task of textual recipe retrieval from image.** Given an image, the goal is to find the recipe that corresponds to that specific image. This recipe not only belongs to the same class of the image, but it also corresponds to the same instance (Adapted from Salvador et al. 2017).

We use the model we trained with AdaMine in our experiments, and we are interested in understanding how points inside these spaces change when the initial data changes. In other words, we want to visualize the impacts of changes in the input inside the representation space, to check if it sensible to small deformations, even if they were not explicitly taught to the model. For example, we would expect to see similar recipes being retrieved if we made small changes to the recipe, since the space should have clusters containing recipes of the same class, imposed by the semantic loss. For this objective, we propose the tests:



	ingr (ingredients)	instr (cooking instructions)	image
Pizza	1. *pizza dough* 2. *hummus* 3. *arugula* 4. *cherry / grape tomatoes* 5. *pitted greek olives* 6. *crumbled feta cheese*	1. *Cut the dough into two 8-ounce sized pieces.* 2. *Roll the ends under to create round balls.* 3. *Then using a well-floured rolling pin, roll the dough out into 12-inch circles.* 4. *Place the dough circles on sheets of parchment paper.*	
Pecan Pie	1. *unsalted butter* 2. *eggs* 3. *condensed milk* 4. *sugar* 5. *vanilla extract* 6. *chopped pecans* 7. *chocolate chips*	1. *Preheat the oven to 375 degrees F.* 2. *In a large bowl, whisk together the melted butter and eggs until combined.* 3. *Whisk in the sweetened condensed milk, sugar, vanilla, pecans, chocolate chips, butterscotch chips, and coconut.*	

Table 4.1 – **Overview of our multi-modal retrieval system.** The multi-modal retrieval neural network is shown on the top, and examples of inputs issued from the large-scale Recipe1M dataset on the bottom.

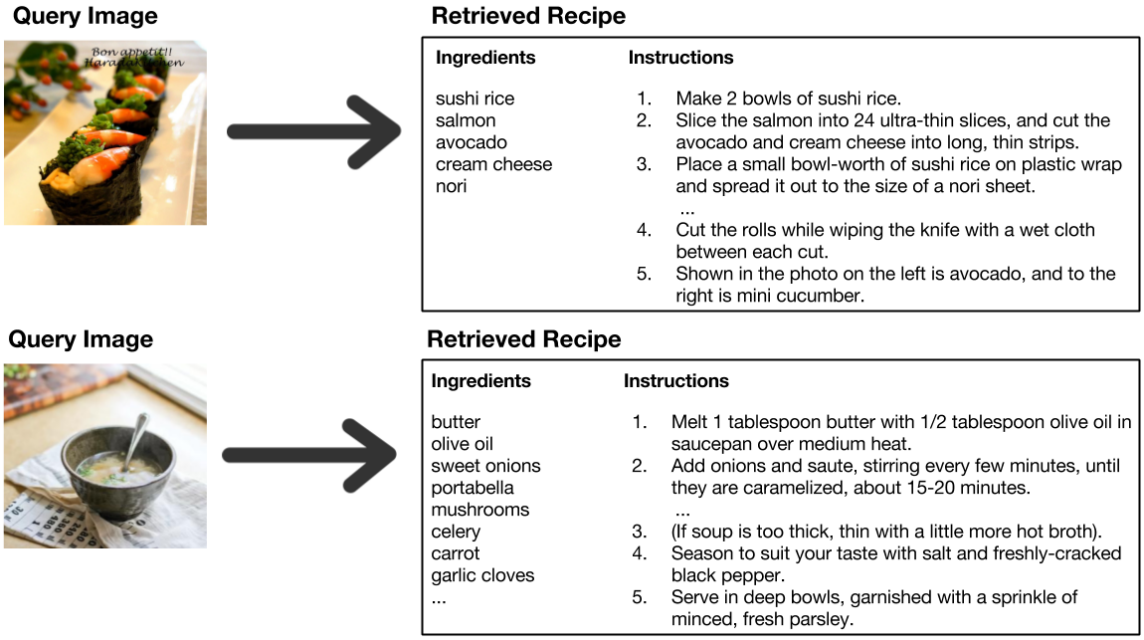


Figure 4.2 – **Recipe1M's task of image retrieval from textual recipe.** Given a textual recipe, the goal is to find the image that corresponds to that specific recipe. This image not only belongs to the same class of the recipe, but it also corresponds to the same instance (Adapted from Salvador et al. 2017).

— **Visualize the feature space** to check if it is correctly organized with the semantic structure and the instance-based information imposed during the training phase;

— **Study the standard retrieval** for the dataset, giving a recipe to the model and verifying the images we retrieve. In particular, we want to know if they correspond to the same instance of the recipe (i.e. picture taken from that exact recipe), and if they are semantically related to the query (same class of the recipe);

— **Different modalities for retrieval** can be used, checking if we are able to retrieve any known modality with our network, no matter which one was used for the query;

— **Retrieve recipes with a specific ingredient**, in order to see if we are able to search for any recipe containing an ingredient, or even recipes inside a specific class that contain the same recipe;

— **Perform ingredient exclusion**, to test if we are able to remove an ingredient from a recipe and retrieve alternative recipes from the same class, but not containing the specified ingredient.

4.3 Feature Space Exploration

In this section, we qualitatively evaluate the behavior of AdaMine, with the goal of better understanding its learned embedding spaces.

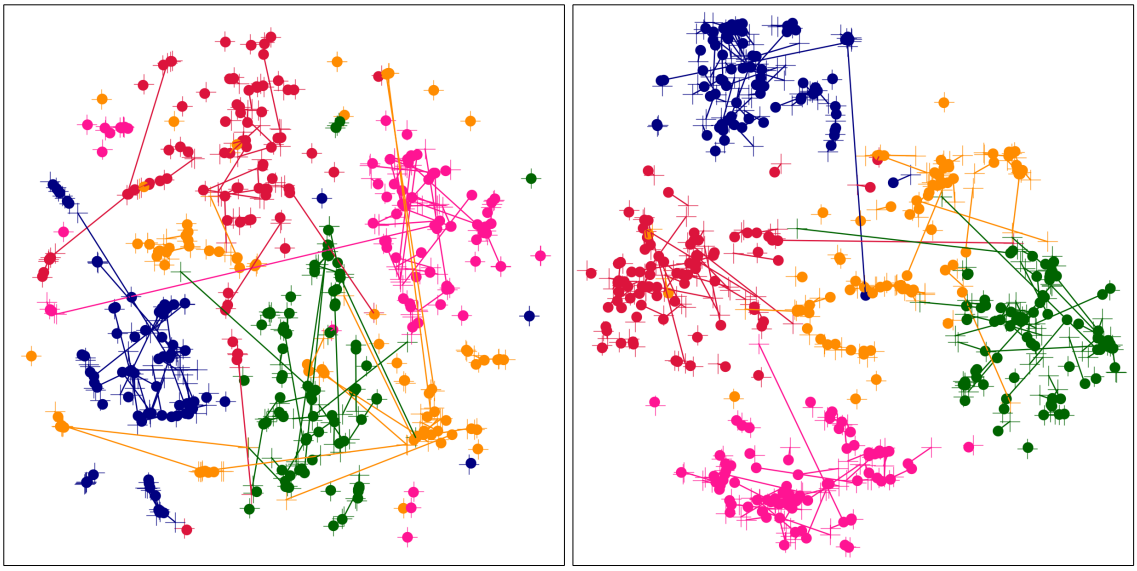
Our model has a twofold objective (namely, cross-modal retrieval and multi-modal representation) which could be beneficial for several cooking-related tasks. Its strength relies on the fact that, on one hand, it is capable of performing retrieval tasks, and on the other hand, the learned representation space allows to identify similar/dissimilar text and visual items. We believe that similar systems may be beneficial for ambitious tasks, such as menu or shopping list generation, or calorie tracking, since these tasks require more insight in terms of model design, as they might include diversity factors, ingredient quantity analysis, or external knowledge in a task-oriented model.

In this chapter, we focus particularly on downstream tasks for which these properties can be exploited. We provide illustrative examples issued from the our model and, for better readability, we always show the results as images, even for text recipes for which we display their corresponding original picture.

Visualization We first want to roughly visualize the representation space issued from the model trained with our technique, with particular focus on its organization and quality of alignment. In other words, we want to see if matching image and recipe items are close together. Additionally, we would like to observe the semantic disposition of this space, verifying the ability of AdaMine to form clusters of items belonging to the same recipe class.

To do so, we first select 5 classes among the most occurring ones in our dataset. We then randomly sample 80 image-recipe pairs for each of the selected classes, and extract their embeddings using our two networks, applying the t-SNE algorithm (Maaten et al. 2008) to project the data points from their original space to a 2 dimensional space.

In Figure 4.3 we show the differences between the feature spaces created by AdaMine and by the instance-only triplet loss. The image is constructed using the representations of the 400 pairs we randomly selected from the 5 classes mentioned above. We observe that the manifold produced by our method displays better semantic structure. This is due to our double loss, described in Subsection 3.3.2, capable of taking advantage of class information in order to create neighborhood constraints.



(a) Triplet (b) AdaMine

Figure 4.3 – **t-SNE visualization of AdaMine's and Triplet's semantic spaces.** Image (resp. Recipe) points are denoted with the + (resp. •) symbol. Matching pairs are connected with a trace. Blue points are associated to the cupcake class, orange to hamburger, pink to green beans, green to pork chops, and red to pizza.

4.3.1 The Retrieval Task

We continue our analysis by exploring the retrieval task. In the first part of this exploration, we will explore the image retrieval from a recipe query, and in the second part we will study the impacts of different modalities being used for the query and the results.

Image Retrieval We start tackling the recipe-to-image retrieval task, whose goal is to find the picture of a dish, given its recipe in textual form. In Table 4.2 and Table 4.4, we compare AdaMine (top row) and the instance-based triplet (bottom row) on four random recipe queries, for which both models are able to rank the correct match in the top-5 among 10,000 candidates.

For recipes of cucumber salad and roasted chicken, in the first and second rows of Table 4.2, respectively, both models are able to retrieve the matching picture in the first position. However, by analyzing the ranking of the top-5 images, we see that AdaMine is able to retrieve images that are semantically related to the query, while the instance-based triplet lacks the ability to enforce a semantic structure into the space, creating mixed neighborhoods. All of the top-5 results selected by AdaMine share critical ingredients with the recipes: cucumber and chicken; while the instance-based seems to focus on the yogurt and on the potatoes.

Ingredients query
Yogurt, cucumber, salt, garlic clove, fresh mint.

Cooking instructions query
Stir yogurt until smooth. Add cucumber, salt, and garlic. Garnish with mint. Normally eaten with pita bread. Enjoy!

Top 5 retrieved images

Ingredients query
Olive oil, balsamic vinegar, thyme, rosemary, brown sugar, lemons, chicken drumsticks with bones and skin, garlic, potatoes, parsley.

Cooking instructions query
Whisk together oil, mustard, vinegar, and herbs. Season to taste with a bit of salt and pepper and a large pinch or two of brown sugar. Cut lemon in half, zest that half and reserve; juice the same lemon half and add juice to marinade. Place chicken in a non-metal dish and pour marinade on top to coat. [...]

Top 5 retrieved images

Table 4.2 – **Recipe-to-images visualization, part 1.** For each recipe, we have the top row, indicating the top 5 images retrieved by our AdaMine model for a given recipe query, and the bottom row, indicating the top 5 images by the triplet loss for the same recipe. In cyan, the matching image. In blue, images belonging to the same class than the recipe. In red, images belonging to a different class.

As for the first and second rows of Table 4.4, for pizza and chocolate chip, respectively, AdaMine is able to rank both the matching image and semantically connected samples in a more coherent way. The pictures corresponding to the recipe are ranked first in both cases, and the rest of the top-5 results belongs to the same class of the query recipe. We believe this difference is due to a better alignment of the retrieval space produced by the semantic modeling in our training procedure.

Multi-modal Retrieval Following the Image Retrieval analysis, we would like to expand it to fully take advantage of the multi-modal scenario. We explore in depth the ability of our model to perform retrieval from any modality to any other modality, therefore we are interested in four retrieval scenarios: image-to-text, text-to-image, text-to-text, and image-to-image.

Specifically for the smart cooking tasks, this type application can be useful when the one wants the recipe of a meal found in a restaurant, or to identify similar recipes should they like to replace a meal in their menu by another one. In our framework, solving this task sums up to retrieving similar items in the semantic space (*i.e.*, the ones with the smallest distances with respect to the query).

	Ingredients	**Cooking instructions**	**Image**
Crunchy Onion Potato Bake	*Milk, Water, Butter, Mashed potatoes, Corn, Cheddar cheese, French-fried onions*	*Preheat oven to 350 degrees Fahrenheit. Spray pan with non stick cooking spray. Heat milk, water and butter to boiling; stir in contents of both pouches of potatoes; let stand one minute. Stir in corn. Spoon half the potato mixture in pan. Sprinkle half each of cheese and onions; top with remaining potatoes. Sprinkle with remaining cheese and onions. Bake 10 to 15 minutes until cheese is melted. Enjoy !*	

Table 4.3 – Query used in the multi-modal retrieval tasks.

To begin, we randomly chose a query item, shown in Table 4.3. The multi-modal data from this dish (picture and recipe) will be used for testing our four scenarios. Then, in Table 4.5 we present the results for this experiment, from which we draw the following observations:

— *image-to-image*: for this first experiment, the query image is given to the CNN, that converts it to a feature vector. We then search in the feature space for all *image* feature vectors close to the one from the query. For obvious reasons, the closest one is from the same image (with distance = 0), so we start our analysis from the second up.

We can see that all the top 5 retrieved images look similar to the query image, not only in term of colors, shapes, and textures, but also semantically.

Ingredients query
Pizza dough, hummus, arugula, cherry or grape tomatoes, pitted greek olives, crumbled feta cheese.

Cooking instructions query
Cut the dough into two 8-ounce sized pieces. Roll the ends under to create round balls. Then using a well-floured rolling pin, roll the dough out into 12-inch circles. Place the dough circles on sheets of parchment paper. [...]

Top 5 retrieved images

AdaMine
Triplet

Ingredients query
Unsalted butter, eggs, condensed milk, sugar, vanilla extract, chopped pecans, chocolate chips, [...]

Cooking instructions query
Preheat the oven to 375 degrees F. In a large bowl, whisk together the melted butter and eggs until combined. Whisk in the sweetened condensed milk, sugar, vanilla, pecans, chocolate chips, butterscotch chips, and coconut. [...]

Top 5 retrieved images

AdaMine
Triplet

Table 4.4 – **Recipe-to-images visualization, part 2.** For each recipe, we have the top row, indicating the top 5 images retrieved by our AdaMine model for a given recipe query, and the bottom row, indicating the top 5 images by the triplet loss for the same recipe. In cyan, the matching image. In blue, images belonging to the same class than the recipe. In red, images belonging to a different class.

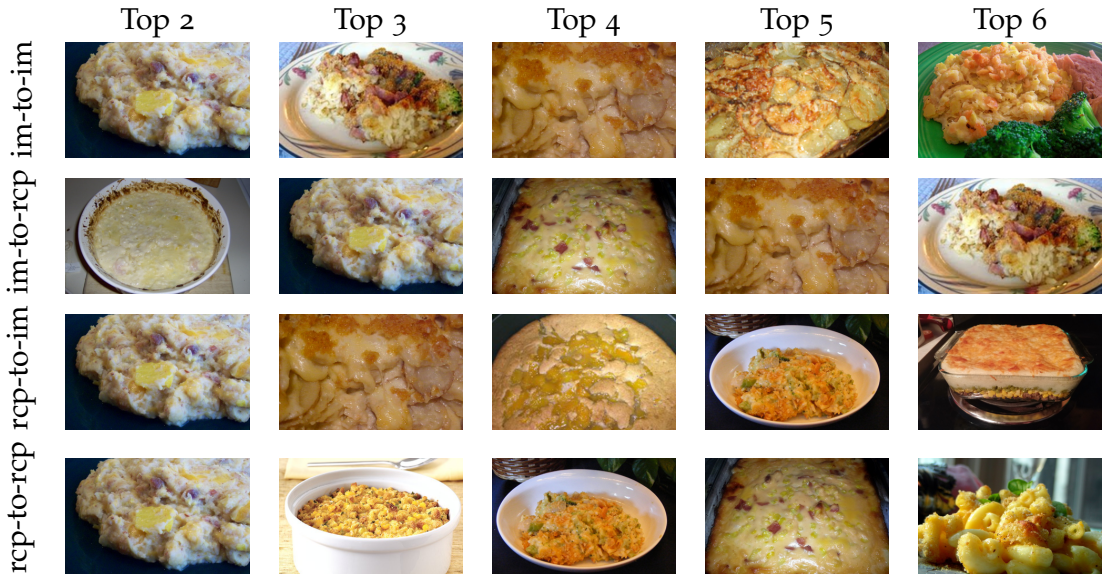


Table 4.5 – **Visualization of our modality-to-modality retrieval tests.** We show the 5 retrieved recipes (the image associated to recipe is displayed) for the multi-modal retrieval tasks.

For instance, the first, third and forth images have grated cheese on top, just like the query image shown in Table 4.3.

Although the goal of this experiment is to analyze the image retrieval with an image query, when we look at the corresponding recipes from the retrieved items, we observe they are all baked, and they also include a similar set of ingredients, with potatoes, milk, butter, cheese, and onion. Small variations in the ingredients are observed, however. For example, the second image has rice instead of potatoes.

— *image-to-recipe*: next, the query image is given to the CNN, that converts it again to a feature vector. We then search in the feature space for all *recipe* feature vectors close to the one from the query. These vectors are extracted using the recipe network, shown in the top-left side of Table 4.1.

From the 5 top retrieved results, 3 are shared with the *image-to-image* search. This is a strong indicator that the embeddings of matching image-recipe pairs are correctly aligned. However, we also obtain results that are less visually similar, but closer in the recipe domain, either in terms of ingredients or in cooking instructions.

— *recipe-to-image*: in the following, the query recipe is given to the recipe network, that converts it to a feature vector. We then search in the feature space for all *image* feature vectors close to the one from the query.

For this case, we also find images of recipes that are similar to the query. Most of them share the same common ingredients as in the *image-to-image* case. We also remark that the images retrieved are visually similar to the picture associated with the query recipe, although no visual information was used for the search.

— *recipe-to-recipe*: for the last of the four experiments, the query recipe is again given to the recipe network, that converts it to a feature vector. We then search in the feature space for all *recipe* feature vectors close to the one from the query.

In Table 4.5, the images are shown merely to provide a visual representation of the recipes, since the query and all the retrieved items are in textual form. Each image we show corresponds to the picture of the recipe retrieved by our model.

Although the recipes also share common ingredients with the query, we observe a richer visual diversity. This is due to the fact the whole search is based upon textual information, and although our model is capable of correctly aligning pictures and recipes, there is still difference between their representations due to the different nature of information contained in each one of them.

4.3.2 Operations with Ingredients

An interesting ability of our model is to map ingredients into the semantic space in order to retrieve recipes containing the same ingredients. This is particularly useful when one would like to know what they can cook using ingredients available in their fridge. The same process should allow us to be able to remove ingredients from recipes, finding alternatives for food allergies, for example. In the following, we explore the potential of these approaches, showing that the semantic organization of the feature space helps with ingredient alignment as well.

Ingredient To Image To demonstrate the first part of this exploration, we create ingredient queries by averaging representation vectors of available ingredients. We then retrieve the nearest neighbors of these queries among 10,000 images randomly picked from the testing set.

In Table 4.6, we showcase images within the top 20 nearest neighbors of the ingredients *carrot* and *mushroom*. We are able to retrieve visually diverse meals containing the query ingredient, even though the number of known ingredients taken into account by the network is big.

Then we propose a second experiment: we first constrain the class of the query to be *pizza*, and then we search for different ingredients inside this class. The results for this experiment are shown in Table 4.7.

Table 4.6 – **Examples for ingredient retrieval.** We show images in the top 20 when searching for the ingredient *Carrot* (top row) or *Mushroom* (bottom row).

Table 4.7 – **Examples for ingredient retrieval inside the pizza class.** We show images in the top 20 results when searching for "mushroom", "pineapple", "olive", "pepperoni", or "strawberry" within the *Pizza* class.

Searching for *pineapple* or *olives* results in different types of pizzas. An interesting remark is that searching for *strawberries* inside the class *pizza* yields images of *fruit pizza* containing strawberries, *i.e.*, images that are visually similar to pizzas while containing the required ingredient. This shows the fine-grained structure of the semantic space where recipes and images are organized by their similarity inside different class clusters at the same time.

Removing ingredients The capacity of finely model the presence or absence of specific ingredients may be interesting for generating menus, specially for users with dietary restrictions, for instance, peanut, lactose or gluten intolerance, as well as vegetarian and vegan diets.

To do so, we randomly select a recipe having *broccoli* in its ingredients list (Table 4.8, first column) and retrieve the top 4 closest images in the embedding space from 1000 recipe images (Table 4.8, top row). Then we remove the *broccoli* in the ingredient list, as well as all of the instructions containing the *broccoli* word. Finally, we once again perform the retrieval task to find the top 4 images associated to this "modified" recipe (Table 4.8, bottom row).

For all of the retrieved images using the original recipe, the broccoli is both visually and textually present, whereas none of the retrieved images using the

Query	Top retrieved images				
Tofu Sauté *Oregano, Zucchini, Tofu, Bell pepper, Onions,* ***Broccoli*** Cut ingredients into small pieces. Boil water [...]					with broccoli
					without broccoli

Table 4.8 – **Retrieving recipes with or without broccoli in the ingredients.** We show the top 4 retrieved images with (top row) and without (bottom row) broccoli in the ingredient and instruction lists for the query.

modified recipe do not contain broccoli. This reinforces our previous statement, highlighting the ability of our semantic space to correctly discriminate items with respect to ingredients, even though this was never taught to the model.

4.4 Conclusion

In this chapter, we explored in-depth the representation space constructed by the AdaMine model presented in Chapter 3. For this task, we adopted the large-scale multi-modal dataset Recipe1M, composed of nearly one million image-recipe pairs and additional class information.

Our apparatus, described in Section 4.2 is composed of two networks. The first one, a Convolutional Neural Network (CNN), is responsible for the image processing, and the second, composed mainly of Long Short-Term Memorys (LSTMs), processes the textual input. The representation spaces of both networks are aligned using AdaMine, that also introduces instance-based and semantic-based structure into them.

In Section 4.3 we show that this strategy is powerful enough to allow the network to learn details down to the ingredient level of a recipe — something that was never explicitly taught to the model. We were able to filter ingredients inside specific classes, as well as to search for specific ingredients on the feature space, retrieving recipes containing them. In one of our experiments, we demonstrate the ability of our model of excluding ingredients from a recipe, and finding alternative recipes not containing that ingredient. This case could be especially useful for applications focused on dietary restrictions and food alternatives.

We have also shown the power of the multi-modal alignment. On Recipe1M, two modalities are available: visual (pictures) and textual (recipes and ingredients). When fully trained, our model was able to perform the cross-modal retrieval task from any modality to any modality, achieving results that vouch for the correctness of the alignment of the feature spaces.

The potential of our learning strategy is not limited to the domain we adopted, as it has the potential to inject information into the representation space and enforce structure to it, as long as there is extra information that can be modeled into a loss function. We have shown that applications of this method for computational cooking are evident, opening interesting perspectives for ambitious tasks as menu composition or cooking with restricted ingredient availability.

Chapter 5

CONCLUSION

In this thesis, we discussed representation spaces created by Deep Artificial Neural Networks (ANNs). We were particularly interested in the transfer of knowledge between deep nets. In the transfer context, these spaces are usually created by training a deep net on a large-scale dataset. Because of the richness in these datasets, the representations created by the network can more easily generalize to other tasks.

We started by studying precision and redundancy properties inherent to these spaces. These studies led us to the exploration of ways to compress the representations extracted from deep nets and, in light of the excellent results we obtained, we also studied compression schemes for the whole network.

On a second part of this manuscript, we focused on refining representation spaces with a fine-tuning strategy. This led us to adopt metric learning strategies for introducing information directly into multi-modal spaces. The method we proposed, AdaMine, achieves a 5-fold improvement over the state-of-the-art for the Recipe1M dataset.

We then provided a deeper analysis of the representation created by our model, showing how our method was able to semantically organize information inside the representation space, opening the way for many multi-task applications in the real world.

In the following, we highlight our contributions, as well as future work emerging from this work.

5.1 Main contributions

Study of deep representation spaces and compression schemes

First, we focused on exploring feature spaces in the context of transfer learning for the image classification task. We have shown that deep architectures have interesting properties of redundancy across dimensions, as well as unnecessary high precision in their representations. We have proposed in Chapter 2 an experimental protocol we call *stress framework* for evaluating these properties.

Our exploratory studies evaluated two strategies for reducing their dimensionality: (1) randomly and (2) PCA-based; as well as three strategies for quantizing their features: (1) erasing part of their binary representation, (2) using a global

dictionary-based approximation, and (3) using a feature-wise dictionary-based approximation. We also evaluated the impacts of these strategies when performed on different depths of the network, corroborating existing observations by the scientific community that the up-to-last layer is generally the best for transfer learning approaches.

This exploration revealed that deep representation spaces learned on the ImageNet classification tasks have excessive richness, and can therefore be compressed. Our feature compression experiment was able to reduce the representations we tested by 98.4% while keeping 99.1% of their original score on the classification task of the PASCAL Visual Object Classes (PASCAL VOC) 2007. This simple scheme serves as showcase, and the adoption of better dimensionality reduction or quantization methods have the potential to improve even further these results.

Next, we presented ways of reducing the number of parameters of Artificial Neural Networks (ANNs) while also copying their feature spaces. One of our toy experiments served to show that Convolutional Neural Networks (CNNs) can be compressed in a teacher-student setup. We were able to create a student architecture with roughly 6.8% of the number of parameters in a teacher network, while keeping 95.6% of its score. Then, a proposal of a family of relaxed regression losses for mimic learning was made. These losses are inspired by the results we obtained with our exploration of feature spaces. A simple experiment showed their potential, and further tuning of parameters could improve their performance, especially for transfer tasks.

Multi-modal alignment strategy AdaMine

Following, we refined feature spaces in Chapter 3. We tackle a multi-modal retrieval task, for which the goal is to find images based on textual descriptions, or texts based on images. We adopt a Convolutional Neural Network (CNN) for the image part, and Long Short-Term Memorys (LSTMs) for the textual part. The CNN is pre-trained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification task, and most of the Long Short-Term Memorys (LSTMs) are trained from scratch. However, we remove the classification layer for these networks, and our learning scheme directly acts upon their representation space.

Our proposal, AdaMine, is composed of two multi-modal and multi-task triplet losses: (1) instance-based and (2) semantic-based; as well as an adaptive sampling strategy capable of providing stability to the weighting of these two losses. AdaMine follows a different path than the one usually followed by classic multi-task approaches: instead of adding heads specialized to each task to the model, it directly enforces the task into the feature space through task-specific losses.

We tested our method in a cross-modal retrieval task, showing its effectiveness when compared to state-of-the-art approaches in both of the tested tasks: retrieval of image based on text and retrieval of text based on image. We then performed extensive ablation studies to analyze the contribution of each compo-

nent of AdaMine, concluding that incorporating semantic information into the representation space has a positive impact on the scores, but when combined with our adaptive sampling strategy, the benefits are even more promising.

Multi-task properties of semantic latent spaces

Finally, we showed that feature spaces constructed with AdaMine can learn semantic subtasks never seen during training (Chapter 4). We focus on cooking-related activities, and in particular recipe retrieval from dish pictures, and dish picture retrieval from recipes. This kind of task has recently emerged as a cross between Machine Learning (ML) and smart cooking, and previous research verified its difficulty, as well as potential solutions [1].

We achieved a 5-fold improvement over the state-of-the-art for the Recipe1M dataset, composed of nearly 1 million image-recipe pairs. Then, we proceeded to explore the representation space refined with AdaMine. A first remark is that the semantic information is well encoded into these spaces, that are correctly organized in clusters of different classes.

In the sequence, we explored different ways of performing the retrieval task in this multi-modal space, not only performing recipe-to-image and image-to-recipe, but also image-to-image and recipe-to-recipe. This experiment gave us clues that the representations found inside these spaces can discriminate between specific ingredients, and not only general classes.

With this finding, we went on to test if we were able to expand the initial task the model was trained for: instead of querying with images or recipes, we chose to search for a single ingredient, and found out that we were able to retrieve recipes containing it. We were also able to search for specific ingredients inside a chosen class (pineapple in pizzas, for example).

Finally, we tested if our model was able to *exclude* ingredients from a recipe. This kind of scenario is particularly useful for menu composition, finding alternatives to allergens, and many other real-world applications. We were able to remove ingredients from recipes, showing that alternative recipes without it were retrieved.

5.2 Future directions

More compression strategies

Now that we have demonstrated that deep features can be redundant, more advanced dimensionality reduction and quantization strategies could be tested to find compression schemes that have the potential to be even more powerful. Important clues were found for signature optimization, in particular for embedded systems or in the context of mobile classification / retrieval.

1. see http://visiir.lip6.fr/ for a demonstration

Study of the relaxed regression losses

With the preliminary results obtained in Chapter 2, many questions related to mimic learning remain open. A broader study of the relaxed retrieval losses we proposed could be conducted under the transfer perspective, as our findings indicate that smaller tasks benefit the most from the compressibility of deep features.

Extended versions of AdaMine

The principles behind AdaMine's strategy can be generalized to other datasets and tasks. Although there are not many of them containing instance-based and semantic-based information, the adaptive mining strategy remains valid for any task, and replacing multiple heads by a multi-task loss can create powerful models for different applications. In our case, the retrieval task and the classification task were combined, but as long as we can model a task as a loss function, it can be injected directly into the feature space.

On a long-term basis, more ways of enforcing particular structures into deep representation spaces can be studied. In particular, it would be interesting to test the applicability of these methods for data generation, fraud detection, and more generalized forms of descriptors.

These extensions can be particularly useful in the cooking context, improving the retrieval performance for subtasks like the ingredient exclusion or inclusion, as well as the basic retrieval for simple ingredients. For example, the neighborhood constraints we have studied and exploited for creating AdaMine could be extended to incorporate more information into the representation space, as the ingredients of the recipe and different tastes — sweetness, sourness, saltiness, bitterness, and savoriness.

BIBLIOGRAPHY

Andrew, Galen, Raman Arora, Jeff Bilmes, and Karen Livescu (2013). "Deep canonical correlation analysis". In: *ICML*, pp. 1247–1255 (cit. on pp. 40, 41).

Antol, Stanislaw, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh (2015). "VQA: Visual Question Answering". In: *International Conference on Computer Vision (ICCV)* (cit. on p. 39).

Arandjelović, R., P. Gronat, A. Torii, T. Pajdla, and J. Sivic (2016). "NetVLAD: CNN architecture for weakly supervised place recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition* (cit. on p. 42).

Audebert, Nicolas, Bertrand Le Saux, and Sébastien Lefèvre (2018). "Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 140. Geospatial Computer Vision, pp. 20–32. URL: http://www.sciencedirect.com/science/article/pii/S0924271617301818 (cit. on p. 40).

Avila, S., N. Thome, M. Cord, E. Valle, and A. De A. Araújo (2013). "Pooling in Image Representation: The Visual Codeword Point of View". In: *Computer Vision and Image Understanding (CVIU)* 117.5, pp. 453–465 (cit. on pp. 14, 19).

Azizpour, H., A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson (Sept. 2016). "Factors of Transferability for a Generic ConvNet Representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.9, pp. 1790–1802 (cit. on p. 12).

Bach, Francis R and Michael I Jordan (2002). "Kernel independent component analysis". In: *Journal of machine learning research* 3.Jul, pp. 1–48 (cit. on p. 41).

Beijbom, O., N. Joshi, D. Morris, S. Saponas, and S. Khullar (2015). "Menu-Match: Restaurant-Specific Food Logging from Images". In: *2015 IEEE Winter Conference on Applications of Computer Vision* (cit. on p. 62).

Bossard, Lukas, Matthieu Guillaumin, and Luc Van Gool (2014). "Food-101 – Mining Discriminative Components with Random Forests". In: *ECCV* (cit. on p. 62).

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1994). "Signature Verification using a "Siamese" Time Delay Neural Network". In: *Advances in Neural Information Processing Systems 6*. Morgan-Kaufmann, pp. 737–744. URL: http://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf (cit. on pp. 31, 43).

Bruna, J. and S. Mallat (2013). "Invariant Scattering Convolution Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 35.8, pp. 1872–1886 (cit. on p. 10).

Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil (2006). "Model Compression". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: ACM, pp. 535–541. URL: http://doi.acm.org/10.1145/1150402.1150464 (cit. on p. 31).

Carvalho, Micael, Rémi Cadène, David Picard, Laure Soulier, and Matthieu Cord (2018). "Images & Recipes: Retrieval in the cooking context". In: *IEEE International Conference on Data Engineering (ICDE), Data Engineering meets Intelligent Food and Cooking Recipe (DECOR) workshop* (cit. on pp. 8, 61).

Carvalho, Micael, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord (2018). "Cross-Modal Retrieval in the Cooking Context: Learning Semantic Text-Image Embeddings". In: *The ACM conference on Research and Development in Information Retrieval (SIGIR)* (cit. on pp. 8, 37).

Carvalho, Micael, Matthieu Cord, Sandra Avila, Nicolas Thome, and Eduardo Valle (2016). "Deep Neural Networks Under Stress". In: *IEEE International Conference on Image Processing (ICIP)* (cit. on pp. 8, 9, 29).

CEA2017: Proceedings of the 9th Workshop on Multimedia for Cooking and Eating Activities in Conjunction with The 2017 International Joint Conference on Artificial Intelligence (2017) (cit. on p. 62).

Chatfield, K., K. Simonyan, A. Vedaldi, and A. Zisserman (2014). "Return of the Devil in the Details: Delving Deep into Convolutional Nets". In: *British Machine Vision Conference (BMVC)* (cit. on pp. 10, 14, 18, 19).

Chen, Jingjing and Chong-Wah Ngo (2016). "Deep-based ingredient recognition for cooking recipe retrieval". In: *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, pp. 32–41 (cit. on p. 52).

Chen, Jingjing and Chong-wah Ngo (2016). "Deep-based Ingredient Recognition for Cooking Recipe Retrieval". In: *MM*, pp. 32–41 (cit. on p. 62).

Chen, M., K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang (2009). "PFID: Pittsburgh fast-food image dataset". In: *ICIP* (cit. on p. 62).

Chen, Weihua, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang (2017). "Beyond triplet loss: a deep quadruplet network for person re-identification". In: *CVPR* (cit. on p. 43).

Cheng, Y., D. Wang, P. Zhou, and T. Zhang (Jan. 2018). "Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges". In: *IEEE Signal Processing Magazine* 35.1, pp. 126–136 (cit. on p. 31).

Chevalier, M., N. Thome, M. Cord, J. Fournier, G. Henaff, and E. Dusch (Sept. 2015). "LR-CNN for fine-grained classification with varying resolution". In: *2015 IEEE International Conference on Image Processing (ICIP)*. URL: http://dx.doi.org/10.1109/ICIP.2015.7351374 (cit. on p. 12).

Courbariaux, M., Y. Bengio, and J.-P. David (2015). "BinaryConnect: Training Deep Neural Networks with binary weights during propagations". In: *Advances in Neural Information Processing Systems (NIPS)* (cit. on p. 11).

Courbariaux, M., Y. Bengio, and J.-P. David (2015). "Training deep neural networks with low precision multiplications". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 11).

Courbariaux, Matthieu, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio (Feb. 2016). "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1". In: arXiv: 1602.02830. URL: http://arxiv.org/pdf/1602.02830v3 (cit. on p. 30).

Dechter, Rina (1986). "Learning While Searching in Constraint-satisfaction-problems". In: *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*. AAAI'86. Philadelphia, Pennsylvania: AAAI Press, pp. 178–183. URL: http://dl.acm.org/citation.cfm?id=2887770.2887799 (cit. on p. 1).

Durand, Thibaut, Nicolas Thome, and Matthieu Cord (2015). "MANTRA: Minimum Maximum Latent Structural SVM for Image Classification and Ranking". In: *International Conference on Computer Vision (ICCV)* (cit. on p. 12).

Durand, Thibaut, Nicolas Thome, and Matthieu Cord (2016). "WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks". In: *Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 10).

El Mahdaouy, Abdelkader, Saïd Ouatik El Alaoui, and Eric Gaussier (Mar. 2018). "Improving Arabic information retrieval using word embedding similarities". In: *International Journal of Speech Technology* 21.1, pp. 121–136. URL: https://doi.org/10.1007/s10772-018-9492-y (cit. on p. 40).

Everingham, M., L. Van Gool, C. Williams, J. Winn, and A. Zisserman (2010). "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision (IJCV)* 88.2, pp. 303–338 (cit. on pp. 10, 19).

Faghri, Fartash, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler (2017). "VSE++: Improved Visual-Semantic Embeddings". In: *arXiv preprint arXiv:1707.05612* (cit. on pp. 50, 55, 59, 60).

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin (2008). "LIBLINEAR: A library for large linear classification". In: *Journal of Machine Learning Research (JMLR)* 9, pp. 1871–187 (cit. on p. 18).

Farinella, Giovanni Maria, Dario Allegra, and Filippo Stanco (2015). "A Benchmark Dataset to Study the Representation of Food Images". In: *ECCV*. Ed. by Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, pp. 584–599 (cit. on p. 62).

Hadsell, R., S. Chopra, and Y. LeCun (2006). "Dimensionality Reduction by Learning an Invariant Mapping". In: *CVP)*, pp. 1735–1742 (cit. on p. 41).

Han, Song, Huizi Mao, and William J. Dally (Oct. 2015). "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and

Huffman Coding". In: arXiv: 1510.00149. URL: http://arxiv.org/pdf/1510.00149v5 (cit. on p. 30).

Harris, Zellig (1954). "Distributional structure". In: *Word* 10.23, pp. 146–162 (cit. on p. 62).

He, K., X. Zhang, S. Ren, and J. Sun (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385. URL: http://arxiv.org/abs/1512.03385 (cit. on p. 10).

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (Mar. 2015). "Distilling the Knowledge in a Neural Network". In: eprint: 1503.02531 (cit. on p. 31).

Hoffer, Elad and Nir Ailon (2015). "Deep Metric Learning Using Triplet Network". In: *ICLR* (cit. on p. 43).

Hotelling, Harold (1936). "Relations between two sets of variates". In: *Biometrika* 28.3/4, pp. 321–377 (cit. on pp. 39, 41).

Hu, J., J. Lu, and Y. P. Tan (2014). "Discriminative Deep Metric Learning for Face Verification in the Wild". In: *CVPR*, pp. 1875–1882 (cit. on p. 48).

Huang, X. and Y. Peng (2017). "Cross-modal deep metric learning with multi-task regularization". In: *ICME*, pp. 943–948 (cit. on p. 43).

Jégou, Hervé, Matthijs Douze, Cordelia Schmid, and Patrick Pérez (June 2010). "Aggregating local descriptors into a compact image representation". In: *CVPR 2010 - 23rd IEEE Conference on Computer Vision & Pattern Recognition*. San Francisco, United States, pp. 3304–3311 (cit. on p. 42).

Joulin, Armand, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache (Nov. 2015). "Learning Visual Features from Large Weakly Supervised Data". In: arXiv: 1511.02251. URL: http://arxiv.org/pdf/1511.02251v1 (cit. on p. 40).

Judd, P., J. Albericio, T. Hetherington, T. Aamodt, N. Jerger, R. Urtasun, and A. Moshovos (2015). "Reduced-Precision Strategies for Bounded Memory in Deep Neural Nets". In: *CoRR* abs/1511.05236. URL: http://arxiv.org/abs/1511.05236 (cit. on pp. 11, 30).

Karpathy, Andrej and Li Fei-Fei (2015). "Deep visual-semantic alignments for generating image descriptions". In: *CVPR*, pp. 3128–3137 (cit. on pp. 39, 41, 50, 55).

Kawano, Yoshiyuki and Keiji Yanai (2014). "FoodCam: A Real-Time Mobile Food Recognition System Employing Fisher Vector". In: *MMM* (cit. on p. 62).

Kim, Minje and Paris Smaragdis (Jan. 2016). "Bitwise Neural Networks". In: arXiv: 1601.06071. URL: http://arxiv.org/pdf/1601.06071v1 (cit. on p. 30).

Kim, Yong-Deok, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin (Nov. 2015). "Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications". In: arXiv: 1511.06530. URL: http://arxiv.org/pdf/1511.06530v2 (cit. on p. 30).

Kingma, Diederik and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 34, 52).

Kiros, Ryan, Ruslan Salakhutdinov, and Richard S Zemel (2015). "Unifying visual-semantic embeddings with multimodal neural language models". In: *TACL* (cit. on pp. 40, 41, 50, 60).

Kiros, Ryan, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). "Skip-Thought Vectors". In: *NIPS*, pp. 3294–3302 (cit. on pp. 39, 52, 55).

Kokkinos, Iasonas (July 2017). "Ubernet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 46, 51).

Krizhevsky, A., I. Sutskever, and G. Hinton (2012). "ImageNet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1–9 (cit. on pp. 1, 10).

Krizhevsky, Alex (2009). *Learning Multiple Layers of Features from Tiny Images*. Tech. rep., pp. 1–60 (cit. on p. 33).

Kusmierczyk, Tomasz, Christoph Trattner, and Kjetil Nørvåg (2016). "Understanding and predicting online food recipe production patterns". In: *HT*, pp. 243–248 (cit. on p. 52).

Lai, Pei Ling and Colin Fyfe (2000). "Kernel and nonlinear canonical correlation analysis". In: *International Journal of Neural Systems* 10.05, pp. 365–377 (cit. on p. 41).

Law, Marc T, Nicolas Thome, and Matthieu Cord (2013). "Quadruplet-wise image similarity learning". In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 249–256 (cit. on pp. 40, 43).

Le Barz, C., N. Thome, M. Cord, S. Herbin, and M. Sanfourche (Sept. 2015). "Exemplar based metric learning for robust visual localization". In: *2015 IEEE International Conference on Image Processing (ICIP)*. URL: http://dx.doi.org/10.1109/ICIP.2015.7351626 (cit. on p. 35).

LeCun, Y., Y. Bengio, and G. Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444. URL: http://www.nature.com/doifinder/10.1038/nature14539 (cit. on p. 10).

LeCun, Y, L Bottou, Y Bengio, and P Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on pp. 4, 33).

LeCun, Yann, Corinna Cortes, and Christopher Burges (n.d.). *The MNIST database*. URL: http://yann.lecun.com/exdb/mnist/ (cit. on p. 30).

Li, Li-Jia, Hao Su, Yongwhan Lim, and Li Fei-Fei (Sept. 2013). "Object Bank: An Object-Level Image Representation for High-Level Visual Recognition". In: *Int J Comput Vis* 107.1, pp. 20–39. URL: http://dx.doi.org/10.1007/s11263-013-0660-x (cit. on p. 3).

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar (2017). "Focal Loss for Dense Object Detection". In: *ICCV* (cit. on p. 51).

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov, pp. 2579–2605 (cit. on p. 67).

Maji, Subhransu, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi (2013). "Fine-Grained Visual Classification of Aircraft". In: *Technical Report* (cit. on p. 12).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). "Distributed representations of words and phrases and their compositionality". In: *NIPS*, pp. 3111–3119 (cit. on pp. 52, 62).

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 p. 12).

Quattoni, A. and A. Torralba (2009). "Recognizing indoor scenes". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 413–420 (cit. on p. 19).

Razavian, A., H. Azizpour, J. Sullivan, and S. Carlsson (2014). "CNN Features off-the-shelf : an Astounding Baseline for Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv: 1403.6382 (cit. on pp. 12–14).

Rippel, Oren, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev (Nov. 2015). "Metric Learning with Adaptive Density Discrimination". In: arXiv: 1511.05939 (cit. on p. 43).

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: *IJCV* 115.3, pp. 211–252 (cit. on p. 1).

Salvador, Amaia, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba (Mar. 2017). "Learning Cross-modal Embeddings for Cooking Recipes and Food Images". In: *CVPR*. CVF / IEEE. Honolulu, Hawaii, USA: CVF / IEEE. URL: http://im2recipe.csail.mit.edu/im2recipe.html (cit. on pp. 40, 41, 46, 48, 49, 51–54, 56–59, 63, 64, 66).

Sanjo, Satoshi and Marie Katsurai (2017). "Recipe Popularity Prediction with Deep Visual-Semantic Fusion". In: *CIKM*, pp. 2279–2282 (cit. on p. 62).

Sohn, Kihyuk (2016). "Improved Deep Metric Learning with Multi-class N-pair Loss Objective". In: *NIPS*, pp. 1857–1865 (cit. on pp. 43, 46).

Song, Hyun Oh, Yu Xiang, Stefanie Jegelka, and Silvio Savarese (Nov. 2015). "Deep Metric Learning via Lifted Structured Feature Embedding". In: arXiv: 1511.06452. URL: http://arxiv.org/pdf/1511.06452v1 (cit. on p. 43).

Song, Hyun Oh, Yu Xiang, Stefanie Jegelka, and Silvio Savarese (2016). "Deep Metric Learning via Lifted Structured Feature Embedding". In: *CVPR* (cit. on pp. 44, 45, 51).

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). "Going Deeper with Convolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9 (cit. on pp. 14, 19).

Tran, T. Q. N., H. L. Borgne, and M. Crucianu (June 2016). "Aggregating Image and Text Quantized Correlated Components". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2046–2054 (cit. on p. 39).

Ustinova, Evgeniya and Victor Lempitsky (2016). "Learning Deep Embeddings with Histogram Loss". In: *NIPS*, pp. 4170–4178 (cit. on p. 43).

Vanhoucke, V., A. Senior, and M. Mao (2011). "Improving the speed of neural networks on CPUs". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1–8. URL: http://research.google.com/pubs/archive/37631.pdf (cit. on p. 11).

Vanni, Laurent, M Ducoffe, D Mayaffre, F. Precioso, D Longrée, V Elango, N. N. Santos, J. Gonzalez, L Galdo, and C Aguilar (July 2018). "Text Deconvolution Saliency (TDS) : a deep tool box for linguistic analysis". In: *56th Annual Meeting of the Association for Computational Linguistics*. Melbourne, France. URL: https://hal.archives-ouvertes.fr/hal-01804310 (cit. on p. 10).

Vedaldi, A. and K. Lenc (2015). "MatConvNet – Convolutional Neural Networks for MATLAB". In: *ACM International Conference on Multimedia (MM)* (cit. on p. 19).

Wang, L., Y. Li, and S. Lazebnik (2016). "Learning Deep Structure-Preserving Image-Text Embeddings". In: *CVPR*, pp. 5005–5013 (cit. on pp. 43, 44).

Wang, Xin, D. Kumar, N. Thome, M. Cord, and F. Precioso (2015). "Recipe recognition with large multimodal food dataset". In: *ICMEW*, pp. 1–6 (cit. on pp. 19, 62).

Weinberger, Kilian Q. and Lawrence K. Saul (June 2009). "Distance Metric Learning for Large Margin Nearest Neighbor Classification". In: *J. Mach. Learn. Res.* 10, pp. 207–244. URL: http://dl.acm.org/citation.cfm?id=1577069.1577078 (cit. on pp. 40, 41, 49).

Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang (May 2016). "A survey of transfer learning". In: *Journal of Big Data* 3.1, p. 9. URL: https://doi.org/10.1186/s40537-016-0043-6 (cit. on p. 12).

Xing, Eric P., Michael I. Jordan, Stuart J Russell, and Andrew Y. Ng (2003). "Distance Metric Learning with Application to Clustering with Side-Information". In: *NIPS*, pp. 521–528 (cit. on pp. 40, 41).

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio (2015). "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning* (cit. on p. 39).

Yan, Fei and Krystian Mikolajczyk (2015). "Deep correlation for matching images and text". In: *CVPR*, pp. 3441–3450 (cit. on pp. 40, 41).

Yao, B. and L. Fei-Fei (June 2010). "Grouplet: A structured image representation for recognizing human and object interactions". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 9–16 (cit. on p. 12).

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson (2014). "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 3320–3328 (cit. on pp. 12–14).

ACRONYMS

ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ML	Machine Learning
AI	Artificial Intelligence
CV	Computer Vision
GPU	Graphics Processing Unit
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DL	Deep Learning
SVM	Support Vector Machine
PASCAL VOC	PASCAL Visual Object Classes
BoW	Bag-of-Words
mAP	mean Average Precision
ACC	Accuracy
LSTM	Long Short-Term Memory
SOTA	State-Of-The-Art