Transfert de connaissance pour la compréhension des images

Praveen Kulkarni

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Knowledge Transfer for Image Understanding

Présentée et soutenue par
Praveen Kulkarni

Thèse soutenue publiquement le 23/01/2017 devant le jury composé de		
M. STEPHANE CANU	Professeur des Universités, INST NAT SC APPLIQ ROUEN	Examinateur
M. MATTHIEU CORD	Professeur des Universités, UNIVERSITÉ PARIS 6 PIERRE ET MARIE CURIE	Rapporteur
M. PATRICK PÉRÉZ	Directeur de Recherche INRIA, HDR, UNIVERSITÉ RENNES 1	Directeur de thèse
M. FRÉDÉRIC JURIE	Professeur des Universités, UNIVERSITÉ CAEN NORMANDIE	Directeur de thèse
M. JAKOB VERBEEK	Chargé de Recherche, INRIA GRENOBLE	Rapporteur
M. JOAQUIN ZEPEDA	Ingénieur de Recherche, RF TECHNICOLOR CESSON-SEVIGNE	Examinateur

Thèse dirigée par M. FRÉDÉRIC JURIE, GREYC-UMR6072 et M. PATRICK PÉRÉZ, INRIA

Logo Ecole Doctorale

UNIVERSITÉ CAEN NORMANDIE

MI/IΣ

Logo laboratoire

GREYC

Contents

List of figures

List of Tables

0.1 Acknowledgement

Firstly, I express my sincere gratitude to the team for their motivation, guidance and the immense knowledge imparted during these three years. It is shared between the Université de Caen Basse-Normandie (UNICAEN) and Technicolor Rennes. A big thank you to Prof. Frédéric Jurie, my thesis director, who has been always there to support and guide during the course of three years. I would like to Thank Prof. Frédéric Jurie for his precious timely advice for moving the research in the direction of Convolutional Neural Network. I strongly want to thank Dr. Joaquin Zepeda because of him I climbed this uphill task of PhD effortlessly. I really thank again Dr. Joaquin Zepeda for all his effort to improve my writing skills. I would like to thank Dr. Patrick Pérez and Louis Chevallier for their timely feedback and suggestions at various points of time. I was at ease to schedule a meeting with Prof. Frédéric Jurie, Dr. Joaquin Zepeda, Dr. Patrick Pérez and Louis Chevallier to get a thoughtful inputs/ideas to propel the PhD work in the right direction. Great thanks to Louis Chevallier who proposed the thesis problem which eventually led to my recruitment in Technicolor and allowed me to be part of this wonderful team of reserchers. I would like to thank Technicolor management and ANRT (Association Nationale Recherche Technologie) for giving me this environment which every researcher dreams of, and extending financial support to conduct this thesis, attend conferences, training, tutorials and summer school. I want to thank my Rapporteurs Prof. Matthieu Cord, Prof. Jakob Verbeek and member of the jury Stéphane Canu who were willing to turn their attention to my works. A big thanks to Prof. Gaurav Sharma, currently an Asst. Prof. in IIT Kanpur, for his valuable suggestion and motivation during the period of internship at Technicolor.

I mainly like to thank my colleague and my friend Saurabh Puri for his valuable suggestions on improving my writing skills, for being tennis mate and for all our technical/non-technical discussions. I extend my big thanks to my friends Saurabh Puri, Combodge Bist, Himalaya Jain and Mohini Ahuja for all those treats, parties, functions and making this place an homely experience. I would like to also thank all my friends and colleagues in Technicolor. I am finally ending with a big thank you to my parents, my brother, my wife (Prachi Kulkarni) and our newborn little daughter (Pranika Kulkarni). You all have always supported me and it's thanks to you that I arrived so far.

0.2 Résumé

Le Transfert de Connaissance (Knowledge Transfer or Transfer Learning) est une solution prometteuse au difficile problème de l'apprentissage des réseaux profonds au moyen de bases d'apprentissage de petite taille, en présence d'une grande variabilité visuelle intra-classe. Dans ce travail, nous reprenons ce paradigme, dans le but d'étendre les capacités des CNN les plus récents au problème de la classification.

Dans un premier temps, nous proposons plusieurs techniques permettant, lors de l'apprentissage et de la prédiction, une réduction des ressources nécessaires - une limitation connue des CNN.

(i) En utilisant une méthode hybride combinant des techniques classiques comme des Bag-Of-Words (BoW) avec des CNN.

(ii) En introduisant une nouvelle méthode d'agrégation intégrée à une structure de type CNN ainsi qu'un modèle non-linéaire s'appuyant sur des parties de l'image. La contribution clé est, finalement, une technique capable d'isoler les régions des images utiles pour une représentation locale.

De plus, nous proposons une méthode nouvelle pour apprendre une représentation structurée des coefficients des réseaux de neurones.

Nous présentons des résultats sur des jeux de données difficiles, ainsi que des comparaisons avec des méthodes concurrentes récentes. Nous prouvons que les méthodes proposées s'étendent à d'autres tâches de reconnaissance visuelles comme la classification d'objets, de scènes ou d'actions.

Mot-clés: Vision par Ordinateur, Apprentissage Machine, Classification d'Images, Transfer de Connaissances, Modèles à Parties.

Discipline: Informatique et applications.

Laboratoire: GREYC CNRS UMR 6072, Sciences 3, Campus 2, Bd Marechal Juin, UniversitÂt'e de Caen, 14032 Caen.

0.3 Summary in English

Knowledge transfer is a promising solution for the difficult problem of training deep convolutional neural nets (CNNs) using only small size training datasets with a high intra-class visual variability. In this thesis work, we explore this paradigm to extend the ability of state-of-the-art CNNs for image classification.

First, we propose several effective techniques to reduce the training and test-time computational burden associated to CNNs:

(i) Using a hybrid method to combine conventional, unsupervised aggregators such as Bag-of-Words (BoW) with CNNs;

(ii) Introducing a novel pooling methods within a CNN framework along with non-linear part-based models. The key contribution lies in a technique able to discover useful regions per image involved in the pooling of local representations;

In addition, we also propose a novel method to learn the structure of weights in deep neural networks. Experiments are run on challenging datasets with comparisons against state-of-the-art methods. The methods proposed are shown to generalize to different visual recognition tasks, such as object, scene or action classification.

Keywords: Computer Vision, Machine Learning, Image Classification, Transfer Learning, Part-Based Models.

Chapter 1

General Introduction

" I am writing first chapter and reminding myself that I'm shoveling sand into box so that later I can build castles. "

Shannon Hale

Contents

1.1 Context

Automatic image classification is a most researched topic in computer vision. Starting from the year 2012 Convolutional Neural Networks (CNNs) emerged as the most successful technique for this task, as well as a number of other computer vision tasks. However to train millions of parameters in CNN one requires a huge amount of annotated data. This requirement poses a significant challenge if the available training data is limited for a target task at hand. To address this challenge, in the recent literature, researchers proposed various ways to apply a technique called Knowledge Transfer (also known as Transfer Learning) to transfer the knowledge gained by training CNNs parameters on some large annotated dataset to the target task with limited availability of training data. Most of our work in this thesis was dedicated to propose novel methods to classify images in benchmark target datasets using pre-trained CNN and Knowledge Transfer technique.

We start by giving few important applications of image classification systems. Next, we discuss challenges posed by classification tasks. In Section 1.1.3 to Section 1.1.5 we present historical developments starting from traditional techniques till CNN era that were used to address classification challenges. Then, in Section 1.1.6 we discuss how in recent literatures, one utilizes Knowledge Transfer to enhance the ability of pre-trained CNN on target tasks. Finally, we present the objectives of this work and briefly list the contributions made in this thesis.

1.1.1 Image classification and applications

Image classification aims to detect whether or not a specific visual concept is present in an image. For example, it detects whether the image contains a *car*, *bus*, *person*, *bottle* or *motorbike*. Note here that classification is different from detection task. In detection task, one roughly locates the object of interest in an image with a bounding box, in contrast to image-level labeling in classification.

Image classification systems have application in many real world scenarios. The popularity of social media or photo sharing websites such as Instagram, Google Plus, Ipernity and Flickr have led to millions of images being uploaded by users on a daily basis. Consider an application of image classification systems [8, 21, 112, 97, 71] implemented in Google Plus where one can search through a large pool of private photos stored in their personal computer using Google Search. When a user enables the option of synchronizing Google Plus with private albums in his personal computer, the user can retrieve the relevant photos automatically based on his query in text format launched from Google Search. For example, assume the user is interested in viewing the images containing 'beachside' scenery from his private collections. Image classification can be used to perform such task by just taking the user query in text form 'beachside' and retrieving the relevant images from his private collections.

Consider an application of image classification in landscape assessment or planning [68]. For landscape assessment one needs to classify landscape images into classes such as forest, water or agriculture. Since the number of landscape images in a database might be very large, it becomes difficult for a user to mine the required relevant images manually from a database for assessment. In such cases, we need an automated image classification system which can perform the task of retrieving the relevant images, based on the user query.

One more application is in the supermarket and grocery store [36], where the su-

permarket assistant is serving the customers for pricing the items list. Here the image classification system can be used to identify the items automatically, based on the visual content in an image and then prize them accordingly.

One more application is in the context of color grading in movie post production [130] where user is interested in scene/category specific color grading of video clips. Color grading is a process of enhancing the cinematic look to various video clips from entire movie. For example, consider a use case of landscape (*e.g.*, mountain, water bodies, vegetation, *etc.*) specific color grading. Here user can use image classification techniques [48] to pre-process the entire movie and automatically retrieve the video clips belonging to 'vegetation' category, and then he can apply color grading that is suited for the 'vegetation' category.

One more use case is in automatically selecting the appropriate movies for children based on their violent contents. De Souza *et al.* [17] proposed to employ image classification techniques to analyze the key frames from entire movie and subsequently classify it into 'violent' or 'non-violent' category. However, note that this can generalize to audience specific recommendation of movies.

Calibrating the camera parameters (*e.g.*, focus distance, shutter speed, *etc.*) based on class of current scenic view that one is capturing can be posed as image classification task [3]. Camera can analyze automatically the current scene that one is capturing and classify it into categories such as *indoor scene*, *outdoor scene*, *mountains*, *etc.* Once class of given scene that is being captured is known, camera can calibrate itself based on the associated pre-defined calibration parameters defined per class.

1.1.2 Challenges

To devise computer algorithms to classify images in visual world like we do is a challenging task. Instances of same class categories have a diverse set of appearances due to variations in object scale, occlusion, deformations, illumination conditions, viewpoint, articulated pose and background clutter. Fig. 1.1 depicts different instances of same class *cat* in various images. One more important challenge to address is intra-class appearance variability. For *e.g.*, two distinct *cars* or two distinct *persons* simply have different appearance (see Fig. 1.2). Conversely, there is inter-class confusion in multi-class context. For *e.g.*, images containing *cat* and *tiger* might be difficult to distinguish (see Fig. 1.3).

Besides addressing above challenges, algorithms have to be efficient in speed and scalable to large datasets. Consider a practical application of these algorithms in social media website, wherein nearly billions of photos and videos are uploaded by users on a daily basis. Highly efficient algorithms are necessary to organize, retrieve, infer and generalize to growing sets of images with time.

1.1.3 Engineered image representation

A image classification system consists of two main components - Image representation and Classifier. In the first component, one must design 'representation per image' in such a way that images belonging to same class must be similar, even under large intra-class variations and dissimilar to images from different classes. The second component *i.e.* classifier defines decision boundary in the space where images are represented. There has been significant work to design complex high dimensional image representations so as to yield good recognition performance even with simple linear classifiers.

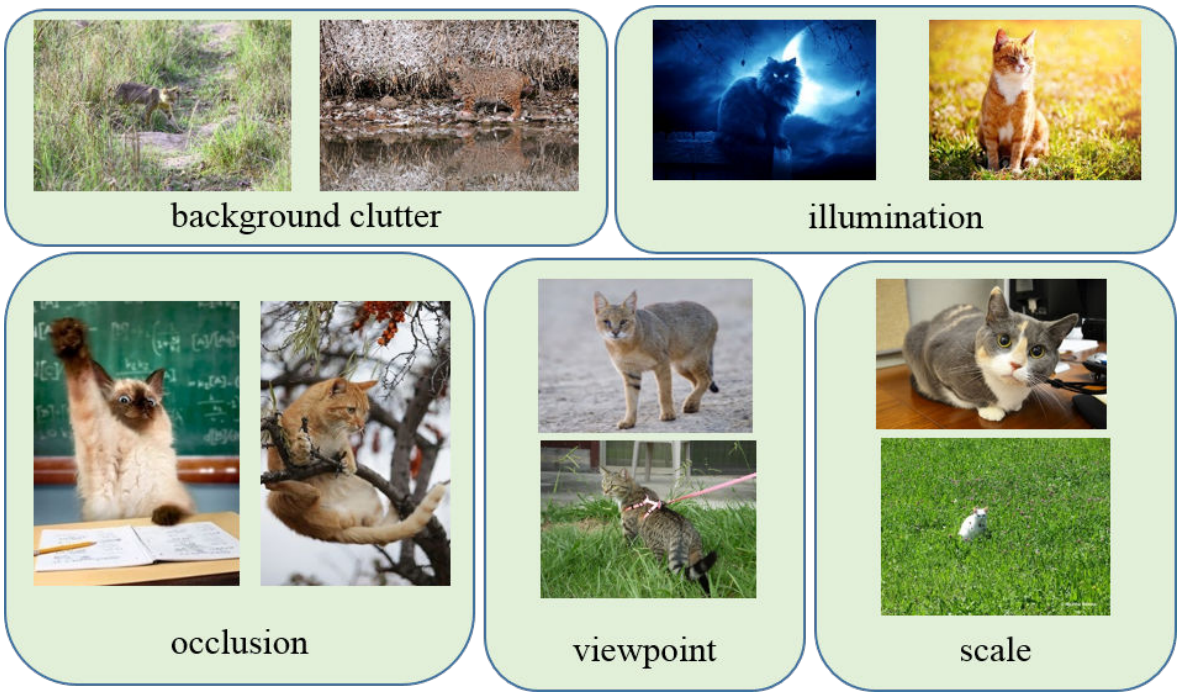


Figure 1.1 – Images containing *cat* class with diverse set of variations.

Figure 1.2 – Intra-class appearance difference between two images belonging to same *car* class (left) and *person* class (right).

Figure 1.3 – Two examples to illustrate the inter-class similarity between images containing *tiger* and *cat* class.

Image can be represented either by global or local features. Global features represent whole image into compact vector which mainly captures shape (*e.g.*, Zhang *et al.*[135]), color (*e.g.*, Swain *et al.*[114]) or texture (*e.g.*, Ro *et al.*[92]). Global features are fast but they do not cope well with occlusion, strong viewpoint variations or deformable objects.

There has been tremendous progress in designing invariant **local descriptors** with little increase in computational complexity compared to global features. The local

descriptors extracted per image at multiple scales and locations tend to be robust to variations in scale and rotation. Some of state-of-the art descriptors are Scale Invariant Feature Transform (SIFT) [75], Speeded-up Robust Features (SURF) [2] and Histogram of Oriented Gradients (HOG) [16]. In an image these descriptors are either extracted at key points (at corners or blobs) or at regularly spaced cells. Last decade saw an emergence of feature encoding strategies which makes it possible to aggregate these local descriptors into a single vectorial representation. Subsequently, this representation can be used to train classifier, e.g. Support Vector Machine (SVM) [14].

Feature encoding method BoW (Bag-of-Words) proposed by Sivic *et al.*[108] is inspired by text retrieval system. In this method a set of visual codewords is constructed and saved in a database by vector quantizing a pool of local descriptors extracted from images in training data. BoW then maps local descriptors extracted per image into a single vector in 2 steps (1) encoding - hard assignment of each local descriptor to the closest visual codeword, thus turning it into a visual code. (2) pooling - visual codes are then aggregated (sum-pooling) into a single vector (image representation). Significant amount of work has been conducted in the literature so far to improve both encoding step of a feature encoding method. For example, Van *et al.*[122] proposed, at encoding step to reduce quantization loss because of hard quantization in BoW by soft quantizing across codewords.

Further improvement in recognition accuracy is obtained by aggregation methods such as Fisher Vector (FV) [88], Super Vector (SV) [138] and VLAD [50]. These methods attempt to reduce the information loss caused by soft/hard quantization. This is done by pooling the weighted difference between local descriptors and each visual codeword. For example, [88] proposed to build visual codewords by employing generative model i.e, fitting (using Expectation maximization) a Gaussian Mixture Model (GMM) to the pool of local descriptors. FV then maps local descriptors extracted per image into a single vector in 2 steps (1) encoding - Fisher score [49] per local descriptor is computed by obtaining the gradient of the log-likelihood with respect to parameters in the GMM (2) pooling step wherein Fisher scores are aggregated into final image representation.

Till now we have described representation scheme using SIFT/HOG as local descriptors. These descriptors typically capture patterns such as edges, corners and color contrast in an image. The introduction of **Part-Based Models (PBMs)** by [107, 63, 22, 53, 31] showed how the notion of "part" can be inculcated into image classification framework. These methods model visual concepts present in an image using their constituent parts instead of edge/corner like features. PBMs has following 3 appealing characteristics:

1. Some "Parts" are distinctive for a class. For example, an image patch containing "head of horse" is distinctive of "horse" class compared to any other class.

2. Presence of at-least one visible discriminative part can classify image efficiently. Thus PBMs provides invariance to strong occlusion.

3. Compact and expressive final image representation since "parts" are shared across class categories. For example, wheels occur in car, aeroplane, bus and bicycles.

PBMs attempts to train part classifiers by discovering, in training data, a set of discriminative regions analogous to visual codewords discussed previously. Next, these trained part classifiers are used in combination with BoW or FV aggregators to map a set of region proposals per image into final part-based image representation.

1.1.4 Classifiers

After successful representation of images in training data using any of above described methods, one can train classifiers such as linear SVM on top of those hand-crafted representations. In this thesis we resort to multi-class classification problems by training one-vs-rest binary classifiers.

1.1.5 End to End learning - CNN

Recently, big gains in classification results have been attributed to state-of-the-art Convolutional Neural Networks (CNNs) Krizhevsky *et al.*[59]. CNN can be described as an end-to-end learning scheme wherein stacked layers (in network) extract the complex patterns in an image in a hierarchical fashion and learns the parameters in these layers jointly with final classifiers. Thus, non-linear feature extraction techniques from an image are directly optimized for a classification task as opposed to traditional methods wherein image representation and classifiers are designed independently. CNNs are usually trained in a supervised manner. CNNs contain huge number of parameters which require a huge amount of training data, time and hardware resources. Training these CNNs has been possible due to the arrival of new large datasets such as ImageNet [20] and Places [137].

1.1.6 Knowledge Transfer (KT)

KT is a broad technique which has shown tremendous progress in classification, regression and clustering problems [84]. We are interested in particular type of KT called as *inductive transfer learning* [84]. It is applicable to supervised learning problem wherein lot of training data is available in source dataset whereas target dataset is very small. One example of KT is Orabona et al.[81] wherein they adapt pre-trained classifiers trained on source dataset to target dataset. Other methods [58, 29], attempt to capture differences in statistical distribution between source and target datasets.

The CNN model easily overfits to training data in target task due to limited annotated samples. Hence, to train CNN weights on ImageNet source dataset (usually the objective is a classification task) and use KT technique to adapt to target task at hand. Note that target task might contain statistically different images (*e.g.*, `Pascal-VOC-2007` [27], `MIT-Indoor-67` [89] and `Willow` [18] datasets) compared to source dataset or objective of task might be completely different (*e.g.*, segmentation, detection, style transfer). Methods [35, 44, 10, 101], are some early attempts wherein subset of pre-trained network layers are fine-tuned to new datasets. Alternatively, the method in [79], showed how one can append few additional adaptation layers to existing structure of network and train only them on new datasets.

1.2 Objectives of this thesis

This thesis aims to improve image classification systems. This subject has been in the focus of attention for many researchers in the computer vision community and is visible from prior works (CNN, BoW, FV and PBMs) cited in previous section. The common objective of the algorithms proposed in this thesis is to employ Knowledge Transfer (KT) to improve recognition accuracy on target dataset while being computationally efficient. In our work we have attempted to address several important questions in

order to improve recognition accuracy on target datasets: How to aggregate the intermediate descriptors resulting from multiple layers in pre-trained CNNs? What is the best deep architecture for current datasets? What is the best fine tuning/adaptation strategy of pre-trained CNNs? What is the best aggregation/pooling strategy in deep architectures? We validate and compare our proposed methods to other state-of-art prior works using publicly available challenging datasets.

1.3 Contributions

The concrete list of thesis contributions are summarized here. We briefly describe here the problems addressed in chapters 3 to 8.

Chapter 3 Zero-shot learning of the visual classifiers has gained momentum in the cases where the number of scene/object categories to distinguish is huge. This is because of unavailability/difficulty in collecting annotated training data for all classes. The On-the-fly Classification system (OTF) by Chatfield *et al.*[8] attempts to address this problem by launching text queries on an image search engine, *e.g.* Google Image, and using retrieved images as training data. In chapter 2 we try to improve the OTF by leveraging attribute classifiers trained for generic attributes such as *long legs*, *hooves*, *paws*, *claws* in the case of animal categories, and propose a novel method to combine these generic attribute classifiers with final visual classifiers to improve classification performance.

Chapter 4 We attempt to combine state-of-the-art CNN method with traditional encoding techniques such as BoW and FV. CNNs are resource hungry in terms of computation and size of training data. Owing to a small dataset in the target task, we show how one can improve classification accuracy by combining cheap (less resource hungry) BoW and FV aggregators with powerful pre-trained CNNs.

Chapter 5 We use a pre-trained CNN to represent images in training data. We design a model which can adapt these extracted features to the new target dataset jointly with visual classifiers governed by a max-margin loss function. In this chapter, we obtain good results even with size of image representation as small as 20.

Chapter 6 We propose a novel method to learn the structure of layers in deep networks during the optimization process. Structure in this context means the size of the weight matrix inside each fully-connected layer comprising deep architecture. We learn the structure by inserting diagonally constrained matrices between the fully connected layers and regularizing the elements in them by L1 norm. The L1 based regularization establish a trade off between classification loss and network size.

Chapter 7 We propose a new pooling strategy to aggregate the local features extracted per image - the so-called pooling step in image classification. We construct the PBMs wherein part classifiers are jointly trained with final visual classifiers. Further, these PBMs operate on local regions described by powerful state-of-the-art CNN. We showed in this work how PBMs can be seen as an effective knowledge transfer technique. We showed that our method gives state-of-the-art results on standard publicly available challenging datasets.

Chapter 8 We propose an extension to our work in chapter 7. The method described in chapter 7 is computationally extensive since one has to compute CNN activations multiple times to describe a huge number of local regions per image (generated by chosen object proposal scheme). In this chapter, unlike in chapter 7, we exploits the benefits of dense proposal scheme [37] to propose local regions per image. In addition, our method operates on multiple scales per input image and we introduced the idea of per-part latent scale selection.

1.4 Flow of thesis

In this section we give the outline of the thesis. Note that all the terminologies, especially, acronyms and notations used in the rest of thesis are explained at the place of their usage.

In Chapter 2, we discuss some of the concepts which are pertinent for image classification methods and discuss their relative advantages. We discuss in detail concepts such as supervised image classification, image representation using: SIFT and CNN. We also discuss coding and pooling concepts that are employed in recent literature. Note that these are also concepts which we have used in our thesis work.

In chapters 3 to 8 we discuss in detail our contributions. Each chapter starts by introducing the context followed by related prior works. Then, we describe the underlining approach. We finally conclude by accessing the proposed approach against publicly available challenging datasets.

In chapter 9 we first summarize all the proposed methods and possible future works. We then draw conclusions about this thesis work.

Chapter 2

Review of the related work

Contents

In this chapter, we present a review of literature work in image classification. We focus on important concepts and corresponding prior art relevant to this thesis work. We start by presenting the basic setup for supervised image classification. Next, in Section 2.2, we delve into an important component *i.e.* image representation.

The problem of image representation has been the important topic of research for several years and it is also the core topic of our thesis. We start by discussing traditional image representation using low-level descriptors (SIFT) in Section 2.2.1. We then discuss various *aggregation* mechanisms used in order to map the local descriptors into a fixed size image representation. Then we discuss state-of-the-art feature extractor: Convolutional Neural Network (CNN) in Section 2.3. In Section 2.4 we discuss methods proposed to mine discriminative regions per image. Then, in Section 2.5 we discuss some details on classifiers such as linear Support Vector Machines (SVM). Finally, in Section 2.6 we present the datasets used in this thesis work.

2.1 Basic setup of supervised image classification

The supervised classification is based on the idea that one is provided with annotated training data. Classification task uses this annotated data to learn a model and empower it to classify unseen/unannotated images in a test data. We present here a specific case of classification task *i.e.* a binary classification problem. In the binary case, the task is to classify an unseen image sample into two groups (positive or negative) on the basis of a decision rule.

The training phase in the classification task is achieved by following three simple steps, as illustrated in Fig. 2.1 (top). The first step in the process is to split the images in training data into positive and negative samples based on ground truth annotations. The second step is extracting a fixed-dimensional representation per image. The third step is learning a classifier by minimizing the classification error. Note here that training data can contain images belonging each to only one class (For instance, a given image can contain either *dog* or *sofa* class) or to more than one class simultaneously. For instance, a given image can contain both *dog* and *sofa* classes. During the testing phase (see Fig. 2.1 (bottom)), the learned classifiers are then used to classify unseen test images. Note that the learned classifiers are applied on the fixed-dimensional representation extracted from the test image. The performance of the whole system is judged based on its ability to correctly classify the images in test data.

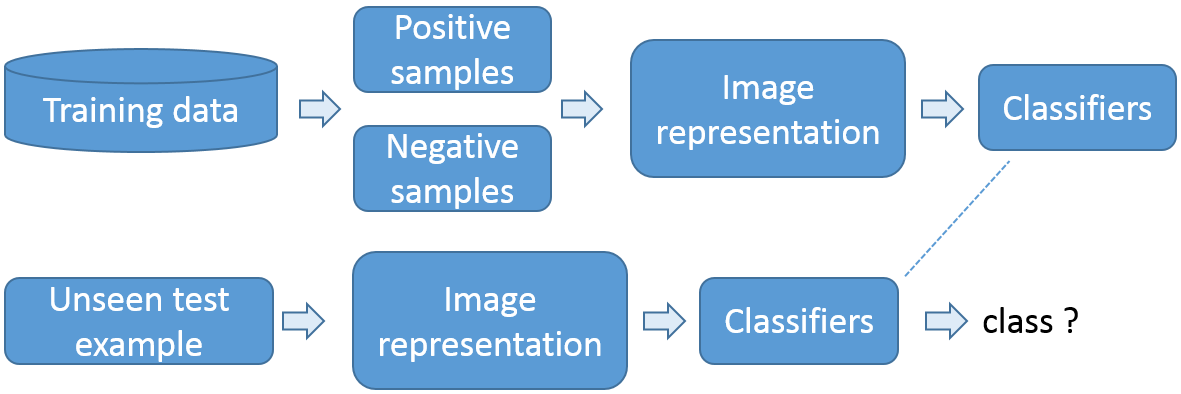


Figure 2.1 – Block diagram to illustrate the pipeline followed during the training (top) and testing phase (bottom) of a supervised binary image classification task.

2.2 Traditional image representation

The aim of the representation step is to encode the content of an image as a fixed-dimensional real-valued vector. This is often approached by first, extracting local descriptors at multiple locations in an image and second, aggregating the local descriptors into the fixed-dimensional representation. We discuss here a well known local descriptor: Scale Invariant Feature Transform (SIFT) and their aggregation technique.

2.2.1 Low-level descriptors (SIFT)

SIFT (Scale Invariant Feature Transform) descriptor is an ubiquitous tool in computer vision literature. It attempts to capture discriminative low-level features in an image like straight edges, sharp corners, curved edges and color *etc.*For example, straight edges and sharp corners like patterns are discriminative features for images containing man-made objects such as *car*, *bus* and *building*. On another hand, curved patterns are discriminative features for natural images (*e.g.*, *beaches*, *mountains*, *trees*, *etc.*) or images containing animals (*e.g.*, *dog*, *cat*, *etc.*).

SIFT is applied to an image patch, usually provided by interest point detector operating at wide range of scales. For each interest point location detected at position (x, y), a rectangular block of size 16×16 is considered with its center at (x, y). The block is then partitioned into 4×4 cells. Orientation histogram is computed per cell by quantizing the gradient directions into 8 orientation bins. The resulting descriptor has $16 \cdot 8 = 128$ entries and is invariant to scale, rotation, viewpoint and contrast (achieved by appropriate normalization).

Fig. 2.2 illustrates two ways to select local regions within an image for SIFT computation. One can compute SIFT at key-point locations or in a dense manner. In this thesis, we have used dense-SIFT [72]. Some extension to SIFT is PCA-SIFT [55] which uses PCA (Principal Component Analysis) for dimensionality reduction.

2.2.2 Aggregators

The above-described method can be used to extract a variable number of local descriptors per image. Next, we must devise an aggregation method which can combine these local descriptors into a global image representation. The classifier is then trained on

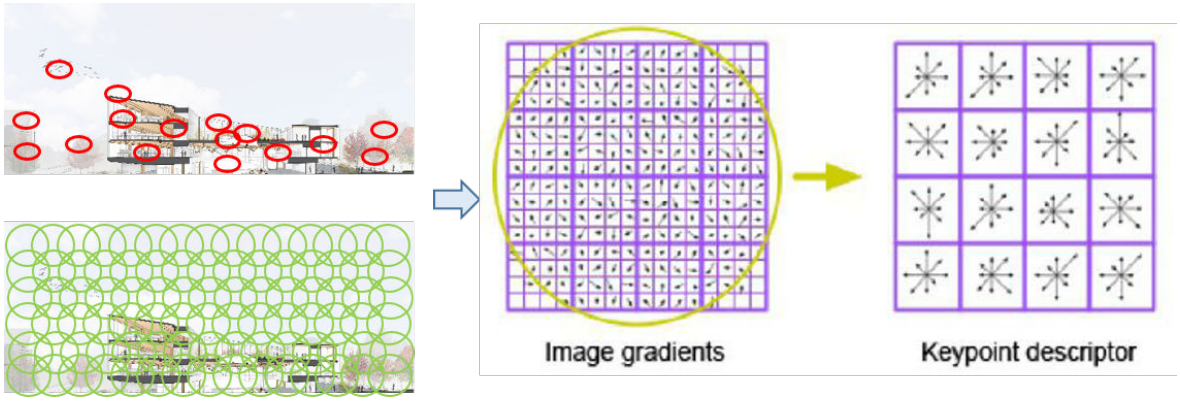


Figure 2.2 – On the right is SIFT by [75]. SIFT is computed either at a specific point (left top) or at dense locations (left bottom).

top of these representations. Most popularly used aggregators along with the SIFT local descriptors are Bag-of-Words (BoW) [108] and Fisher Vector (FV) [88]. Generally, these methods employ three steps:

Construct visual codewords: Visual codewords are summarized representation of space spanned by the local descriptors obtained from training data. In practice, the visual codewords are constructed off-line and saved in a database. For example, in BoW method, visual codewords are obtained by vector-quantizing SIFT descriptors extracted from all/subset of the images in training data. Quantization is done by using the k-means algorithm. Alternatively, one can use approximate nearest neighbor [78] to speed up the process of quantization.

Many works has been proposed to replace unsupervised k-means clustering algorithm used in BoW. Yang *et al.* [132] proposed to use sparse coding algorithm to obtain visual codewords. One step further [54] learned the codewords in a discriminative manner. FV [88] on other hand rely on generative method, achieved by representing visual codewords by means of a Gaussian Mixture Model (GMM). The GMM is better because it models the clustering of local descriptors using an advanced statistics such as mean and covariance compared to simple count statistics used in BoW. Note here that the GMM is learned using maximum likelihood estimation.

Coding: The local descriptors per image are embedded onto visual code words to obtain visual codes. Each visual code per local descriptor is a vector containing either binary (in the case of BoW [108]) or a real-valued vector (in the case of FV also known as Fisher score Jaakkola *et al.*[49]). These visual codes have appealing properties such as compactness or sparseness.

Pooling: The visual codes associated with local descriptors are aggregated using different pooling mechanisms. Some of the standard pooling mechanisms are *sum* and *max* pooling. Boureau *et al.*[7] have done a theoretical analysis of different pooling mechanisms. One important conclusion from their work is that *max* pooling yields better recognition accuracy. Boureau *et al.*[7] also showed how the pooling step induces desirable properties such as translational invariance into a final representation. But, one can argue that pooling tends to ignore spatial relationships among the local descriptors.

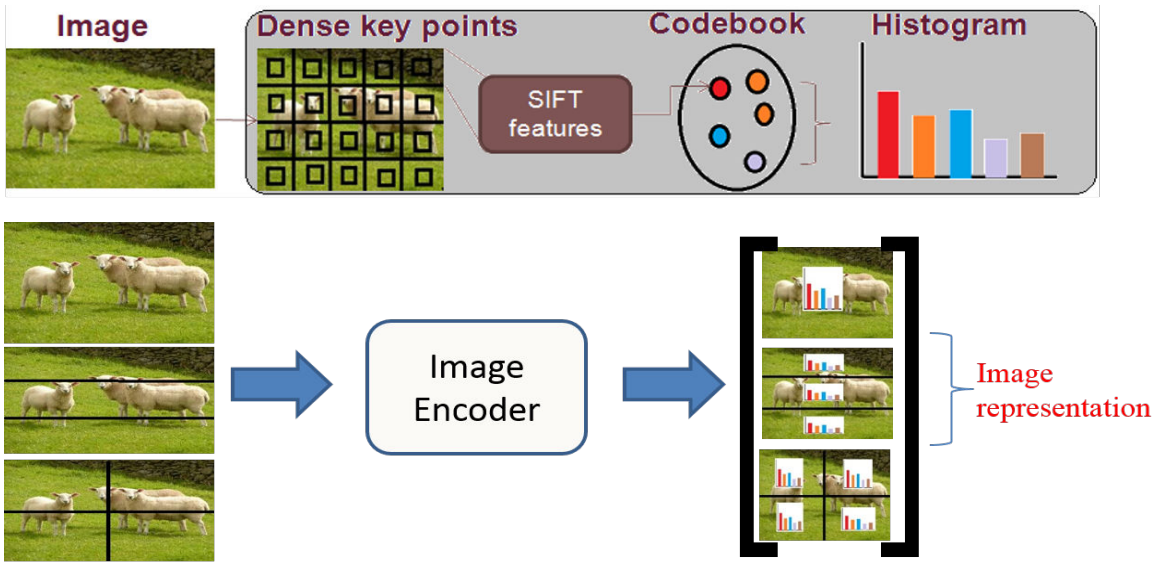


Figure 2.3 – Block diagram of the classic image encoder using BoW (top). Images are divided into a multi-scale rectangular grid of size 1x1, 3x1, and 2x2 (bottom left). Histograms computed per spatial cell are stacked to get the final image representation (bottom right).

As an extension to the pooling mechanism, Lazebnik *et al.*[64] proposed Spatial Pyramid Pooling (SPM), to enhance recognition accuracy by taking into account geometric relationships among the local descriptors. The SPM embeds weak geometrical constraints into the image representation by dividing the image into a multi-scale rectangular grid (see Fig. 2.3 for the graphical illustration of SPM employed in BoW). In Fig. 2.3, the BoW histograms are computed per spatial cell and subsequently stacked to get a global image representation (see Fig. 2.3).

The description of images by BoW and subsequently training non-linear classifiers have shown significant improvement in the classification performance [119, 136]. The size of image representations are usually very high dimensional, thus approaches [119, 136] does not scale well with increasing training data. For the visual codebook size K, the image representation size in case of BoW is K and the dimension of FV is $(2D+1)K$, where D is the size of each local descriptor (*e.g.*, $D = 128$ for SIFT descriptor). The value of K is usually in the order of thousands and scales rapidly with the size of training data. Hence, due to huge size of image representation, particularly in case of FV, linear classifiers are preferred option as opposed to the non-linear classifiers. The linear classifiers scales linearly with the number of training examples and are also easy to learn using popular Stochastic Gradient Descent (SGD) [94] technique.

2.3 Convolutional Neural Networks (CNNs)

CNNs have steadily replaced traditional hand-crafted representations, beating them by a large margin in recognition accuracy. CNN extracts feature in a hierarchical fashion. Zeiler *et al.*[134] have shown that CNNs in their initial layers contain Gabor-like filters which capture edges or corners whereas their intermediate layers capture high-level features and their final layers tend to capture even more complex structures. This hierarchical/deep aspect of the CNN enhances its ability to represent any complex pattern in an input signal.

In the case of CNN [59] one can view the intermediate convolutional filters as equivalent to visual codewords. In CNN, codewords (also known as kernels) are learned on-the-fly *i.e.* during the training process. CNN enjoys an inherent advantage over FV and BoW because of the codewords, in CNN, are directly optimized by minimizing classification loss. Each kernel operates on the previous layer's output, to obtain a feature map (analogous to visual code). The resulting visual codes are then pooled to obtain a global image representation. In the context of CNN, max-pooling is the preferred option over average pooling and it drastically reduces the number of parameters in the optimization process.

In practice, CNNs are pre-trained using the huge ImageNet dataset (source dataset). Razavian *et al.* [101] showed that a global descriptor extracted per image from the intermediate layers in the pre-trained CNN generalizes even to other datasets on which it is not trained. However, methods [40, 13] observed that by extracting CNN activations from multiple regions per image, then discovering discriminative regions, and aggregating them further improves the recognition accuracy on target dataset. This is because pre-trained CNNs fail to select discriminative regions within its intermediate layers when applied globally on images in the target dataset.

2.4 Discovering discriminative regions

The pursuit for discriminative regions has been the topic of interest in many recent works [22, 95, 69, 32, 26, 107, 30, 6]. These works referred discriminative regions by many names such as mid-level features, discriminative or distinctive visual elements/parts/regions. These are richer features compared to low-level descriptors. They tend to capture human understandable high-level concepts. For example *wheel*, *door*, *windows*, *head*, *leg*, *head*, *etc.* are some of the discriminative regions present in an image. Fig. 2.4 displays some of the mid-level concepts belonging to different classes.

One can model each class by a set of discriminative and representative regions. They are discriminative because they are found in images belonging to a particular class. The representativeness comes from the fact that they frequently occur in many images belonging to the class of interest. This kind of property is not well established in the traditional techniques that use low-level SIFT descriptors. Thus the mid-level features captures appearance distribution much better than the low-level descriptors.

One can see that discovering discriminative regions requires extensive human annotations for informative regions per image in a training data. Thus it is desirable to discover discriminative regions using only image-level annotations. Methods [22, 95, 69, 32, 26, 107, 30, 6] have proposed various solutions to discover discriminative regions using limited supervision *i.e.* using only image-level annotations. Yet, selection of the good candidate regions is far from being solved. This is because one cannot learn the appearance models before knowing discriminative regions and vice versa. Thus making problem highly non-convex and initialization critical.

Figure 2.4 – Discriminative regions relevant to different classes. The manually cropped regions from an entire image are used to illustrate the discriminative regions.

2.5 Linear classifiers

We have already discussed the advantage of linear classifiers over non-linear classifiers in Section 2.2.2. The linear classifiers are scalable to large quantities of training data. The goal of the classification task in linear case is to find a hyperplane in order to separate samples belonging to different classes. The Support Vector Machine (SVM) [14] is the most popular supervised machine learning algorithm which constructs the hyperplane in a high-dimensional space, where input training samples lie.

The generalization ability is an indicator of merit of any classification algorithm, *i.e.* how well it classifies images in unannotated datasets. The generalization capability of SVMs stems from the principle of maximum margin. To achieve this, a linear SVM computes the separating hyperplane that is farthest from closest training samples belonging to either class (see visualization in Fig. 2.5, illustrated for the two-dimensional case).

SVM classifiers operate on an image $\mathbf{I}$ represented by $\mathbf{f}(\mathbf{I})$ using anyone of the above mentioned (previous section) representation techniques and predicts the class by $\langle \mathbf{f}(\mathbf{I}), \mathbf{w} \rangle$ (dot product). The parameter $\mathbf{w}$ is learned by minimizing an objective function governed by two terms: 1) Loss term (For example hinge loss) and 2) Regularization term ($L1$ or $L2$ regularization). The SVM maintains a trade off between loss and regularization term by setting a penalty weight on regularization term. In practice, the penalty weight is set using proper cross-validation strategy (*i.e.* using held out training samples). In addition, [88] have shown that it is beneficial to $L2$ normalize the global image representation $\mathbf{f}(\mathbf{I})$ before learning SVM.

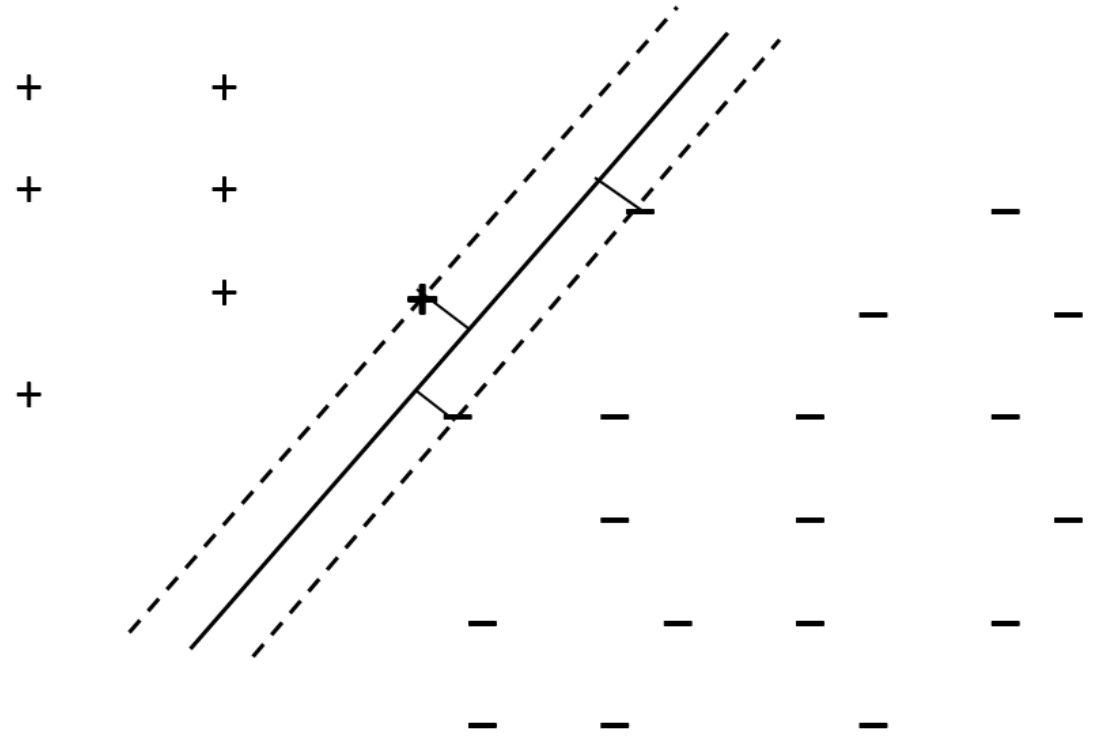


Figure 2.5 – Hyperplane separating the two classes with maximum margin.

2.6 Dataset used in this thesis

We are using popular publicly available datasets to conduct and validate our proposed methods. Each dataset is split into training, validation and test sets. Ground truth data is available for the complete test dataset.

`Pascal-VOC-2007.` The `Pascal-VOC-2007` dataset [27] is an image classification dataset consisting of 9963 images of 20 different visual object classes divided into 5011 training images (of which 2510 are validation images) and 4952 testing images. The images contain natural scenes and the visual object classes span a wide range, including animals (*e.g.*, dog, cat), vehicles (*e.g.*, aeroplane, car) and other manufactured objects (*e.g.*, tv monitor, chair). We can observe from Fig. 2.6 that visual concepts are embedded in complex scenes with various conditions: scale, lighting, occlusion and pose.

`MIT-Indoor-67.` As opposed to objects, scenes are non-localized visual concepts and might even be characterized by the presence or absence of several objects. The `MIT-Indoor-67` [89] dataset is a scene recognition dataset consisting of 67 indoor scenes (*e.g.*, nursery, movie theater, casino or meeting room) each represented by 80 training images and 20 test images. We use 20 randomly chosen training images from each class as a validation set. Fig. 2.7 illustrates some example images spanning six categories.

`Willow.` Recognizing human action in photos is a challenging task due the absence of temporal information. Dedicated to this task, the `Willow` dataset [18] consists of 7 action categories such as “play instrument”, “walk” or “ride horse” spread across 490

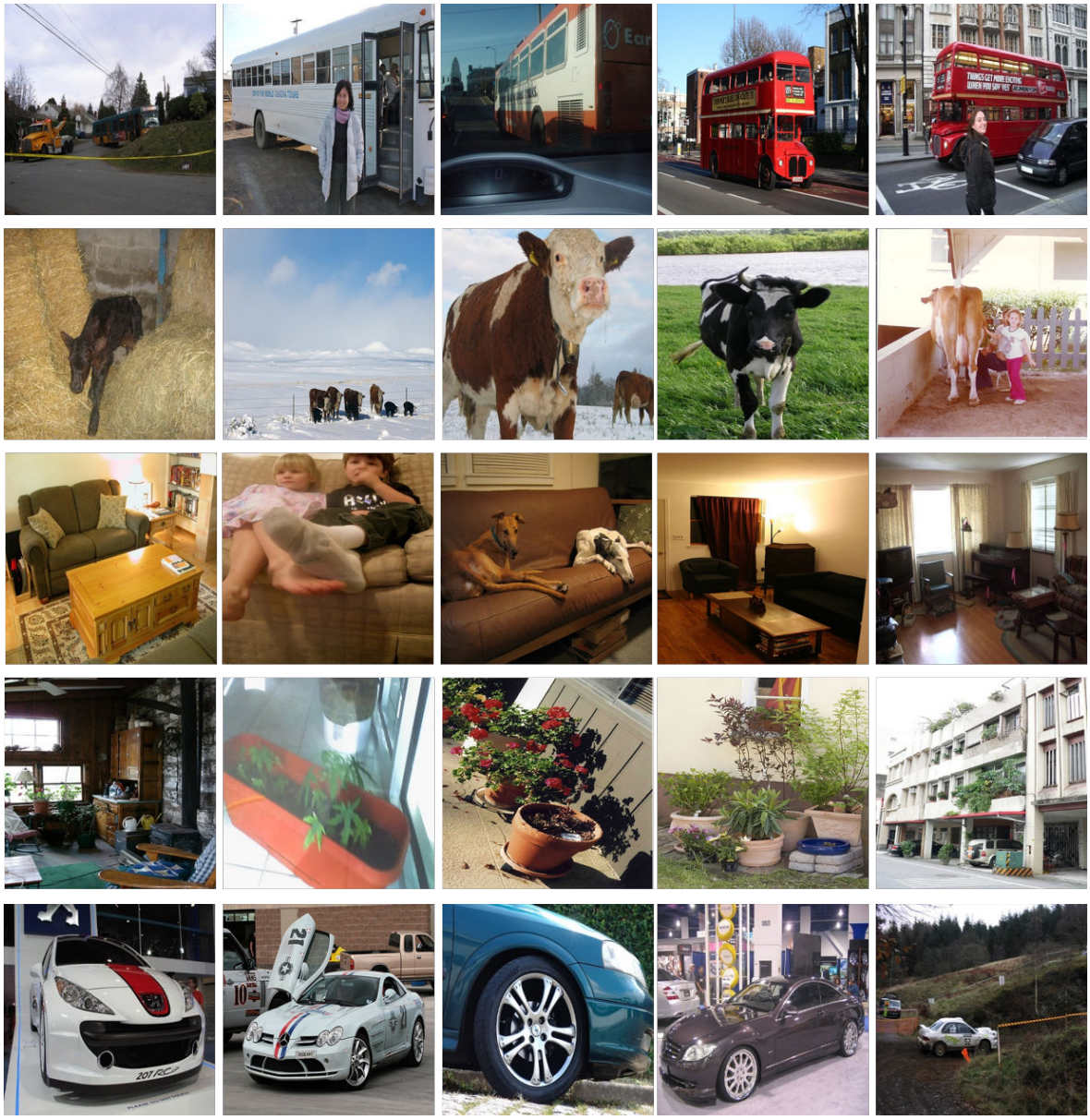


Figure 2.6 – Example images in `Pascal-VOC-2007` test dataset: each row from top to bottom belongs to classes: *bus, cow, sofa, potted plant, car*. Note that images are stretched for convenience to display.

Figure 2.7 – Example images in `MIT-Indoor-67` test dataset: each row from top to bottom belongs to classes: *airport inside, restaurant, hospital room, grocery store, inside bus, corridor*.

training images (of which 210 are validation images) and 421 test images. Fig. 2.8 illustrates some example images spanning seven categories.

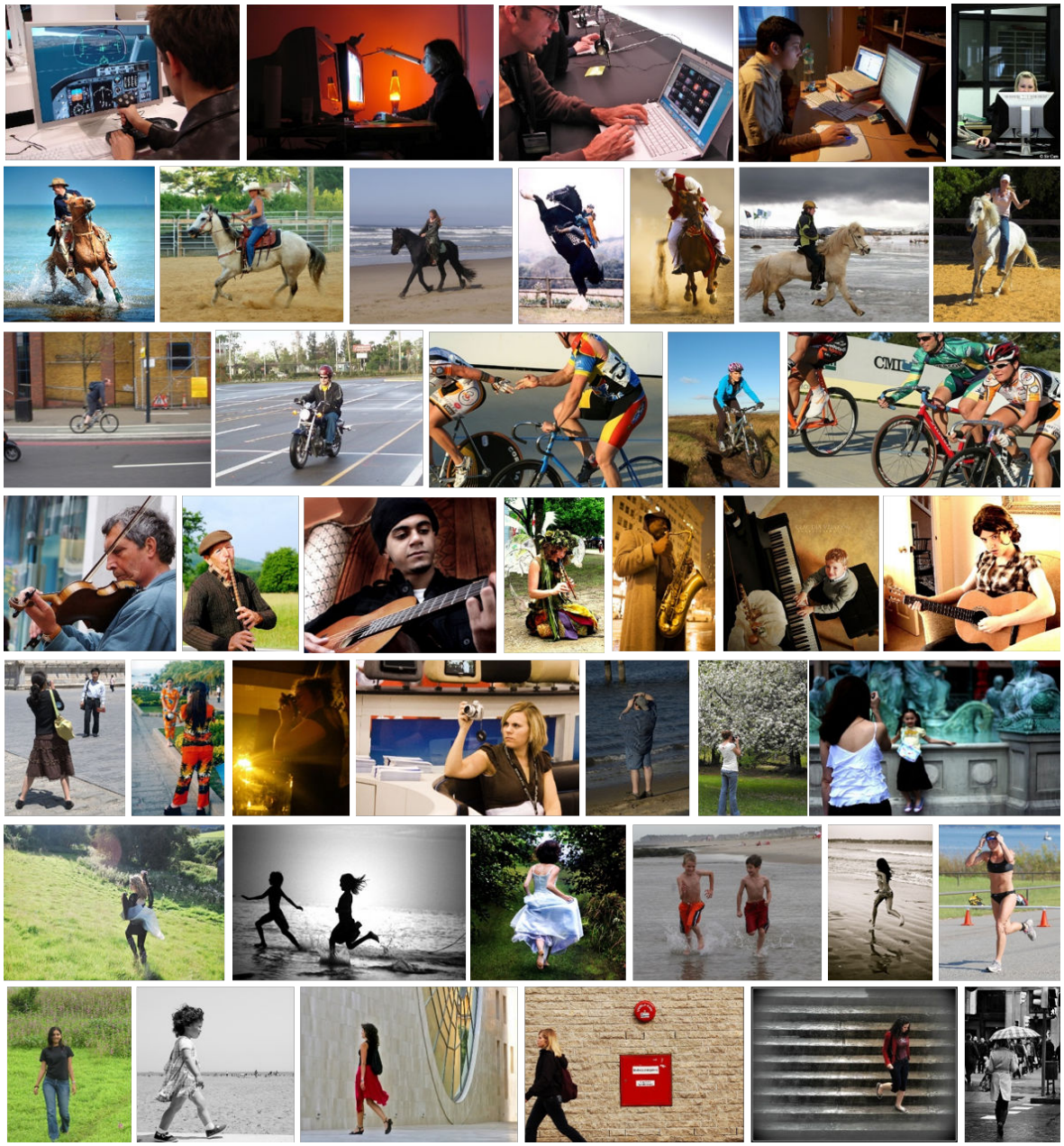


Figure 2.8 – Example images in `Willow` test dataset: each row from top to bottom belongs to classes: *Interacting with computer, Riding Horse, Riding Bike, Playing Instrument, Photographing, Running, Walking.*

Chapter 3

Transfer Learning via Attributes for Improved On-the-fly Classification

Contents

3.1 Introduction

Many works *e.g.* [64, 27] have been proposed to address the problem of classification of scenes and objects. The problem has been traditionally addressed in a supervised learning scenario where the task is to learn an image classifier when some *positive* examples i.e. images containing the scene or object of interest are given. However, systems developed with such assumptions suffer from obvious limitations: (i) the number of scene or object categories is very large and (ii) annotating, let alone conceiving all textual queries that users might be interested in, is impractical. To address these limitations Chatfield and Zisserman [11] proposed to learn visual classifiers for the user provided textual queries, *on-the-fly*. In their proposed method, they first used the user query to search for images on the internet using an image search engine. Then they used the top images returned by the search as the visual examples of the query and trained the corresponding visual classifier against a fixed set of generic negative images. They then finally used this classifier to obtain a ranking of the images in the database.

However, relying solely on internet image search for on-the-fly classification leads to the following problem in practice. As shown in Fig. 3.1, quite a few of the top ranked images returned for even simple queries contain (i) objects with artifacts, or (ii) objects in rare and/or unusual poses/appearances/viewpoints or (iii) artistic/'professional' images with misleading (*e.g.* white) background context. These images degrade the performance of the on-the-fly classifier. For example, Fig. 3.2 shows the retrieval results for three different animal queries, with the classifier trained with one such image as the only positive example ([76]). We can see that clearly such images are not suitable for the task. Thus one of the basic task addressed by the present work is automatic filtering of such images towards the goal of improving on-the-fly classification.

We propose to do such filtering by performing *transfer learning*. We use the domain of *attributes* e.g. 'furry', 'has four legs', 'spotted' etc. and transfer knowledge from here to the domain of on-the-fly query based classification. Like many previous works [28, 63, 113] we argue that attributes are useful for visually characterizing a class. A large set of classes can be potentially covered by combining smaller number of attributes. Hence, if we have annotations for a relatively small set of attributes we can transfer this knowledge to the much larger set of all queries resulting from the combination of these attributes. We note here that although a large number of works using attributes have been reported in the recent past, such use of attributes in the context of on-the-fly classification has never been explored before.

The two main contributions of this work are for improving on-the-fly classification with the help of attributes. First, we use the attributes to do a *zero-shot* classification on the set of images returned by the internet image search. We discard the images which score low with such attribute based classifier as they are likely to be visually less informative, if not completely wrong or misleading. Second, we use this zero-shot classifier to also score the database images. We combine this score with that obtained by the on-the-fly classifier to obtain a hybrid ranking of the images. We describe our two contributions in more detail in Sec. 7.3. We show qualitative and quantitative experimental results (Sec. 3.4) to demonstrate that the proposed approaches improve the baseline on-the-fly system. We now discuss some closely related works in the following section.

Figure 3.1 – Some top ranked images retrieved by Google image search for query 'dog'.

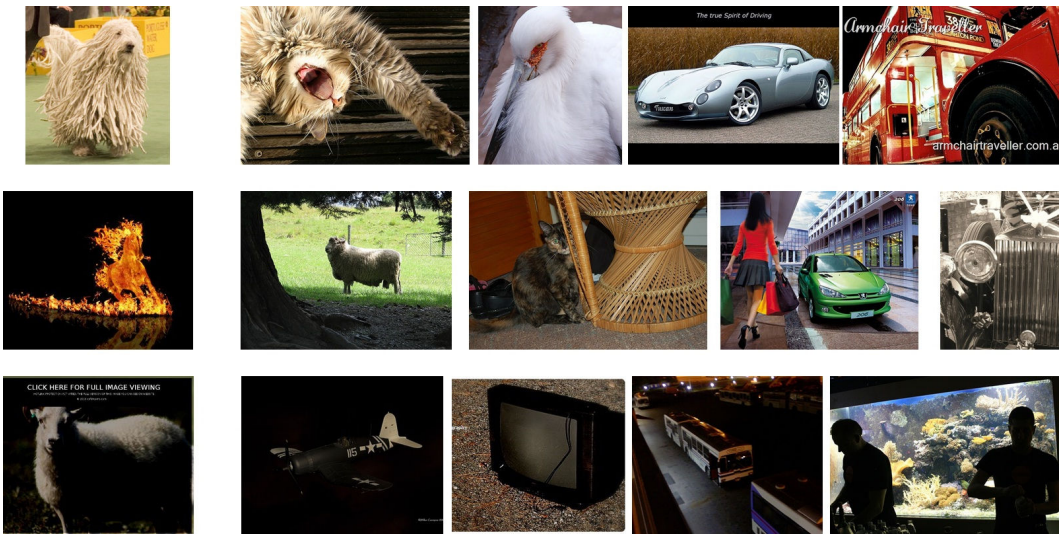


Figure 3.2 – Top retrievals for some of the non-pertinent images returned by Google image search: for each row, the images from Google (on the left) were used to train exemplar SVMs [76] which were in turn used to retrieve images from the Pascal VOC 2007 dataset with some top false positives shown on right. Note how rare appearances, abstract/artistic representations and misleading background leads to poor retrieval *wrt* the queries.

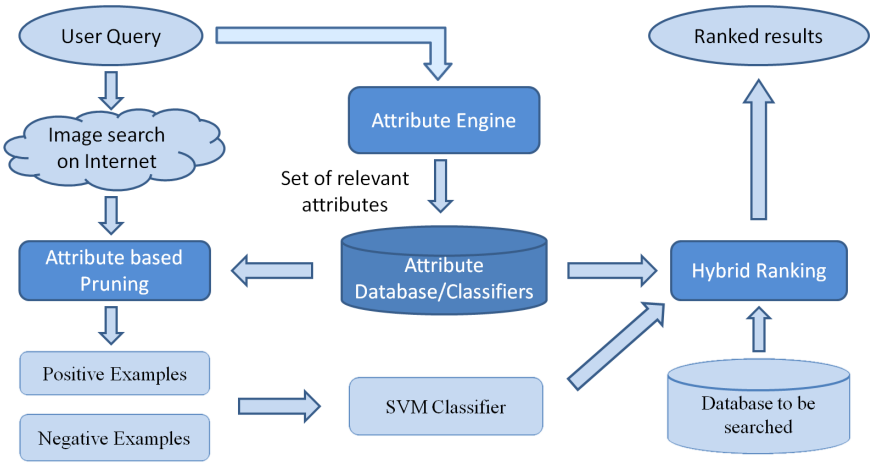


Figure 3.3 – Block diagram of the proposed system. Our contributions are highlighted (see Sec. 7.3).

3.2 Related Works

Our work is primarily related to image classification, on-the-fly classification, attributes and transfer learning. We have already discussed briefly the transfer learning approach in context to CNN in Chapter 1 under section Section 1.1.6. In this chapter, we describe the transfer learning and its related work in context to attributes. We also discuss some of the closely related works under this section.

Image classification Many of the current classification systems e.g. [11, 27, 64, 102] are based on the Bag-of-Words representation [15, 108] where local appearance features (e.g. SIFT [74]) are extracted for patches on a dense grid over the image and then vector quantized w.r.t.a visual codewords. The visual codewords itself is learned offline by performing standard clustering algorithm e.g. k-means on randomly samples features from the training images. To encode some spatial information [64] proposed to pool over spatial cells i.e. make a spatial pyramid gaining substantial performance. We follow these works here and use this image representation.

On-the-fly classification is an extension of supervised image classification to the case of arbitrary user-specified queries [11]. It addresses the limitation of standard classification, i.e. necessity of annotated positive images pertinent to the query, by constructing a positive set of images on-the-fly by querying image search engines on the internet. After such positive images are obtained standard classification algorithms are applied.

Attributes have become quite popular in computer vision. They have been used to describe objects [28], and improve image classification performance [113]. An interesting use of attributes was shown in zero-shot learning [63, 133] where classifiers were learnt without any images for the class via attributes *e.g.* Yu and Aloimonos [133] learned the generative attribute models which were used as priors. These attribute priors were shown to improve image classification performance in zero-shot and/or one-shot learning framework.

A lot of work has also been done to represent the images in low dimensional attribute space [67] with the coefficients of the feature vector as the scores of the attribute classifiers. [126] proposed to learn image similarity from Flickr image groups. Vogel and Schiele [124] represented images by concatenating the local semantic descriptions into one large global descriptor and then used them for retrieval of natural scenes. [117] used large number of weakly trained attribute classifier outputs to represent an image and used it for classification. [62] used attribute classifiers (e.g. gender, race, hair color) for the task of face verification i.e. to tell if two given faces are of the same person or not.

In addition to image representation using attributes, a lot of work in cognitive science [45, 82, 83, 109] has focused on understanding how humans perceive the relations between attributes and objects.

Transfer learning is defined as using the knowledge learned in one task to new task which share some statistical relationship. [24] learned a target classifiers using a set of independent auxiliary classifiers learnt in some other domain. [131] proposed to address two problems in classifier adaptation. First, adapting the multiple classifiers learnt in auxiliary domain to do classification in target domain and second, learning

the selection criteria for best classifier in auxiliary domain. [25] proposed to learn both cross domain kernel function and also robust SVM classifier in video concept detection. Wu and Dietterich [129] worked in SVM framework and used the kernel derived from auxiliary domains, containing large amount of training data, in the target domain containing very less training data.

Transfer learning has also been applied to attribute based image classification. Russakovsky and Fei-Fei [93] proposed to obtain the visual connection between object categories based on transfer learning. They started with learning 20 visual attributes from ImageNet data and used these attributes to find the connection between the object categories.

3.3 Approach

In the following, first we set the background context by describing the on-the-fly classification [11] method briefly. We then describe our proposed attribute-based positive image set pruning via zero-shot. Finally, we describe our proposed hybrid ranking system obtained by combining the on-the-fly classifier and an attribute-based zero-shot classifier. Fig. 3.3 gives the overall block diagram of the proposed system with blocks corresponding to our novel contributions highlighted.

3.3.1 On-the-fly classification

On-the-fly classification is a method of image retrieval based on arbitrary user queries. It uses standard binary supervised classification setup with the positive images obtained from internet image search while keeping the negative images fixed (a set of generic negative images). As the user provides a query, the system makes the same query to an image search engine on the internet. Then the system downloads the top images returned for the query as the positive examples of the query. These are used to learn a SVM classifier which is in turn used to score the database images. The number of positive images obtained is small and the features for the negative set is already cached, hence the overall features are obtained in reasonable time. A linear SVM classifier is usually learned using stochastic gradient descent which is also fast. Finally, the pertinence score, for the database images, is just a dot product between the learnt classifier and the previously computed and cached features of the database images.

3.3.2 Attribute-based pruning

However, as discussed in the introduction (Sec. 3.1) the problem with using internet based image search is the risk of obtaining uninformative and/or rare and misleading images as positive examples (Fig. 3.1). To prune out such images we propose to use transfer learning based on an auxiliary domain of attributes. We propose to do this by constructing a *zero-shot* classifier, inspired by the work of Lampert et al. [63], by mapping the query to a subset of attributes in the attribute dataset. Such mapping could be obtained by a textual analysis system. Using the attributes we learn a zero-shot classifier as follows. We learn a set of attribute classifiers $\{a_i | i \in \mathcal{A}\}$, where $\mathcal{A}$ is the set of attributes, offline. Given a test query q, we obtain a set of attributes $\mathcal{A}_q$ corresponding to the query. We then calculate the score matrix for the attribute classifier for all the images X^+ downloaded from the internet:

$$A = (a_i^{\mathrm{T}} x_j)_{ij} \forall i \in \mathcal{A}_q, x_j \in X^+.$$

In order to bring the scores of the different attribute classifiers into the same scale, we then normalize the attribute score matrix along its rows. Letting μ_i and σ_i denote, respectively, the mean and variance of the entries in the i-th row of A, the normalized score matrix A' is given by

$$A' = (\frac{A_{ij} - \mu_i}{\sigma_i})_{ij} \tag{3.1}$$

Finally the attribute based score is given by the sum over all the attributes for positive images. This is our zero-shot attribute based classifier score as it was derived by transferring knowledge from the auxiliary attribute domain, and without the need for training images for the query. The resulting scores are indicative of the presence of the attributes related to the query in the corresponding downloaded images. We hence discard the lowest scoring k images, as they are likely uninformative. Using these pruned images as positive examples, we learn a linear SVM classifier $\mathbf{w}$. We then compute the pertinence of the images in the database X^R, w.r.t.on-the-fly classifier, as

$$s^o = \mathbf{w}^{\mathrm{T}} X^R. \tag{3.2}$$

3.3.3 Hybrid ranking with attributes and on-the-fly classifier

The attribute based zero-shot query classifier can also be used to test the pertinence of the images in the retrieval database X^R. To this end, we build a score matrix $B = (a_i^{\mathrm{T}} x_j)_{ij} \forall i \in \mathcal{A}_q, x_j \in X^R$ using the same attributes $\mathcal{A}_q$ used for the positive set pruning process. The score matrix B is also centered and normalized row-wise as in (3.1) to produce B'. The summation over the columns of B' (*i.e.*scores from only relevant attributes $\mathcal{A}_q$) given by

$$s^a = \mathbf{1}^{\mathrm{T}} B' \tag{3.3}$$

is score of the database images w.r.t.the query, based on zero-shot attribute classifier.

We propose a hybrid ranking score that combines the two pertinence scores s^o and s^a. To do so, we need to bring the two scores into the same scale. Letting (μ_1, σ_1) and (μ_2, σ_2) denote, respectively, the mean and variance of s^o and s^a, we define the hybrid score for image k as

$$\alpha \frac{s^o_k - \mu_1}{\sigma_1} + (1 - \alpha) \frac{s^a_k - \mu_2}{\sigma_2} \tag{3.4}$$

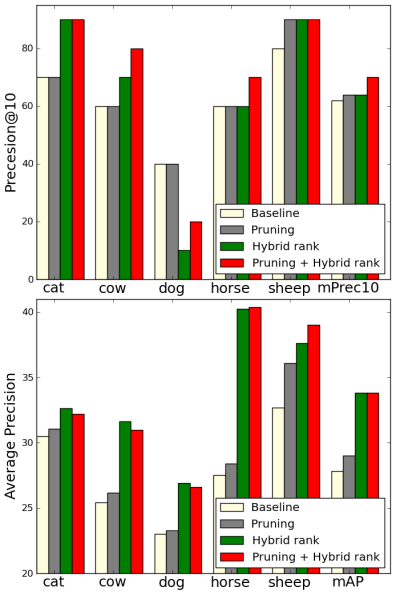
The weight α controls the relative importance of the attribute-based score s^o and the score s^a of on-the-fly classifier with pruned images.

3.4 Experimental results

We validate our method on publicly available `Pascal-VOC-2007` [27] dataset. We restrict our domain of interest to the animal classes. For the auxiliary domain of attributes for transferring knowledge we use the Animals with Attributes [63] dataset. We use the original on-the-fly system [11] as the baseline method and show by experiments how our methods improves the baseline. We first give details of the datasets and the implementation and then proceed to show our quantitative and qualitative results.

3.4.1 Datasets used

`Pascal-VOC-2007` We use all the test images but restricting the performance evaluation to the domain of animals, we report results using five animal classes as queries i.e.



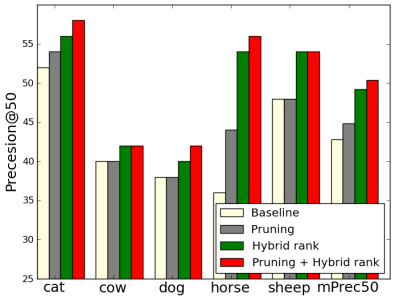


Figure 3.4 – The performances (precision at 10 and 50 and average precision) for all the animal classes, along with mean performances, for the proposed method vs. the baseline of on-the-fly classification [11]. See Sec. 3.4.3 for discussion.

Figure 3.5 – Top false positive for 'dog' class. We see that the results obtained are dominated by animals with closely related attributes like 'furry', 'long legs'. See Sec. 3.4.3 for more discussion.

horse, cat, dog, cow and sheep. Note that the other classes are present as distractors in the test set. Performance is evaluated by computing the precision at 10 images, that at 50 images and the average precision for each class, as well as the mean of the three metrics for all the five classes.

Animals with Attributes dataset [63] consists of 30475 images. There are in total 50 animal classes with at least 95 images of each class. Annotations are also provided for 85 attributes related to the animals. We use this database as the source of auxiliary knowledge to be transferred to the query classification domain. [56] computed a matrix with values specifying the relative strength of association of attributes with object categories. This matrix was built based on feedback by human subjects on association strength between 50 animal classes with 85 attribute categories. We use this matrix in our case to train our attribute classifiers and for the attributes to be used for each animal query, Tab. 3.1 gives the list of attributes used for each of the five animal queries.

3.4.2 Implementation details

Internet image search. We use Google Image search via the publicly available API to obtain image results obtained for a given textual query. Once the results are obtained

Table 3.1 – The attributes used for the five animal queries.

Query	Attributes
horse	furry, big, toughskin, hooves, longleg, longneck, fast, strong, agility, quadrapedal, vegetation, grazer, plains, fields
dog	spots, furry, paws, claws, lean, longleg, fast, strong, quadrapedal, active, inactive, plains, fields, ground, fierce, solitary, newworld
sheep	furry, bulbous, hooves, mountains, ground, timid
cat	furry, small, quadrapedal, weak, active, inactive, agility, hunter, newworld
cow	patches, spots, toughskin, hooves, horns, big, quadrapedal, vegetation, grazer, plains, fields, ground, group

we download the images in the search results with a timeout threshold. On an average we download about 85 images per query due the limitation imposed by the API and the timeout threshold.

Bag-of-Words (BoWs). We represent images similar to [11] with BoWs histograms. We use densely sampled gray scale SIFT features extracted at 4 scales with step size of 3 pixels. We use VLfeat library [123] for extracting SIFT features. We learn a visual codebook of size 4,000 using randomly sampled SIFT features from the Pascal VOC 2007 $train + val$ dataset. We use nearest neighbor based hard assignment of SIFT features to codebook vectors. Finally, we use three level spatial pyramid [64] by dividing the image into 1x1, 3x1 and 2x2 spatial grid.

Attribute based transfer. For the zero-shot classifier based pruning we discard the bottom $k = 8$ images and for the hybrid scoring, the weight parameter is set to $\alpha = 0.3$, both parameters were set based on validation experiments.

3.4.3 Quantitative results

Fig. 3.4 shows the results of the proposed methods vs. the baseline of on-the-fly classification scheme of Chatfield and Zisserman [11]. We can make the following observations. First, the attribute based pruning of test images improves the performance over the baseline by a modest amount specially at the higher end of recall (precision at 50 and mAP). This is consistent for all the classes. Second, doing hybrid attribute and on-the-fly classification based ranking gives large performance improvements again at the higher end of recall. Third, doing both pruning and then hybrid ranking improves performance at lower end of the recall.

The proposed method improves the baseline in all cases except the 'dog' query, where the number of true positives among the top 10 retrieved images decreases from 4 to 2. We analyzed the results in this case and found that the top retrieved images for this case were those of animals with very closely related attributes e.g. 'furry' and 'has four legs'. Fig. 3.5 shows some of the top false positives. However, we note that the performance recovers at higher recall e.g. the precision at 50 and the average precisions improve for dog class as well.

Processing times. The time (on a single core) taken by different steps are as follows. Downloading 85 images (for a query) takes 4s using the Google Search API. Feature extraction takes $\sim 3s$ per image, (ii) SVM learning takes $\sim 6s$ and scoring the database images (which is matrix dot product and sum) is negligible. When multiple cores are used the system is reasonably responsive.

3.4.4 Qualitative results

Fig. 3.6 shows some qualitative results for our system. Each row corresponds to an animal query, on the left the images retained, for training the on-the-fly system, and on the left the discarded images are shown. We can see that the images which are more natural and are pertinent to the queries are retained by the proposed method. While the image which have either rare object appearance/pose or uninformative/misleading background or are abstract/artistic rendering of the animals have been discarded.

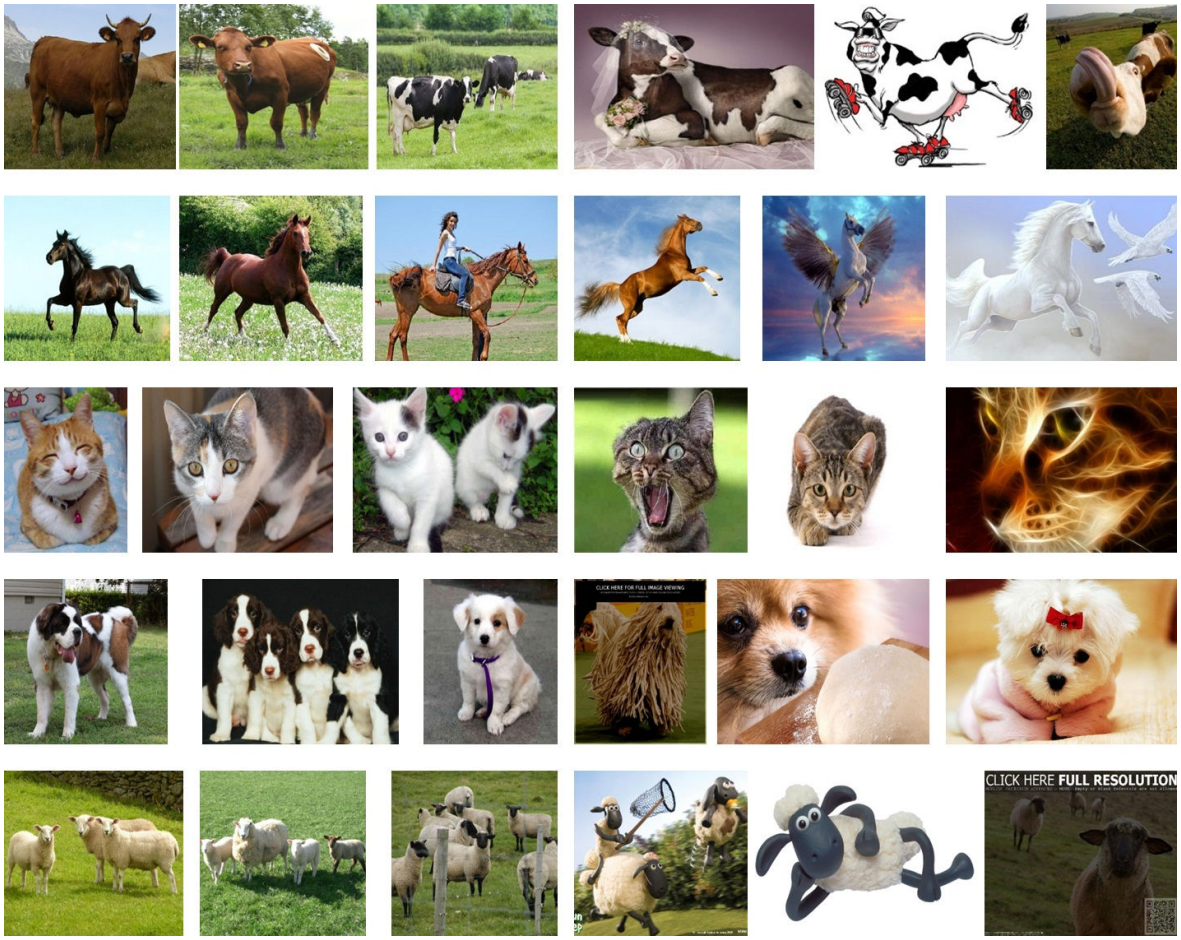


Figure 3.6 – Example of images from Google image search which are (left three) retained by our system as pertinent to the query (animals i.e. cow, horse, cat, dog, sheep, from top to bottom) and (right three) discarded by our system as being uninformative. We see that the images which are more natural and pertinent for the class queries and retained while those which have objects on rare or unnatural poses, or objects on uninformative/misleading background or artistic or abstract rendering of the objects are discarded by the system.

3.5 Discussion and conclusion

In the present chapter we presented a method to use attributes to improve the classification performance of on-the-fly [11] classification for retrieval. We showed that transferring knowledge from the attribute domain to the query domain is effective in pruning out images containing (i) objects in rare or unnatural poses, (ii) objects on uninformative/misleading background and/or (iii) artistic or abstract rendering of the objects. We also proposed a hybrid ranking system which, along with the on-the-fly classification, takes the attribute classifiers into account. We showed by experiments on standard publicly available datasets that our methods improves upon the baseline.

The attribute engine, which maps the query to a set of relevant attributes, was assumed to be given in the present work. The design of an automatic attribute engine is a challenging future work that we would like to pursue. Also, the attribute dataset was assumed to be given, fixed and annotated. It would also be interesting to explore creation or extension of such attribute datasets on-the-fly as well.

Chapter 4

Hybrid multi-layer CNN/Aggregator feature for image classification

Contents

4.1 Introduction

In this chapter we propose a new hybrid image feature for image classification obtained from a mix of the classical image feature extraction pipeline and the more recent and very successful Convolutional Neural Network (CNN) pipeline.

As already described in Chapter 2, the classical image feature extraction pipeline consist of three major steps: 1) Extracting local descriptors such as SIFT [75] from the image; 2) mapping these descriptors to a higher dimensional space; 3) and sum or max-pooling the resulting vectors to form a fixed-dimensional image feature representation. Examples of methods corresponding to this classical approach include Bag-of-Words (BoW) [15], Fisher Vector (FV) [87], Locality-constrained Linear Encoding [127], Kernel codebooks [121], super-vector encoding [138] and VLAD [19]. We refer to these type of image feature extraction schemes as *aggregators* given that they aggregate local descriptors into a fixed dimensional representation. Generally these approaches require computationally inexpensive unsupervised models of the local descriptor distribution, and the resulting image features can be used to learn likewise inexpensive linear classifiers using SVMs.

CNNs consist of multiple interconnected layers including spatial convolution layers, half-wave rectification layers, spatial pooling layers, normalization layers, and fully connected layers. While this method attains outstanding classification performance, it also suffers from large testing complexity, particularly due to the first fully connected layer, as well as large training complexity, since all the coefficients in the pipeline are learned in a supervised manner and require lots of training images. To address this latter issue, [79] proposed to use CNN models pre-trained on the Imagenet dataset (consisting of many million images) and then transfer all but the last layer of this pre-trained CNN to a new target dataset, where two new adaptation layers are learned. This reduces training time and the amount of required training data, but the training data needs to be annotated with bounding box information. The fact that the method works on a per-patch basis further increases the testing complexity relative to standard CNNs.

Several approaches exist that, like ours, attempt to bridge the classical approach and the CNN approach using hybrid mixes. Inspired by the popularity of CNNs, Simonyan [105] proposed to incorporate the deep aspect of CNNs into traditional SIFT/FV schemes by stacking multiple layers of FV aggregators, with each layer operating on successively coarser overlapping spatial cells. Sydorov [115] instead proposed viewing the standard FV aggregator as a deep architecture, substituting the unsupervised GMM parameters of the FV aggregator by supervised versions.

While these methods adopted only the deep aspect of CNNs, our goal is to combine the advantages of both approaches (CNNs and classical aggregators) using hybrid mixes of both pipelines. We do this by treating the output of the pre-trained intermediate layers of the CNN architecture as local image descriptors, which we aggregate using standard aggregators such as BoW or FV. There is no need to carry out costly tuning of the CNN adaptation layers [79] to the target dataset, as both BoW and FV rely on unsupervised learning. The closest related method in the literature is that of Gong *et al.* [40], who propose using the output of the previous-to-last fully connected layer as a local descriptor, computing this descriptor on multi-scale dense patches subsequently aggregated using VLAD on a per-scale basis. This approach is very complex because, contrary to our approach, one needs to compute the full CNN pipeline not only on the original image but also on a large number of multi-scale patches and further apply two

levels of PCA dimensionality reduction.

The remainder of this chapter is organized as follows: In Section 4.2, we describe the two classical aggregators (BoW and FV) that we use in our experiments, as well as the CNN architecture. In Section 4.3, we describe our hybrid image feature extraction pipeline. We evaluate our proposed method in Section 4.4 and provide concluding remarks in Section 4.5.

4.2 Background

In this section we present an overview of two classical local descriptor aggregation methods: the BoW aggregator [108, 64, 4] and the FV aggregator [88]. Up until recently, such aggregation schemes together with SVM classifiers were the reference in image classification [9]. We then present an overview of the new state-of-the art CNN image classification pipeline [59].

4.2.1 Image Classification using Local Descriptor Aggregators

The classical image classification procedure consists of first mapping images to a fixed-dimensional image feature space where linear classifiers are computed using SVMs. The image feature construction process operates by aggregating the local descriptors extracted from the image in question, $\mathbf{f} : \{\mathbf{x}_k \in \mathbb{R}^d\}_k \mapsto \mathbb{R}^D$, where the $\mathbf{x}_k$ are the local descriptors of the image.

The Bag-of-Words (BoW) aggregator offers one such way to map local descriptors to image features. A training set of local descriptors $\mathcal{T}$ from a representative set of images is first used to build a codebook $\mathbf{C} = [\mathbf{c}_j]_j$ using K-means. Letting $\mathcal{C}_j$ denote the Voronoi cell for codeword $\mathbf{c}_j$, the BoW aggregated image feature is the relative frequency of occurrence of local descriptors in the Voronoi cells:

$$\mathbf{f} = \left[\#\left(\{\mathbf{x}_k, \mathbf{x}_k \in \mathcal{C}_j\}_k\right)/\#\left(\{\mathbf{x}_k\}_k\right)\right]_j, \tag{4.1}$$

where we let $\#$ denote set cardinality. The BoW encoder offers an intuitive image feature and enjoys a low computational cost that can be important in user-in-the-loop applications such as [86].

A more recent image feature, the Fisher vector, offers an important gain in image classification performance [9]. The Fisher encoder requires that a training set of local descriptors $\mathcal{T}$ be used to learn a GMM model $\mathcal{G} = \{\beta_k, \Sigma_k, \mathbf{c}_k\}_k$ with k-th mixture component having prior weight β_k, covariance matrix (assumed diagonal) Σ_k and mean vector $\mathbf{c}_k$. The first order Fisher vector for a given image can then be computed as follows:

$$\mathbf{f} = \left[\frac{1}{M}\sum_{k=1}^{M}\frac{p(j|\mathbf{x}_k)}{\sqrt{\beta_j}}\Sigma_j^{-1}\left(\mathbf{x}_k - \mathbf{c}_j\right)\right]_j. \tag{4.2}$$

Both the BoW and Fisher aggregators are built from unsupervised models for the distribution of local descriptors, with supervision coming into play only at the classifier learning stage. CNNs instead construct a fully supervised image-to-classification score pipeline.

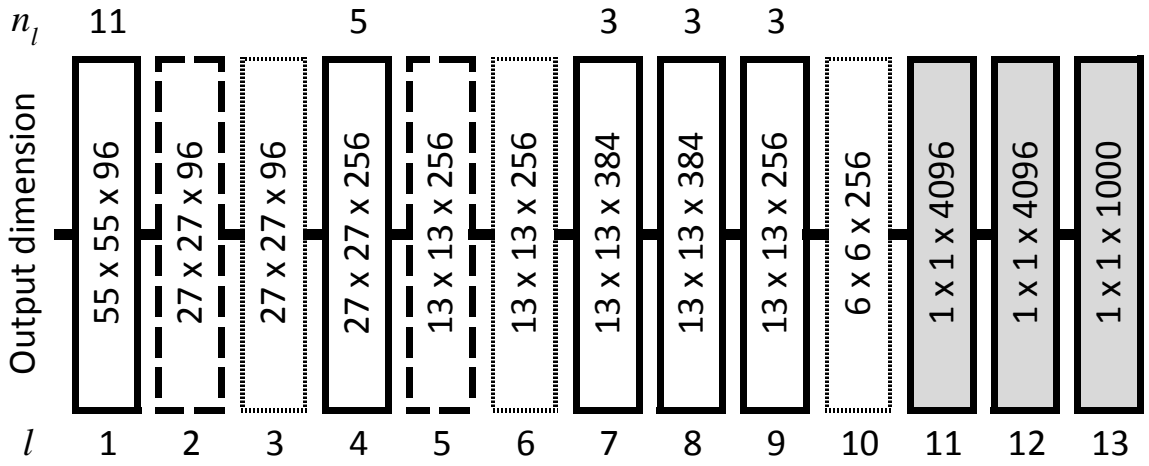


Figure 4.1 – Architecture of the CNN pipeline of [59] trained on ImageNet 2012 and used in this chapter. Each layer, represented by a box, is labeled with the size $R_l \times C_l \times K_l$ of its output in (4.3). The K_l kernels at layer l have dimension $n_l \times n_l \times K_{l-1}$. The layer index l (respectively, kernel spatial dimension n_l) is indicated below (above) the box for each layer. The input image is assumed normalized to size $224 \times 224 \times 3$, and $4\times$ downsampling is applied during the first layer. *Dark-lined boxes:* convolutional layers; *dash-lined boxes:* normalization layers; *light-lined boxes:* max-pooling layers; *grayed-in boxes:* fully-connected layers.

4.2.2 Convolutional Neural Networks (CNNs)

In this section we describe the architecture of Convolutional Neural Networks (CNNs) in more detail. It is well known that CNNs have established an overwhelming presence in image classification starting with the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [59]. The performance gap of CNNs relative to the second entry in that year's competition (and relative to SIFT-based Fisher aggregation schemes [96]) is in excess of 10 percentage points in absolute improvement of top-5 error rate.

In Fig. 4.1 we illustrate the deep CNN processing pipeline of [59]. It consists of convolutional layers, max-pooling layers, normalization layers and fully connected layers. At any given layer l, the layer's output data is an $R_l \times C_l \times K_l$ array

$$[\mathbf{x}_{ij}^l \in \mathbb{R}^{K_l}]_{i=1,\ldots,R_l;\, j=1,\ldots,C_l}, \tag{4.3}$$

that is the input to the next layer, with the input to layer $l = 1$ being an RGB image of size $R_0 \times C_0$ and $K_0 = 3$ color channels.

The *convolutional layers* ($l = 1, 4, 7 - 9$) first compute the spatial convolution of the input with K_l kernels of size $n_l \times n_l \times K_{l-1}$ and then apply entry-wise Rectified Linear Units (ReLUs) $\max(0, z)$. The *normalization layers* ($l = 2, 5$) normalize each $\mathbf{x} \in \{\mathbf{x}_{ij}^{l-1}\}_{ij}$ at the input using what can be seen as a generalization of the l_2 norm consisting of dividing each entry x_m of $\mathbf{x}$ by $(2 + 10^{-4} \sum_{n \in \mathcal{I}_m} x_n^2)^{0.75}$. The summation indices $\mathcal{I}_m$ are taken to be the m-th sliding window over the indices of all entries. The *max-pooling layers* ($l = 3, 6, 10$) carry out per-kernel spatial max-pooling by taking the maximum value from each spatial bin of size 3×3 spaced every 2 pixels.

The *fully connected layers* ($l = 11 - 13$) can be seen as convolutional layers with kernels having the same size as the layer's input data. The last layer ($l = 13$) uses a softmax non-linearity instead of the ReLU non-linearity used in other layers and acts as a multi-class classifier, having as many outputs as there are classes targeted by the system.

4.2.3 Transfer learning using CNNs

The architecture in Fig. 4.1 contains more than 60 million parameters and training it can be a daunting task requiring expensive hardware, large annotated training sets (ImageNet 2012 contains 15 million images and 22,000 classes) and training strategies including memory management schemes, data augmentation and specialized regularization methods. Moreover, extending the architecture to new classes would potentially require re-training the entire structure, as the full architecture is learned for a specific set of target classes.

To address this last difficulty, Oquab *et al.*[79] use transfer learning to apply the architecture in Fig. 4.1 to new classes while incurring reduced training overhead. Their approach consists of substituting only the last fully-connected classification layer by two learned adaptation layers, a fully-connected ReLU layer with 4096 neurons followed by a fully-connected softmax classification layer with as many neurons as target classes. The first 12 layers are transferred from the net in Fig. 4.1 (learned from ImageNet 2012 data), and only the new adaptation layers are learned using training data for the new set of target classes (*e.g.* those of the Pascal VOC 2007 test bench).

While their approach reduces the training overhead and required training set size, training the adaptation layers still requires non trivial complexity as these contain a large number of parameters (more than 16 million). To obtain an adequately large training set from Pascal VOC 2007 data, they derive a patch-based training set, labeling every patch according to its intersection with the provided object bounding boxes. Their approach thus operates on a per-patch classification basis, and the overall class score is obtained by summing this per-patch scores over the entire image for each class. This brings the important benefit of also providing the object localization, but it requires laborious bounding-box annotations on the training set and costly training of millions of parameters.

4.3 A hybrid CNN/Aggregator feature

Inspired by the transfer learning approach of [79], in this section we propose a new hybrid feature that combines parts of the CNN architecture in Fig. 4.1 trained on ImageNet 2012 with the unsupervised BoW or Fisher local descriptor aggregation schemes in (4.1) and (4.2). The resulting feature is used with one-vs-all linear SVM classifiers and hence new classes can be added with little training overhead and without the need for costly object bounding box annotations.

4.3.1 Per-layer aggregation of CNN local descriptors

Our hybrid scheme is based on the observation that the vectors $\mathbf{x}_{ij}^l$ in (4.3) comprising the output of layers $l = 1, \ldots, 10$ in Fig. 4.1 (*i.e.*all layers except fully-connected layers) can be treated as densely extracted local descriptors. We will hence build one aggregated feature $\mathbf{f}_l$ for each layer l (or a subset of layers $l \in \mathcal{L}$) and concatenate all the resulting aggregated layer features to form a single image feature

$$\mathbf{f} = [\mathbf{f}_l^T]_{l\in\mathcal{L}}^T. \tag{4.4}$$

Using only a subset of layers $\mathcal{L} \subseteq \{1, \ldots, 10\}$ allows us to control training, testing and storage complexity and further serves as a means of regularization.

4.3.2 Training per-layer aggregators

In order to train the per-layer aggregators adapted to the CNN layers, we take each image from a representative set of training images and extract from it all vectors $\mathbf{x}_{ij}^l$ for $l = 1, \ldots, 10$. We then group all the resulting local descriptors $\mathbf{x}_{ij}^l$ for each layer l to form a training set $\mathcal{T}^l$ for the l-th layer. Each training set $\mathcal{T}^l$ of local descriptors is then used to train a codebook $\mathbf{C}^l$ for layer l using K-means when using BoW aggregators. Likewise, a GMM model $\mathcal{G}^l$ is learned for the l-th layer when using Fisher aggregators.

4.3.3 Extensions based on classic approaches

Our proposed approach shares similarities with several existing approaches and we now discuss these and related extensions.

One first observation is that the spatial support (relative to the original image) used to compute the $\mathbf{x}_{ij}^l$ is of size 11 (in each spatial dimension) for the first layer and grows by $4 \times 2 \cdot (n_a - 1)$ for each convolutional layer $1 < a \leq l$, yielding possible supports of size $11, 43, 59$ and 75. Dense approaches likewise compute local descriptors from supports of varying size $(16, 24, 32, 40)$ by means of multi-resolution spatial grids [9], but all descriptors for all supports are pooled together (for the benefit of scale invariance) and used to form a single aggregated image feature. A similar pooling approach could be used for CNN local descriptors $\mathbf{x}_{ij}^l \in \mathbb{R}^{K_l}$ by first mapping all layers to a common dimensionality via, *e.g.* PCA or discriminative dimensionality reduction.

The layer feature concatenation scheme (4.4) that we use instead is reminiscent of spatial pyramid matching [64, 4], where one feature $\mathbf{g}_c$ is computed for each spatial cell $c = 1, \ldots, 8$ and these are subsequently concatenated. Our concatenated image features $\mathbf{f}_l$ are instead computed from high-dimensional filtered versions of the image, and indeed this approach can be combined with SPM to produce per-spatial-cell layer features $\mathbf{f}_{lc}$.

Other standard successful approaches can also be combined with our proposed hybrid CNN/aggregator features, including power normalization of the $\mathbf{x}_{ij}^l$ [1], application of an explicit Hellinger kernel-map to our hybrid feature [9] and late fusion with other feature channels. Alternate aggregation schemes such as VLAD or triangulation embedding [19, 51] can also be used, but we chose BoW for its low computational cost and Fisher given that is the best performing aggregator in classification.

4.4 Results

In this section we validate our proposed hybrid CNN/aggregator feature using the publicly available `Pascal-VOC-2007` dataset [27]. We use the standard mean Average Precision (mAP) measure computed over the test set as a performance metric.

4.4.1 Impact of layer subset $\mathcal{L}$

In Fig. 4.2 we evaluate the impact on performance of the layer subset $\mathcal{L}$ in (4.4) used to build hybrid features. We consider three strategies for selecting $\mathcal{L}$: using a single layer, $\mathcal{L} = \{L\}$, using the first L layers, $\mathcal{L} = \{1, \ldots, L\}$, and using the last L layers, $\mathcal{L} = \{10, 9, \ldots, 10 - L + 1\}$. As seen in Fig. 4.2, the results for the single-layer strategy indicate that layers further down the pipeline are more informative (although the curve is not monotonic). Indeed the best strategy overall consists of using the last 5

layers (and using only 3 layers results in marginal performance decrease). The resulting hybrid feature performs substantially better than BoW+SPM with 4,000 codewords and performs similar to FV+SPM with 256 mixture components [88], despite being 150 times smaller.

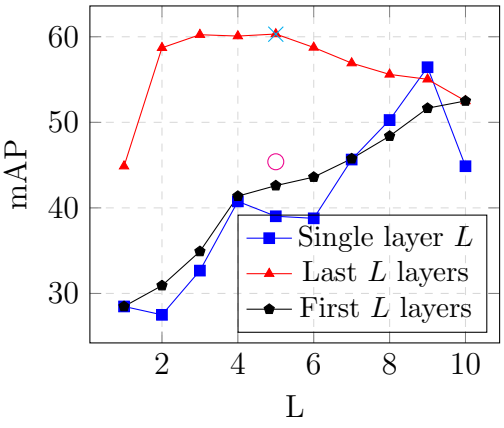


Figure 4.2 – The mAP is plotted for hybrid features built using a single layer, the last L layers and the first L layers (excluding fully connected layers 11-13), for codebook size 500. Baseline results for BoW and FV are displayed using ∘ and × markers.

4.4.2 Impact of codebook size

In Fig. 4.3 we evaluate the impact on performance of varying the codebook size when using hybrid CNN/BoW features built from the last 5 layers. A codebook of size 500 yields the best performance. And even with a codebook size of 30, which amounts to a feature vector size of 150, our method outperforms BoW + SPM.

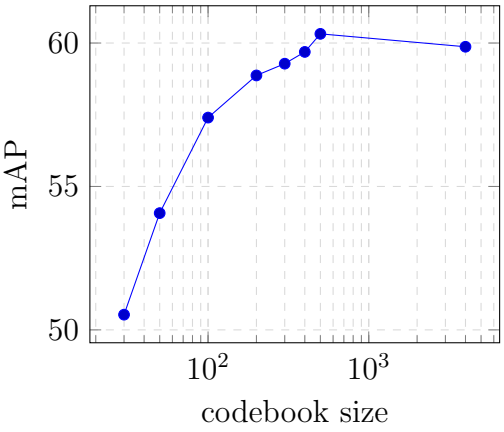


Figure 4.3 – The mAP vs the codebook size in log scale when using last L=5 layers.

4.4.3 Comparison to other approaches

In Table 4.2 we compare our results with some of the best results reported in the literature. We include results for hybrid features built using FV aggregators with 64 mixture components. Despite the established superiority of FV aggregation over BoW aggregation, the FV-based hybrid features perform poorly relative to BoW-based

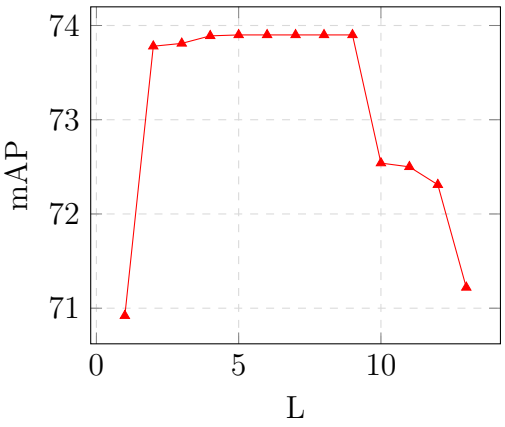


Figure 4.4 – Using last L layers from Fig. 4.1. Here we include the fully connected layers 11-13.

method	Training time+resource
PRE1000C [79]	≈ 1 day (GeForce GTX Titan GPU)
Hybrid CNN/BoW, N=500	≈ 1hr + 5min (8 core CPU)

Table 4.1 – Table illustrating training time for 500 codebook size and when using the last 5 layers. Training times are for the unsupervised learning part with and without supervised learning of linear SVM classifiers for all Pascal VOC 2007 classes. This is compared to the training time taken by the method [79].

hybrid features. We believe that this is due to the small number of local descriptors in CNN layers, as this makes the vector-averaging process in (4.2) statistically noisy.

The best performing system in Table 4.2 is PRE1000C [79]. Their approach consists of substituting layer 13 in Fig. 4.1 by two adaptation layers trained on Pascal VOC. As is the case for CNN pipelines, this training procedure is time consuming and requires expensive GPU cards, as illustrated in Table 5.1. Furthermore, at testing time, their approach requires applying the full 13-layer CNN pipeline to each of 500 patches from an image, increasing testing complexity considerably. Our approach requires a single CNN pipeline pass over the non fully-connected layers, resulting in dramatically lower testing time, as the CNN complexity is largely concentrated in the first fully-connected layer.

The same complexity problem is incurred by the feature construction scheme of [40], where the authors propose using the output of CNN layer 13 as a local descriptor computed on multi-scale dense image patches. Inspired by this approach, we further consider stacking the output of the fully connected layers (11, 12, and 13) to our hybrid CNN/aggregator feature. We illustrate the results of this approach in Fig. 4.4, where the non-fully connected layers are processed according to (4.4), and the fully-connected layers are concatenated without any processing. Note that using the 3 fully connected layers and the last non-fully connected layer results in performance close to 74 mAP points. This compares very well to the performance of 77.73 of PRE1000C in Table 4.2, particularly considering the drastic difference in training time and testing time.

method	feature dimension	mAP
BoW + SPM, N=4000 [9]	32000	45.39
FV (SIFT) [88]	262144	58.3
FV (SIFT + color) [88]	262144	60.3
PRE1000C [79]		77.73
Hybrid CNN/FV, m=64	81920	54.56
Hybrid CNN/BoW, N=30	150	50.53
Hybrid CNN/BoW, N=500	2500	60.32

Table 4.2 – Comparison of our results (using last 5 layers) with the state-of-the-art (N represents the codebook size in BoW).

4.5 Conclusion

In this work, we proposed a hybrid Convolutional Neural Network (CNN) / Bag-of-Words (BoW) image feature extraction approach. Treating the output of intermediate layers of a pre-trained CNN as local descriptors allowed us to use an unsupervised Bag-of-Words aggregator to obtain an image feature that outperforms standard aggregators based on local descriptors substantially on the Pascal VOC 2007 benchmark. Appending the output of the fully-connected layers to our hybrid feature further improves the performance of our approach, making it competive with CNNs variants adapted to Pascal VOC 2007, and at a fraction of the training and testing cost.

Chapter 5

Max-Margin, Single-Layer Adaptation of Transferred Image Features

Contents

In the previous chapter we have presented a novel method which combines the Convolutional Neural Networks (CNNs) with traditional aggregators to reduce the computational complexity of the overall classification system. We have shown that the proposed method was competitive against Oquab *et al.* [79] and Chatfield *et al.* [10] in terms of computational resources (*i.e.* training and test time). However, in terms of classification accuracy, we did not perform very well. Therefore, the work in this chapter is focused on improving the classification accuracy while keeping the computational complexity similar to the previously proposed method of Chapter 4.

5.1 Introduction

CNNs learned on the ImageNet dataset have been shown to be excellent feature extractors that, combined with linear SVM classifiers, yield outstanding results when transferred to target datasets (*e.g.*, Pascal VOC, MIT Indoor 67 and Caltech) not used during the CNN learning process [101]. Given the large number of free parameters in CNN models (tens of millions), learning CNNs directly on these smaller target datasets is a difficult task. Yet recent work [79, 10] has established that it is possible to adapt the transferred CNN parameters to the smaller target dataset to further improve results.

The approach we present herein addresses this scenario by learning a single adaptation layer jointly with linear classifiers under a max-margin objective. Unlike existing adaptation schemes, our learning process is very fast, taking in the order of a few minutes when running on a single core CPU, and does not rely on expensive dataset augmentation methods. We further show that it is possible to obtain very good results with features as little as size 20.

Oquab *et al.* [79] have proposed a related CNN adaptation method that consists of re-learning, on the Pascal VOC dataset, the last two layers of the transferred architecture. The learned adaptation layers operate on patches from the original images, with the patch overlap with the object bounding box determining each patch's label at training time. The approach is hence limited by the availability of expensive bounding box annotations, and in this work we show that comparable results can be obtained without relying on bounding box annotations.

Chatfield *et al.* [10] consider an objective based on a hinge-loss that is similar to the max-margin objective we propose herein. Their adaptation approach consists of continuing the learning process began on ImageNet on the new target dataset, but with a slower learning rate. The transition layer between the last convolutional and first fully connected layer, in particular, is updated in the process, a costly procedure (in terms of learning time and required hardware) given the large size of this layer (tens of millions of coefficients).

5.2 Proposed approach

Our approach consists of jointly optimizing the linear classifiers $\mathbf{w}_1 \dots \mathbf{w}_K$ of the K target classes, along with the model parameters $\mathbf{M} \in \mathbb{R}^{r \times D}, \mathbf{b} \in \mathbb{R}^D$ using

$$\underset{\substack{\mathbf{M},\mathbf{b};\\ \mathbf{w}_1,\dots,\mathbf{w}_K}}{\operatorname{argmin}} \sum_{k=1}^{K} |\mathbf{w}_k|^2 + \frac{C_1}{N} \sum_{i=1}^{N} \ell\left(y_{i,k}\, \mathbf{h}\left(\mathbf{M}\mathbf{x}_i + \mathbf{b}\right)^\top \mathbf{w}_k\right) \tag{5.1}$$

where $\mathbf{h}$ and ℓ represents the Rectified Linear Unit (ReLU) and hinge loss operator, respectively, $\mathbf{x}_i \in \mathbb{R}^D$ are the CNN feature representations of the image and $y_{i,k} \in \{-1, 1\}$

are labels indicating the absence/membership of image i in class k. For notational simplicity, we disregard the classifiers' bias terms.

Block-coordinate SGD. We use block-coordinate Stochastic Gradient Descent (SGD), sequentially updating $\mathbf{w}_1, \ldots, \mathbf{w}_K$, $\mathbf{M}$ and $\mathbf{b}$ on the same batch of b images, correspondingly using b SGD steps per block of coordinates before randomly drawing a new batch (without repetition in each epoch). We initialized $\mathbf{M}$ using r randomly selected training features. Bias term $\mathbf{b}$ and $\mathbf{w}_1, \ldots, \mathbf{w}_K$ are initialized to zero.

Early stopping. To avoid over-fitting of the model to the training data, we use an early stopping criterion guided by the performance over the validation set.

Adaptive learning rate. We use an adaptive learning rate for $\mathbf{M}$ that starts at 1 and decays by half to an empirical minimum of 10^{-2}. This rate is updated every 200 images using cross-validation by running SGD on 50 training images chosen from the next batch. At the end of the process, the final learning rate is used to do a single training run over the validation images.

5.3 Results

We evaluate our method using `Pascal-VOC-2007` as a target dataset, using the standard mean Average Precision (mAP) performance measure. This dataset consists of 4192 test and 5011 training images. We hold out 811 training images, choosing them uniformly over all classes, and use them as a validation set. Each image is represented using the VGG-128 (128-dimensional) CNN model [10].

In Fig. 5.1 we evaluate the impact on test set mAP of varying the numer of rows r of $\mathbf{M}$. We consider two cases: joint optimization of $\mathbf{M}$ and the classifiers, as per (6.4), and optimization of only the classifiers, keeping $\mathbf{M}$ fixed to its initialization value. We can observe that our performance is constant for all output feature sizes r. The adaptation layer provides a large advantage for all r values of as much as 17 points in mAP for $r = 20$, and 1.2 points for $r = 128$.

In Table 5.1 we compare our method with two state-of-the-art CNN adaptation methods [79, 10]. Our result of 77.58 is comparable to the 77.73 mAP of [79], yet our model has $4e3\times$ less free parameters and takes only a few minutes to learn on a single core CPU ([79] takes close to one day on a GPU). The results of [10] are better than ours, but their CNN model is even larger than [79], with a comparable increase in learning time and required processing power.

In Table 5.2 we show the advantage of using an adaptive learning rate versus a fixed learning rate by displaying the test set mAP after a fixed number of iterations. In Fig. 5.2, we illustrate our early stopping approach by plotting the test set, training set and validation set mAP for a sample run.

In Fig. 5.3 we evaluate the effect of batch size in our block-coordinate SGD optimization process, noting that, for a fixed number of epochs, larger batch sizes (up to the training set size) result in better performance.

Method	Train time	Dim	# params.	mAP
PRE1000C [79]	≈ 1 day	-	∼8.5M	77.73
CNN S TUNE-RNK[10]	-	4K	∼100M	82.42
Ours	120s	20	2759	76.24
Ours	190s	70	9550	77.58

Table 5.1 – Comparison of our proposed method with two existing CNN adaptation schemes.

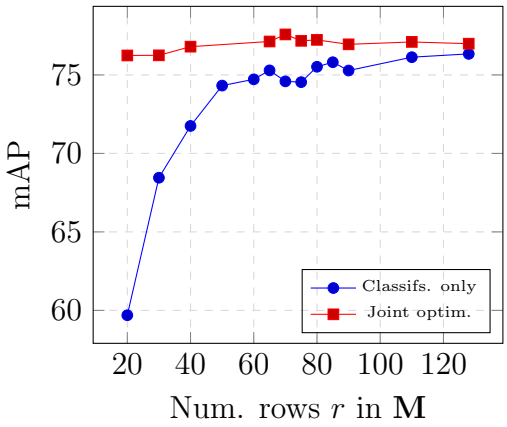


Figure 5.1 – Effect of varying the number of rows r in $\mathbf{M}$ on performance. Here we also compare the results of Joint Optim vs Classifs.

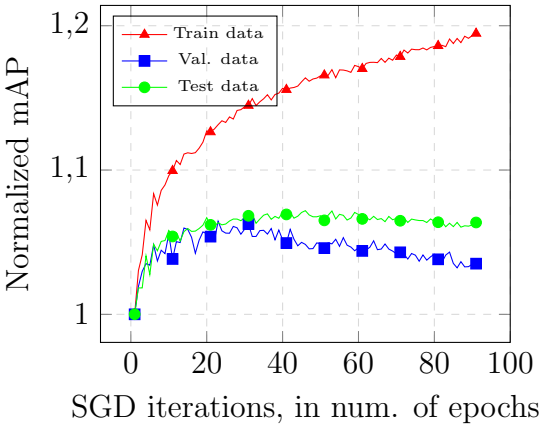


Figure 5.2 – Illustration of cross-validation strategy: the optimal parameters are chosen based on the best performance on the validation set, which occurs at approximately 20 epochs.

	Fixed					Adaptive
Learn rate	10^{-5}	10^{-2}	0.09	0.5	1.0	-
mAP	74.66	75.60	76.81	74.96	73.43	77.25

Table 5.2 – Test set mAP when using fixed learning rate and adaptive learning rate.

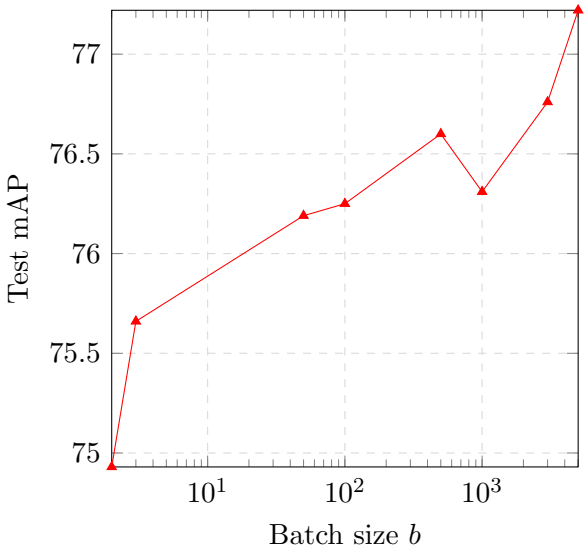


Figure 5.3 – Performance as a function of batch size for 20 epochs of training.

Chapter 6

Learning the Structure of Deep Architectures Using ℓ_1 Regularization

" Sometimes A Scream Is Better Than A Thesis."

Ralph Waldo Emerson

Contents

6.1 Introduction

The first several layers in Convolutional Neural Network (CNN) are usually *convolutional layers* consisting each of n_j (with j the layer index) spatially-convolutional kernels that operate on the n_{j-1} dimensional spatial signals output by the previous layer. Interspersed between these layers are normalization layers and spatial max-pooling layers that can be seen as non-linear convolutional operators. The normalization layers process a single spatial position and effectively balance kernel output energy across space. The max-pooling layers reduce the spatial support of the signal by max-pooling signals in small spatial neighborhoods. Given the convolutional nature of all these operators, they can process input images of arbitrary dimensions.

The latter layers of CNN architectures are instead fully connected layers that require inputs of fixed size. This constraint on input size propagates down the convolutional layers, effectively fixing the CNN architecture's expected input image size (sizes around 225×225 are common [59, 52]). In this respect, the convolutional layers of CNN architectures can be seen as very large fully connected layers that have been constrained to have a weights matrix that is structured and highly sparse, effectively regularizing the architecture.

One important consideration when designing CNN architectures is the choice of weights matrix size across the various Fully Connected Layers (FCLs). Besides being another important means of regularization, the size of the FCL weight matrices has a strong impact on system complexity. The first fully connected layer, in particular, can account for 90% of the number of coefficients in the CNN [59]. When using pre-learned CNNs as generic image feature extractors [101, 40, 10], rectangular weights matrices further have the potential advantage of producing reduced-size feature vectors [10, 61]. But, up to now, the approach used to select the dimensions of the rectangular matrices across the various layers has been empirical, and researchers have mostly focused on using constant sizes across all fully connected layers.

The main contribution of the present work is hence to show that the sizes of the FCL weight matrices can be selected as a part of the supervised CNN learning procedure. We do this by inserting a diagonal matrix $\mathbf{D}^j$ between layers $(j, j+1)$, accordingly penalizing the CNN learning objective with the sparsity inducing ℓ_1 norm on the diagonal coefficients of each $\mathbf{D}^j$. Our method can be seen as formalizing the tradeoff between the generalization power of the model and its storage/computation requirements, as represented by the FCL weight matrix sizes. We then show experimentally that it is indeed possible to choose optimal FCL weight matrix sizes and that these vary with the layer index. Our experiments further show that the approach results not only in smaller feature vectors, but also in higher image classification performance.

The remainder of this chapter is organized as follows: in the next section, we present a review of CNN methods that are related to our work. In Section 6.3, we present our proposed method, subsequently evaluating it experimentally in Section 6.4. We then provide some concluding remarks in Section 6.5.

6.2 Background

Given the large number of free parameters in CNN architectures, regularization is an important consideration when learning deep architectures, and various works have explicitly addressed it. One very successful approach currently deployed in publicly available CNN learning algorithms [52] is *dropout* [111]. Dropout aims to prevent

neighboring units from memorizing training samples. This is accomplished by randomly zeroing a subset of the activation values at the output of each fully connected layer, choosing a different random subset for each training example. The end result is that a different effective training set is used for neighboring weights. A related approach, DropConnect, zeros the FCL weights instead of the activation coefficients [125]. Other regularization approaches instead consist of using unsupervised learning either as an initializing stage before a second, strongly supervised stage, or in a mixed supervised/unsupervised approach [39].

A related line of work consists of pre-learning an entire architecture on a large, generic image dataset and then adapting the resulting architecture to a new target dataset. The approach of [10], for example, adapts the entire architecture by continuing the learning process at a reduced learning rate using the samples from the new target dataset. In [79], the adaptation is instead carried out by entirely re-learning the last two layers on the new target dataset. A similar idea consists of using the activations of the penultimate layer of a pre-learned architecture as an image feature. When used for image classification, SVM classifiers are then learned on top of these features [101], and the collection of such linear classifiers can be seen as a single fully connected adaptation layer. Linear classifiers have also been used in place of the soft-max classifier when learning either the entire architecture [10], or along with one other fully-connected adaptation layer [61].

Various authors have also considered CNN variants of classical approaches including the ubiquitous spatial pyramid [64], the Fisher Vector (FV) [88] and Bag-of-Words (BoWs) [108]. The approach of [60], for example, consists of treating the n_j-dimensional spatial signals at the output of the j-th layer as a densely extracted local descriptor, and then building an aggregated representation such as a bag-of-words or a Fisher vector on top of these local descriptors.

The work of [46] uses a Spatial Pyramid Pooling (SPP) layer between the last convolutional layer and the first fully-connected layer. This should make it possible to use input images of arbitrary size, as the SPP layer maps the arbitrarily-sized output from the last convolutional layer to a vector of constant size compatible with the subsequent fully connected layer. In practice, however, the approach is implemented using a max-pooling layer with large stride.

The approach presented in [40] consists of using a CNN as a local feature extractor by treating each patch from a dense sampling of image patches as an image. The activation features resulting from each patch are then treated as local descriptors to build an aggregated, global feature vector. A similar approach is presented in [38]: given a large number of region proposals derived from the input image, the method extracts CNN activation features for each region and classifies them into given set of classes using linear classifiers. Approaches such as that of [41, 38] that use CNNs as local feature extractors suffer from a very large computational complexity, and could hence benefit from complexity constrained CNNs such as the ones presented herein.

6.3 Learning the structure of deep architectures

In this section we present our proposed approach, illustrated in Fig. 6.1. The architecture we consider consists of a sequence of fully-connected layers, with a diagonal matrix between them. We will constrain the diagonal matrix to have a sparse diagonal, and this will implicitly define the size of the weights matrices of each layer. Rather than using the standard soft-max classification layer as the last layer, we will use a bank of

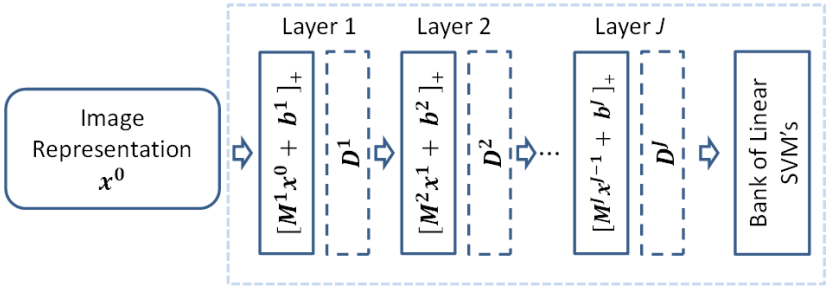


Figure 6.1 – **Proposed deep processing pipeline**. Given an image representation, e.g., the output of the convolutional part of a pre-trained state-of-art CNN, J fully connected layers, each involving a diagonal matrix that controls its effective dimensions, are jointly learned with final linear SVM classifiers. Here, $[\mathbf{z}]_+ = [\max(0, z_i)]_i$ is the commonly used Rectified Linear Unit (ReLU) non-linearity.

linear SVM classifiers similarly to the approaches in [10, 61].

6.3.1 Constraining layer complexity

Formally, we can express the architecture in Fig. 6.1 as a concatenation of units of the following form:

$$f^j(\mathbf{x}) = \mathbf{D}^j \left[\mathbf{M}^j\mathbf{x} + \mathbf{b}^j\right]_+, \tag{6.1}$$

where $[\mathbf{z}]_+ = [\max(0, z_i)]_i$ is the commonly used Rectified Linear Unit (ReLU) non-linearity, here applied to a vector $\mathbf{z} = [z_i]_i$. Input vector is n_{j-1}-dimensional and output is n_j-dimensional. This layer is defined by n_j-dim vector $\mathbf{b}^j$, n_j-dim diagonal matrix $\mathbf{D}^j$ and matrix $\mathbf{M}^j$ of size $n_j \times n_{j-1}$. A deep architecture can be derived from (6.1) using the standard stacking approach. Letting $\circ$ denote the composition operator such that $f \circ g(\mathbf{x}) = f(g(\mathbf{x}))$, this can be denoted as

$$f^J \circ \ldots \circ f^1(\mathbf{x}), \tag{6.2}$$

where, in this case, the vector $\mathbf{x}$ denotes the representation of the image at the input of the architecture. The image representation can consist of a direct re-ordering of the RGB values in the image [59, 10], or it can be a feature derived from the image [79, 61], which is the approach we follow in the present work.

We are interested in the case when the diagonal entries of $\mathbf{D}^j$ are sparse. When this is the case, the corresponding rows of $\mathbf{M}^j$ and $\mathbf{b}^j$ can be removed, as well as the corresponding columns of $\mathbf{M}^{j+1}$. To see this, let $\mathcal{I}$ denote the support (indices of non-zero positions) of the diagonal entries of $\mathbf{D}^j$. Let $\mathbf{A}_{\mathcal{I},.}$, $\mathbf{A}_{.,\mathcal{I}}$ and $\mathbf{A}_{\mathcal{I},\mathcal{I}}$ denote, respectively, the sub-matrices derived from matrix $\mathbf{A}$ by retaining respectively the rows, the columns and both at positions indexed by $\mathcal{I}$. After being processed by the weights matrix $\mathbf{M}^{j+1}$ of the next layer, the expression in (6.1) becomes

$$f^{j+1}(\mathbf{x}) = \left[\mathbf{M}^{j+1}_{.,\mathcal{I}}\mathbf{D}^j_{\mathcal{I},\mathcal{I}}\left[\mathbf{M}^j_{\mathcal{I},.}\mathbf{x} + \mathbf{b}^j_{\mathcal{I}}\right]_+ + \mathbf{b}^{j+1}_{\mathcal{I}}\right]_+. \tag{6.3}$$

In this sense, the sparsity of the diagonal of $\mathbf{D}^j$ encodes the computational complexity of the system, as it selects the effective dimensions of the $\mathbf{M}^j$ matrices and the $\mathbf{b}^j$ vectors.

6.3.2 Problem formulation

The architecture in Fig. 6.1 consists of a large number of parameters. Besides the variables $\mathbf{M}^j, \mathbf{D}^j, \mathbf{b}^j$ associates to each layer $j = 1, \ldots, J$ in (6.1), one needs to learn the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_K$ that define the SVM classifiers for the K classes. We will learn these variables from an annotated training set comprised of N training images $\mathbf{x}_i, i = 1, \ldots, N$, each with K labels $y_i^k \in \{-1, 1\}, k = 1, \ldots, K$ indicating whether image i belongs to class k or not. Given such a training set, our approach consists of minimizing the following objective over all the variables $\{(\mathbf{M}^j, \mathbf{b}^j, \mathbf{D}^j)\}_{j=1}^J$ and all the classifiers $\{\mathbf{w}^k\}_{k=1}^K$:

$$\frac{1}{K}\sum_{k=1}^{K}\left(\|\mathbf{w}^k\|_2^2 + \frac{C}{N}\sum_{i=1}^{N} l\left(y_i^k\left(f_J \circ \ldots \circ f_1(\mathbf{x}_i)\right)^\top \mathbf{w}^k\right)\right) + \delta\sum_{j=1}^{J}\|\mathbf{D}^j\|_1 + \mu\sum_{j=1}^{J}\|\mathbf{M}^j\|_F^2. \tag{6.4}$$

In the above expression, we have used *(i)* $l(x)$ to denote the hinge loss, given by $\max(0, 1-x)$; *(ii)* $\|\mathbf{D}\|_1$ to denote the trace norm, given by $\sum_i |D_{ii}|$ for the case of diagonal $\mathbf{D}$; and *(iii)* $\|\mathbf{M}\|_F^2$ to denote the squared Frobenius norm $\sum_{ij} M_{ij}^2$.

To illustrate the motivation behind this learning objective, we note first that the terms inside the summation over k in (6.4) are recognizable as an SVM objective for class k, where the scalar C is the SVM regularization parameter. The feature vectors used within this SVM objective are given by $f_J \circ \ldots \circ f_1(\mathbf{x}_i)$, which depends on $\{(\mathbf{M}^j, \mathbf{b}^j, \mathbf{D}^j)\}_{j=1}^J$. Hence we are learning the classifiers jointly with the feature extractor used to represent the input images. A similar approach has been used in [115] in the context of Fisher vector encoders to learn the encoder's GMM parameters under a discriminative objective.

The two regularization terms comprised of summations over j in (6.4) serve multiple purposes. One first purpose is to keep the SVM terms from decreasing indefinitely. Recall that the aim of an SVM objective is to maximize the margin between positive and negative examples. If it were not because of the penalty terms $\|\mathbf{w}^k\|_2^2$, it would be possible to minimize this indefinitely by multiplying the linear classifiers by a very large scalar. The penalty terms on the $\mathbf{D}^j$ and $\mathbf{M}^j$ serve a similar purpose when learning the features jointly with classifiers.

A second important purpose is to automatically select the shapes of the weights matrices $\mathbf{M}^j$ and $\mathbf{b}^j$. When applied to diagonal matrices such as $\mathbf{D}^j$, the trace norm is equivalent to an ℓ_1 norm computed from the diagonal vector of $\mathbf{D}^j$, and ℓ_1 norms are known to be well-behaved (*i.e.*convex and differentiable almost everywhere) sparsity inducing norms. They are hence excellent surrogates for the ℓ_0 pseudo-norm that counts the number of non-zero entries of a vector. Since the matrices $\mathbf{D}^j$ multiply corresponding $\mathbf{M}^j$ and $\mathbf{M}^{j+1}$ matrices, low-valued coefficients of $\mathbf{D}^j$ can be compensated with increased norm of rows in $\mathbf{M}^j$ and columns in $\mathbf{M}^{j+1}$. The penalty terms on the norm of the $\mathbf{M}^j$'s hence address this ambiguity.

Approaches other than the one presented in (6.4) and Fig. 6.1 are indeed possible. For example, it would be possible to dispense altogether of the matrices $\mathbf{D}^j$ by using an alternative, structure inducing penalization on the matrices $\mathbf{M}^j$. The $\ell_{1,2}$ matrix norm is an appealing alternative. Letting $\mathbf{m}_i'$ denote the transposed i-th row of $\mathbf{M}$, it is given by

$$\|\mathbf{M}\|_{1,2} = \sum_i \|\mathbf{m}_i'\|_2, \tag{6.5}$$

and this is in turn an ℓ_1 penalization of the vector $[\|\mathbf{m}_i'\|_2]_i$ of ℓ_2 norms of the rows of $\mathbf{M}$. As such, it forces entire rows of $\mathbf{M}$ to be zero. Yet in the context of learning

problems such as ours, Stochastic Gradient Descent (SGD) optimization methods that process a single example at a time are a must. An adaption of SGD is thus required that processes one example at a time while retaining nonetheless the sparse structure. As we will see next, our approach (6.4) using diagonal matrices can accomplish this with a simple adaptation of SGD-based solvers for ℓ_1 penalized SVM problems [5].

6.3.3 Learning approach

In order to derive an algorithm to minimize (6.4), we will first re-write it in a more convenient form using the following definitions:

$$r^{k,i} = \|\mathbf{w}^k\|_2^2 + Cl\left(y_i^k \left(f_J \circ \ldots \circ f_1(\mathbf{x})\right)^\top \mathbf{w}^k\right), \tag{6.6}$$

$$R_i = \frac{1}{K}\sum_{k=1}^{K} r^{k,i} + \delta \sum_{j=1}^{J} \|\mathbf{D}^j\|_1 + \mu \sum_{j=1}^{J} \|\mathbf{M}^j\|_F^2, \tag{6.7}$$

the resulting form of (6.4) is

$$\frac{1}{N}\sum_{i=1}^{N} R_i. \tag{6.8}$$

In order to minimize (6.8), we will employ a block-coordinate SGD approach wherein, for each sample i, we process each block of coordinates $\mathbf{M}^1, \mathbf{D}^1, \mathbf{b}^1, \ldots, \mathbf{M}^J, \mathbf{D}^J, \mathbf{b}^J$, $\mathbf{w}^1, \ldots, \mathbf{w}^K$ successively. Letting θ denote a generic block of coordinates, the update step for that block at time instance t is as follows, where i_t is a sample index drawn randomly at time t, and the coefficient γ_t is the learning rate chosen using cross-validation:

$$\theta_{t+1} = \theta_t - \gamma_t \left.\frac{\partial R_{i_t}}{\partial \theta}\right|_{\theta_t}. \tag{6.9}$$

The required sub-gradient $\frac{\partial R_i}{\partial \theta}$ is given by

$$\frac{\partial R_i}{\partial \theta} = \frac{1}{K}\sum_{k=1}^{K} \frac{\partial r^{k,i}}{\partial \theta} + \delta \sum_{j=1}^{J} \frac{\partial \|\mathbf{D}^j\|_1}{\partial \theta} + \mu \sum_{j=1}^{J} \frac{\partial \|\mathbf{M}^j\|_F^2}{\partial \theta} \tag{6.10}$$

Expressions for $\frac{\partial r^{k,i}}{\partial \theta}$ when θ represents a classifier $\mathbf{w}^k$ are readily available from the literature on SGD-based SVM solvers [99, 5], and expressions for the case when θ represents a weights matrix $\mathbf{M}^j$ or offset vector $\mathbf{b}^j$ are available from the literature on deep learning [65]. The gradient of the squared Frobenius norm is just a rasterization of $2\mathbf{M}^j$ (whenever θ represents the corresponding block of coordinates $\mathbf{M}^j$).

Concerning the gradient with respect to the matrices $\mathbf{D}^j$, it is possible to apply the update rule specified in (6.10) directly, but this will produce matrices $\mathbf{D}^j$ with diagonals that are only approximately sparse. Instead, we will use a procedure adapted from ℓ_1 penalized SVM solvers [5]. To this end, we represent each $\mathbf{D}^j$ in terms of two non-negative vectors $\mathbf{v}^j$ and $\mathbf{u}^j$:

$$\mathbf{D}^j = \mathrm{diag}(\mathbf{v}^j) - \mathrm{diag}(\mathbf{u}^j), \text{ where } \mathbf{v}^j, \mathbf{u}^j \geq 0. \tag{6.11}$$

In order to enforce the non-negativeness of the $\mathbf{v}^j$ and $\mathbf{u}^j$, whenever we are optimizing over one of these blocks, we will follow the SGD update step in (6.9) by a projection into the set of non-negative vectors. The modified update step is given by

$$\theta_{t+1} = \left[\theta_t - \gamma_t \left.\frac{\partial R_{i_t}}{\partial \theta}\right|_{\theta_t}\right]_+. \tag{6.12}$$

Choice of learning rate. In order to make our algorithm converge at a fast rate, we will use an adaptive learning rate that gets updated and kept fixed for each batch of B samples. The learning rate is updated at the beginning of the batch using $\gamma_t = \gamma_{t-1} \cdot 2^{-n}$, where successive values of $n = 0, 1, 2, \ldots$ are tested until further increasing n no longer reduces the cost computed over a subset of the B samples from the batch. This approach has the advantage that the subset used to choose n is small, and hence adaptation is fast.

Early-stopping and choice of regularization parameters. We used two different early-stopping strategies that allowed us to select the number of iterations T (expressed as number of epochs) to use, both based on cross-validation. In one case we used the iteration that produced the highest mean Average Precision (mAP) over the validation set, while in the second case we chose the iteration giving the lowest cost, again computed over the validation set. We likewise use the validation set to choose the regularization parameters.

6.4 Results

In this section we evaluate our proposed learning algorithm and compare it against various state-of-the-art algorithms. To this end, we use the `Pascal-VOC-2007` dataset, which consists of 4192 test images and 5011 training images. We hold out 811 training images, choosing them uniformly over all classes, and use them as a validation set. As an image representation, we will use the VGG-128 model of [10] which produces 128-dimensional features, accordingly using weights matrices of size 128×128.

Choice of regularization values Our learning algorithm is governed by three different regularization methods: the penalty weights μ and δ, and the number of training epochs T. The number of training epochs is determined using the validation set by computing either the *(i)* the validation mAP or *(ii)* the validation cost and choosing the number of training epochs that gives the best value. We found that using the validation mAP to determine T resulted in higher test mAP, but diagonal matrices $\mathbf{D}^j$ that are not necessarily sparse. Using the validation cost to determine T, on the other hand, resulted in slightly lower mAP values but in much higher sparsities for the matrices $\mathbf{D}^j$.

In Fig. 6.2 we illustrate the cross-validation method we use to choose the penalty weights μ and δ. The approach consists of keeping one penalty weight fixed while varying the second one, and then choosing the penalty weight that results in the highest validation mAP (this corresponded roughly to the value producing the highest test mAP).

Varying the number of layers J In Table 6.1, we evaluate the performance of our method as a function of the number of layers J in the architecture. We choose the regularization values following the procedure outlined above, using the validation cost to select the stopping point T. Note that our method succeeds in choosing sparse matrices $\mathbf{D}^j$ while retaining a high test mAP. For $J > 2$, only the first layer results in a matrix $\mathbf{D}^j$ with sparse diagonal. The reason for this is that the cross-validation procedure used to select δ in (6.7) only takes the mAP into account.

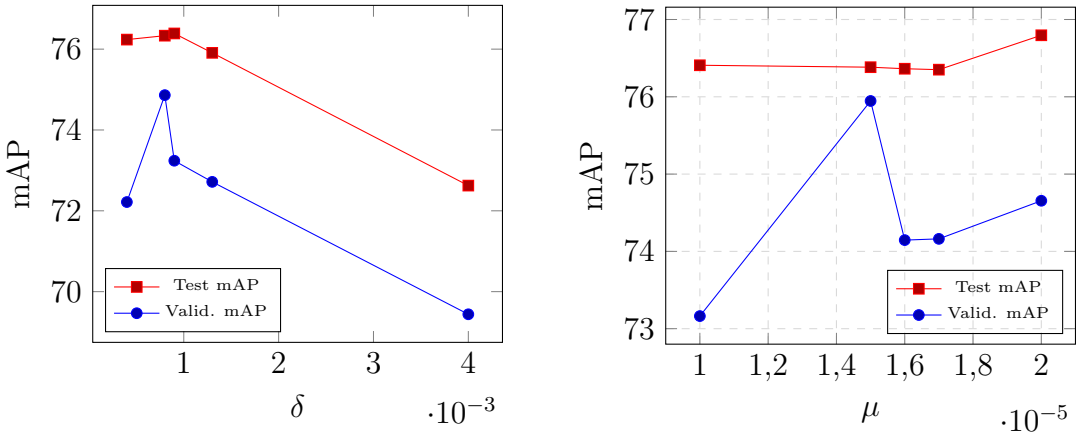


Figure 6.2 – Effect of the penalty weights δ and μ for a single-layer architecture, using validation cost as a stopping criterion.

J	Test mAP	Number of zeros in diagonal of $\mathbf{D}^j$						δ	μ	T
		$j=1$	2	3	4	5	6			
1	76.38	93	-	-	-	-	-	8.0e-4	1.5e-5	120
2	76.20	105	105	-	-	-	-	1.4e-3	3.5e-5	180
3	76.58	92	0	0	-	-	-	8.0e-4	1.0e-5	240
5	76.55	91	0	0	0	0	-	8.0e-4	2.5e-5	360
6	76.19	88	0	0	0	0	0	7.0e-4	1.4e-5	420

Table 6.1 – Using validation cost to choose the number of training epochs T, and validation mAP to choose the best δ first, and then the best μ.

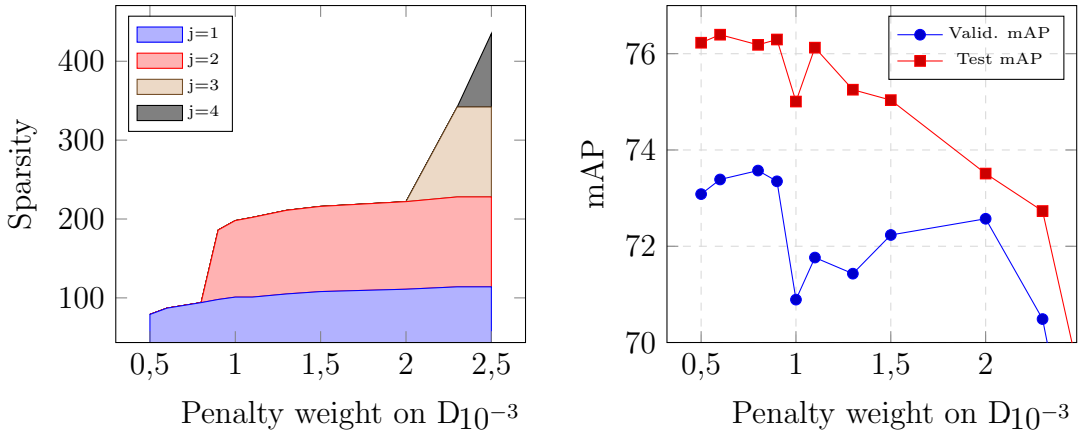


Figure 6.3 – Effect of the penalty weight δ on *(left)* the number of zero diagonal entries of $\mathbf{D}^j$, $j = 1, \ldots, 4$, and *(right)* on the classification performance as measured by mAP. The zero diagonal entries are presented as stacked plots so that the vertical displacement of any shaded regions corresponds to the number of zero diagonal entries of $\mathbf{D}^j$ for the corresponding layer.

In Fig. 6.3, we hence plot both the sparsity for all layers and the corresponding test and validation mAPs when varying the penalty weight δ. Note that increasing δ drastically increases the number of zero diagonal entries in the architecture while only slightly affecting the classification performance. For example, for $\delta = 2.5e^{-3}$, close to 82% of the diagonal entries in all $\mathbf{D}^j$ are zero, while the test mAP has only dropped

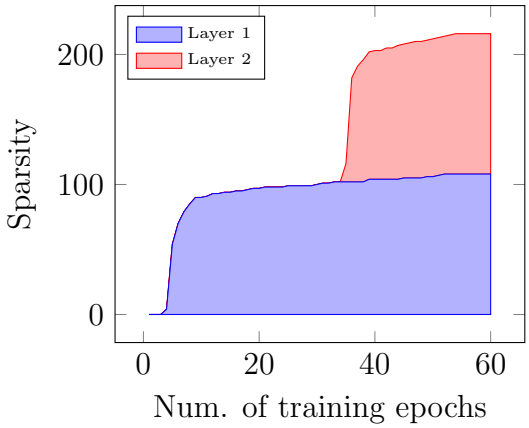


Figure 6.4 – Number of zero entries in diagonal of $\mathbf{D}^j$ versus iteration number (expressed as number of epochs) for an architecture of $J = 2$ layers.

by 4.5%. For $\delta = 8e{-}4$, close to 40% of the diagonal entries are zero, while the system mAP is nearly unaffected.

For completeness, in Fig. 6.4 we present a plot of layer sparsity as a function of the training epoch. Note that $\mathbf{D}^1$ becomes sparse initially, and this reduces the intrinsic dimensionality of the signals at the input of the second layer, hence enabling $\mathbf{D}^2$ to become sparse.

Comparison against state-of-the-art In Table 7.5.4 we compare our proposed method against various state-of-the-art algorithms derived from CNN methods. We include four reference methods that rely on data augmentation (the first four), and two reference methods that do not (the next two). The last three lines in the table correspond to three variants of our architecture. The first two have depth $J = 1, 3$ and are learnt with sparsity in mind by using a stopping criterion T selected using validation cost. The last one is learned using validation mAP to select the stopping criterion T. Note that, for $J = 1, 3$, our method produces very small features of size 35 and 36, respectively, while at the same time outperforming all the reference methods not-relying on augmentation, even when these use features many times larger. Our architecture with $J = 6$ results in a mAP value that outperforms not only the non-augmented references approaches, but also two of the four approaches relying on augmentation.

Method	Train time	Dim	# params.	mAP
CNN S TUNE-RNK_aug[10]	-	4K	∼100M	82.42
VGG-128_aug [10]	∼ 10s	128	2560	78.90
off-the-shelf_aug[100]	-	4K	81K	77.2
PRE1000C [79]	∼ 1 day	-	∼8.5M	77.73
VGG-128 [10]	∼ 10s	128	2560	76.34
off-the-shelf[101]	-	4K	81K	73.9
Ours($J = 1$)	210s	35	7040	76.38
Ours($J = 3$)	630s	36	9760	76.58
Ours($J = 6$)	1260s	128	100K	77.63

Table 6.2 – Comparison of our proposed method with various existing CNN methods. The training time indicated includes only the training time related to Pascal VOC, and not training time incurred when learning on ImageNet. The top four methods rely on some form of data augmentation and have training sets that effectively many times bigger than the PascalVOC training set. The bottom five methods (including ours) only use the training images specified in PascalVOC.

6.5 Conclusion

We present a method that automatically selects the size of the weight matrices inside fully-connected layers comprising a deep architecture. Our approach relies on a regularization penalty term consisting of the ℓ_1 norm of the diagonal entries of diagonal matrices inserted between the fully-connected layers. Using such a penalty term forces the diagonal matrices to be sparse, accordingly selecting the effective number of rows and columns in the weights matrices of adjacent layers. We present a simple algorithm to solve the proposed formulation and demonstrate it experimentally on a standard image classification benchmark.

Chapter 7

SPLeaP: Soft Pooling of Learned Parts for Image Classification

Contents

7.1 Introduction

This chapter addresses the problem of image classification with Part-Based Models (PBMs). Decomposing images into salient parts and aggregating them to form discriminative representations is a central topic in the computer vision literature. It is raising several important questions such as: How to find discriminative features? How to detect them? How to organize them into a coherent model? How to model the variation in the appearance and spatial organization? Even if works such as the pictorial structure [33], the constellation model [128], object fragments [118], the Deformable Part Model [31] or the Discriminative Modes Seeking approach of [22] brought interesting contributions, as well as those in [107, 53, 23], the automatic discovery and usage of discriminative parts for image classification remains a difficult and open question.

Recent PBMs for image classification *e.g*, [22, 107, 53, 22, 85] rely on five key components: (i) The generation of a large pool of candidate regions per image from (annotated) training data; (ii) The mining of the most discriminative and representative regions from the pool of candidate parts; (iii) The learning of part classifiers using the mined parts; (iv) The definition of a part-based image model aggregating (independently) the learnt parts across a pool of candidate parts per image; (v) The learning of final image classifiers over part-based representations of training images.

One key challenge in the 2nd and 3rd components of PBMs lies in the selection of discriminative regions and the learning of interdependent part classifiers. For instance, one cannot learn the part classifiers before knowing discriminative regions and vice-versa. Extensive work has been done to alleviate the problem of identifying discriminative regions in a huge pool of candidate regions, *e.g*, [53, 23, 22].

Once the discriminative regions are discovered and subsequently part classifiers are trained, the 4th component in a PBM – *i.e.*, the construction of the image model based on the per image part presence – is basically obtained by average or sum pooling of part classifier responses across the pool of candidate regions in the image. The final classifiers are then learnt on top of this part-based image representation. Although the aforementioned methods address one of the key components of PBMs, *i.e.*, mining discriminative regions by using some heuristics to improve final classification, they fail to leverage the advantage of jointly learning all the components together.

The joint learning approach of all components of PBMs is indeed particularly appealing since the discriminative regions are explicitly optimized for the targeted task. But intertwining all components makes the problem highly non-convex and initialization critical. The recent works of Lobel *et al.*[73] and Parizi *et al.*[85] showed that the joint learning of a PBM is possible. However, these approaches suffer from several limitations. First, their intermediate part classifiers are simple linear classifiers and the expression power of these part classifiers is limited in capturing complex patterns in regions. Furthermore, they pool the part classifier responses over candidate regions per image using max pooling which is suboptimal [47]. Finally, as the objective function is non-convex they rely on a strong initialization of the parts.

In the present work, we propose a novel framework, coined "Soft Pooling of Learned Parts" (SPLeaP), to jointly optimize all the five components of the proposed PBM. A first contribution is that we describe each part classifier as a linear combination of weak non-linear classifiers, learned greedily and resulting in a strong classifier which is non-linear. This greedy approach is inspired by [77, 34] wherein they use gradient descent for choosing linear combinations of weak classifiers. The complexity of the part detector is increased along with the construction of the image model. This classifier is eventually

able to better capture the complex patterns in regions. A second contribution is that we softly aggregate the computed part classifier responses over all the candidate regions per image. We introduce a parameter, referred as the "pooling parameter", for each part classifier independently inside the optimization process. The value of this pooling parameter determines the softness level of the aggregation done over all candidate regions, with higher softness levels approaching sum pooling and lower softness levels resembling max pooling. This permits to leverage different pooling regimes for different part classifiers. It also offers an interesting way to relax the assignment between regions and parts and lessens the need for strong initialization of the parts. The outputs of all part classifiers are fed to the final classifiers driven by the classifier loss objective.

The proposed PBM can be applied to various visual recognition problems, such as the classification of objects, scenes or actions in still images. In addition, our approach is agnostic to the low-level description of image regions and can easily benefit from the powerful features delivered by modern Convolutional Neural Nets (CNNs). By relying on such representations, and outperforming [79, 10], the proposed approach can also be seen as a low-cost adaptation mechanism: pre-trained CNNs features are fed to a mid-to-high level model that is trained for a new target task. To validate this adaptation scheme we use the pre-trained CNNs of [10]. Note that this network is not fine-tuned on target datasets.

We validated our method on three challenging datasets: `Pascal-VOC-2007` (object), `MIT-Indoor-67` (scenes) and `Willow` (actions). We improve over state-of-the-art PBMs on the three of them.

The rest of the chapter is organized as follows. The next section presents a review of the related works, followed by the presentation of the method in Section 7.3. Section 7.4 describes the algorithm proposed to jointly optimize the parameters, while Section 7.5 contains the experimental validation of our work.

7.2 Related works

Most of the recent advances on image classification are concentrated on the development of novel Convolutional Neural Networks (CNNs), motivated by the excellent performance obtained by Krizhevsky *et al.*[59]. As CNNs require huge amount of training data (*e.g*, `ImageNet`) and are expensive to train, some authors such as Razavian *et al.*[90] showed that the descriptors produced by CNNs *pre-trained* on a large dataset are generic enough to outperform many classification tasks on diverse small datasets, with reduced training cost. Oquaba *et al.*[79] and Chatfield *et al.*[10] were the first to leverage the benefit of fine-tuning the pre-trained CNNs to new datasets such as `Pascal-VOC-2007` [27]. Oquab *et al.*[79] reused the weights of initial layers of CNN pre-trained on `ImageNet` and added two new adaptation layers. They trained these two new layers using multi-scale overlapping regions from `Pascal-VOC-2007` training images, using the provided bounding box annotations. Chatfield *et al.*[10], on the other hand, fine-tuned the whole network to new datasets, which involved intensive computations due to the large number of network parameters to be estimated. They reported state-of-art performance on `Pascal-VOC-2007` till date by fine-tuning pre-trained CNN architecture.

In line with many other authors, [101, 10] utilized the penultimate layer of CNNs to obtain global descriptors of images. However, it has been observed that computing and aggregating local descriptors on multiple regions described by pre-trained CNNs provides an even better image representation and improves classification performance.

Methods such as Gong *et al.*[40], Kulkarni *et al.*[60] and Cimpoi *et al.*[13] relied on such aggregation using standard pooling techniques, *e.g*, VLAD, Bag-of-Words and Fisher vectors respectively.

On the other hand, Part-Based Models (PBMs) proposed in the recent literature, *e.g*, [107, 22, 53, 23], can be seen as more powerful aggregators compared to [40, 50, 60]. PBMs attempt to select few relevant patterns or discriminative regions and focus on them in the aggregation, making the image representation more robust to occlusions or to frequent non-discriminative background regions.

PBMs differ in the way they discover discriminative parts and combine them into a unique description of the image. The Deformable Part Model proposed by Felzenszwalb *et al.*[31] solves the aforementioned problems by selecting discriminative regions that have significant overlap with the bounding box location. The association between regions and part is done through the estimation of latent variables, *i.e.*, the positions of the regions w.r.t. the position of the root part of the model. Differently, Singh *et al.*[107] aimed at discovering a set of relevant patches by considering the representative and frequent enough patches which are, in addition, discriminative w.r.t. the rest of the visual world. The problem is formulated as an unsupervised discriminative clustering problem on a huge dataset of image patches, optimized by an iterative procedure alternating between clustering and training discriminative classifiers. More recently, Juneja *et al.*[53] also aimed at discovering distinctive parts for an object or scene class by first identifying the likely discriminative regions by low-level segmentation cues, and then, in a second time, learning part classifiers on top of these regions. The two steps are alternated iteratively until a convergence criterion based on Entropy-Rank is satisfied. Doersch *et al.*[22] used density based mean-shift algorithms to discover discriminative regions. Starting from a weakly-labeled image collection, coherent patch clusters that are maximally discriminative with respect to the labels are produced, requiring a single pass through the data.

Contrasting with previous approaches, Li *et al.*[70] were among the first to rely on CNN activations as region descriptors. Their approach discovers the discriminative regions using association rule mining techniques, well-known in the data mining community. Sicre *et al.*[104] also build on CNN-encoded regions, introducing an algorithm that models image categories as collections of automatically discovered distinctive parts. Parts are matched across images while learning their visual model and are finally pooled to provide images signatures.

One common characteristic of the aforementioned approaches is that they discover the discriminative parts first and then combine them into a model of the classes to recognize. There is therefore no guaranty that the so-learned parts are optimal for the classification task. Lobel *et al.*[73] showed that the joint learning of part and category models was possible. More recently, Parizi *et al.*[85] build on the same idea, using max pooling and l_1/l_2 regularization.

Variour authors have likewise studied learned soft-pooling mechanisms. Gulcehre *et al.*[43] investigate the effect of using generalized soft pooling as a non-linear activation unit, bearing some similarity with the maxout non-linear unit of [42]. In contrast, our method uses a generalized soft pooling strategy as a down sampling layer. Our method is close to that of Lee *et al.*[66], who use linear interpolation of max and average pooling. Our approach, on the other hand, uses a non-linear interpolation of these two extrema.

7.3 Proposed Approach

Our goal is to represent each category involved in the visual recognition problem of interest as a collection of discriminative regions. These regions are automatically identified using learned part classifiers, that will operate on a pool of proposed fragments. A "part" classifier is meant to capture specific visual patterns. As such it does not necessarily capture a strong, human understandable semantic: it might respond highly on more than one region of the given image or, conversely, embrace at once several identifiable parts of the object. On images from "horse" class for instance, one part classifier might focus on the head of the animal when another one turns out to capture a large portion of the horse body.

Formally, we consider an image as a bag of R regions, each one equipped with a descriptor $\mathbf{x}_r \in \mathbb{R}^D$. The image is thus represented at first by the descriptor collection $\mathcal{X} = \{\mathbf{x}_r\}_{r=1}^R$. The number of regions will be image-dependent in general even if we assume it is not for notational convenience.

Based on training images spanning C images categories, P "part" classifiers will be learned, each as a weighted sum of K base classifiers applied to a region's descriptor (K chosen by cross-validation). The score of the p-th part classifier for a given descriptor $\mathbf{x}$ is defined as:

$$H_p(\mathbf{x}; \boldsymbol{\theta}_p) = \sum_{k=1}^{K} a_k^p \sigma(\mathbf{x}^\top \mathbf{u}_k^p + b_k^p), \tag{7.1}$$

where σ is the sigmoid function, a_k^p is the weight of the k-th base classifier, $\mathbf{u}_k^p \in \mathbb{R}^D$ and $b_k^p \in \mathbb{R}$ are its parameters and $\boldsymbol{\theta}_p = \mathrm{vec}(a_{1:K}^p, \mathbf{u}_{1:K}^p, b_{1:K}^p) \in \mathbb{R}^{K(D+2)}$ is the vector of all the parameters that define the part classifier. This score is aggregated over the pool of R regions a follows:

$$f_p(\mathcal{X}) = \sum_{r=1}^{R} \pi_r^p H_p(\mathbf{x}_r; \boldsymbol{\theta}_p), \tag{7.2}$$

where normalized weights are defined as

$$\pi_r^p \propto \exp\Big(\beta_p H_p(\mathbf{x}_r; \boldsymbol{\theta}_p)\Big), \ \sum_{r=1}^{R} \pi_r^p = 1, \tag{7.3}$$

with β_p a part-dependent "pooling" parameter. For large values of this parameter the scores are max-pooled, while they are averaged for very small values.

Given a set of part classifiers with parameter $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1 | \cdots | \boldsymbol{\theta}_P]$ and associated pooling parameters $\boldsymbol{\beta} = [\beta_p]_{p=1}^P$, the bag of R region descriptors $\mathcal{X} = \{\mathbf{x}_r\}_r$ attached to an input image is turned into a part-based description:

$$\mathbf{f}(\mathcal{X}; \boldsymbol{\Theta}, \boldsymbol{\beta}) = [f_p(\mathcal{X})]_{p=1}^P. \tag{7.4}$$

The multiclass classification problem at hand is cast on this representation. Resorting to logistic regression, we aim at learning P-dimensional vectors, $\mathbf{w}_c = [w_1^c \cdots w_P^c]^\top \in \mathbb{R}^P$, one per class, so that the class label $y \in \{1 \cdots C\}$ of an input image $\mathcal{X}$ is predicted according to distribution

$$\Pr(y = c | \mathcal{X}; \boldsymbol{\Theta}, \boldsymbol{\beta}, W) = \frac{\exp\Big(\mathbf{w}_c^\top \mathbf{f}(\mathcal{X}; \boldsymbol{\Theta}, \beta)\Big)}{\sum_{d=1}^{C} \exp\Big(\mathbf{w}_d^\top \mathbf{f}(\mathcal{X}; \boldsymbol{\Theta}, \beta)\Big)}, \tag{7.5}$$

where $\mathbf{W} = [\mathbf{w}_1|\cdots|\mathbf{w}_C]$. For simplicity in notation, we have omitted the bias term associated with each class. In practice, we append each of them to the corresponding vectors $\mathbf{w}_c$s and entry one is appended to descriptor $\mathbf{f}(\mathcal{X}; \boldsymbol{\Theta}, \beta)$.

Discriminative learning is conducted on annotated training dataset $\mathcal{T} = \{(\mathcal{X}_n, y_n)\}_{n=1}^N$, with $\mathcal{X}_n = \{\mathbf{x}_r^n\}_{r=1}^R$ and $y_n \in \{1, \ldots, C\}$. Part-level and category-level classifiers are jointly learned by minimizing a regularized multiclass cross entropy loss:

$$\min_{\boldsymbol{\Theta},\boldsymbol{\beta},\mathbf{W}} -\sum_{n=1}^{N}\sum_{c=1}^{C}[y_n = c] \ln \Pr(c|\mathcal{X}_n; \boldsymbol{\Theta}, \boldsymbol{\beta}, W) + \mu\|\boldsymbol{\Theta}\|_F^2 + \delta\|\mathbf{W}\|_F^2, \tag{7.6}$$

where [.] is Iverson bracket. The two regularization weights μ and δ, the number P of part classifiers and the number K of base learners in each part are set by cross-validation. Learning is done by block-wise stochastic gradient descent, as explained next into more details.

The multi-class loss in (7.6) being based on softmax (7.5), it requires that each image in the training set is assigned to a single class. If this is not the case, one can use instead a one-vs-all binary classification approach, which can be easily combined as well with the proposed PBM.

7.4 Optimization specific details

In this section we provide details on how the joint optimization problem (7.6) is addressed. It aims at learning the final category-level classifiers (defined by $\mathbf{W}$), the part classifiers (defined by $\boldsymbol{\Theta}$) and the part-dependent pooling coefficients in $\boldsymbol{\beta}$. By conducting jointly these learnings, part classifiers are optimized for the target recognition task. Additionally, learning part-specific parameter β_p enables to accommodate better the specifics of each part by adapting the softness of its region pooling.

Algorithm 1 summarizes the different steps of the optimization. In Alg. 1, we denote $\boldsymbol{\theta}_{(k)}$ the vector of parameters associated to k-th base classifiers in matrix $\boldsymbol{\Theta}$, that is $\boldsymbol{\theta}_{(k)} = \text{vec}(a_k^{1:P}, \mathbf{u}_k^{1:P}, b_k^{1:P})$ and $\mathcal{L}_n = \log \Pr(y_n|\mathcal{X}_n; \boldsymbol{\Theta}, \boldsymbol{\beta}, W)$ the log-likelihood of n-th training pair $(\mathcal{X}_n, y_n)$.

We perform E epochs of block-coordinate stochastic gradient descent. If part-related parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\beta}$ were known and fixed, the optimization of image classifiers $\mathbf{W}$ alone in the proposed algorithm would amount to the classic learning of logistic regressors on image descriptors $\mathbf{f}(\mathcal{X})$ defined in (A.5). The interleaved learning of the P part-classifiers defined by $\boldsymbol{\Theta}$ is more involved. It relies on a stage-wise strategy whereby base classifiers are progressively incorporated. More precisely, we start with a single weak classifier per part, randomly initialized and optimized over the first S epochs. Past this first stage with training a single weak classifier, each part-classifier is then allowed an additional weak classifier per epoch. With initialization to zero of the parameters of this new learner, non-zero gradients for these parameters is produced by training samples that were previously misclassified. Note that at each epoch, only the last weak classifier is updated for each part while previous ones are kept fixed.

We set all algorithm's parameters (number P of parts, number K of weak classifiers per part, number S of epochs with part classifiers based only on a single weak learner, learning rates γ_W, γ_θ and γ_β) through careful cross-validation.

Algorithm 1 SPLeaP Training: joint part-category classifier learning

procedure LEARN($\mathcal{T}$)
  parameters: P, K, μ, δ, S, γ_W, γ_θ, γ_β
  $W \leftarrow 0$
  $\boldsymbol{\theta}_{(1)} \leftarrow$ `rand()`
  $\boldsymbol{\theta}_{(2:K)} \leftarrow 0$
  $\boldsymbol{\beta} \leftarrow$ `rand()`
  $k \leftarrow 1$
  for $e = 1$ to $E = K + S - 1$ **do**
    $\mathcal{T} \leftarrow$ *RandomShuffle*($\mathcal{T}$)
    for $n = 1$ to N **do**
      $\mathbf{W} \leftarrow (1 - \gamma\mathbf{w})\mathbf{W} + \gamma\mathbf{w} \sum_{(\mathfrak{X}_n, y_n) \in \mathcal{T}} \nabla\mathbf{w}\mathcal{L}_n$
      $\boldsymbol{\theta}_{(k)} \leftarrow (1 - \gamma_\theta)\boldsymbol{\theta}_{(k)} + \gamma_\theta \sum_{(\mathfrak{X}_n, y_n) \in \mathcal{T}} \nabla_{\boldsymbol{\theta}_{(k)}}\mathcal{L}_n$
      $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \gamma_\beta \sum_{(\mathfrak{X}_n, y_n) \in \mathcal{T}} \nabla_{\boldsymbol{\beta}}\mathcal{L}_n$
    end for
    if $e > S$ **then**
      $k \leftarrow k + 1$
    end if
  end for
  Return $\mathbf{W}, \boldsymbol{\Theta}, \boldsymbol{\beta}', \boldsymbol{\beta}''$
end procedure

7.5 Results

7.5.1 Experimental settings

We evaluate our proposed method using three well-known datasets: `Pascal-VOC-2007`,`MIT-Indoor-67` and `Willow`.

Region proposal schemes.

We explored three different strategies to extract the pool of region proposals from each image:

Selective Search (SS). We use the selective search region proposal scheme of [120] to extract between 100 and 5000 region proposals per image, with an average of 800, using Matlab code provided by [12].

Augmentation (aug). Following the data augmentation technique of [10], we derive ten images from each input image by taking one center crop and four corner crops from the original image and further mirroring each crop vertically. The ten resulting modified image crops are used as region proposals.

Selective search + augmentation (SS + aug). We also explore merging the outputs of the two previous strategies into a single pool of region proposals.

Region feature extraction.

From each of the candidate regions obtained using one of the above described region proposal methods, we extract one feature vector consisting of the activation coefficients of the previous-to-last layer of several state-of-the-art CNN architectures. The CNN architectures we consider, available in CAFFE [52], are *(i)* the 128-dimensional feature extracted from the 13-layer architecture of [10] (VGG-128), *(ii)* the 16-layer architecture of [106] producing 4096 dimensional features (VD-16) and *(iii)* the architecture of [137] corresponding to Krizhevsky's architecture [59] pre-trained using `ImageNet` (978 categories) and the `Places` database (HybridCNN).

Cross-validation of hyper-parameters.

We use Stochastic Gradient Descent (SGD) to train our model. The performance of the model depends on the value of the various hyper-parameters: the number of parts P and of weak learners in each part classifier K, the regularization weights μ and δ in (7.6), the number of epochs E and the various learning rates (see Algorithm 1). For the `Pascal-VOC-2007` and `Willow` datasets, we use piecewise-constant learning rates decreased every ten epochs empirically, similarly to the appraoch of [10]. For the `MIT-Indoor-67` dataset, we use learning rates of the form $\gamma(i) = \frac{\gamma_0}{1.0+\lambda i}$, where γ_0 and λ are hyper parameters that are cross-validated.

We select the values of these hyper-parameters using cross-validation. After the cross-validation phase, the hyper-parameters are set accordingly and the training and validation data are merged to re-train our model.

7.5.2 Experimental validation of the contributions

We now establish experimentally the benefits of our main contributions: weakly supervised parts learning, soft-max pooling with learned, per-part softness coefficients, and part detectors based on weak learners. To this end, we use the `Pascal-VOC-2007` dataset along with the mean Average Precision (mAP) performance measure specified by the dataset's authors, using VGG-128 features to represent all region proposals.

Comparison with unsupervised aggregation.

In Table 7.1, we first verify that the improvements of our method are not due to simply the region proposal strategies we employ. We hence compare our supervised SPLeaP method to three analogous baseline features not employing supervised learning. The first baseline, denoted *VGG-128-G*, uses the global feature vector extracted from the whole image. The second baseline, denoted *VGG-128-sum*, aggregates VGG-128 features extracted from each candidate region using average pooling, similarly to an approach used in [61]. Both of these baselines result in 128-dimensional feature vectors. In a third baseline, denoted *VGG-128-K-means*, we perform K-means on all candidate regions from all images in the database to obtain $P = 40$ centroids. Computing an image feature then consists of selecting the image's $P \ll R$ candidate region whose features are closest to the P centroids and concatenating them into a single vector of size $128P$.

For each of the aforementioned feature construction methods, the resulting image feature vectors are ℓ_2-normalized and then used to learn linear SVMs using a one-vs-rest strategy.

Table 7.1 – Comparison against unsupervised aggregation baselines

VGG-128-G	VGG-128-sum			VGG-128-K-means		SPLeaP	
	SS	aug	SS+aug	SS	SS+aug	SS	SS+aug
75.32	77.31	78.21	77.36	76.28	76.8	84.21	**84.68**

Table 7.2 – Importance of per-part softness coefficients

Average pooling	Max pooling	Cross-valid. $\beta_p = \beta$	Learned β_p
80.77	83.23	84.31	**84.68**

The results in Table 7.1 establish that large performance gains (more than 8 mAP points) are obtained by proposed SPLeaP method relative to the different baseline aggregation strategies, and hence the gain does not follow simply from using our region proposal strategies. Interestingly, contrary to the baseline strategies, our method succeeds in exploiting the merged *SS+aug* region proposal strategy (0.47 mAP improvement relative to *SS*).

Importance of per-part softness coefficient.

In Table 8.1, we evaluate our proposed soft-max pooling strategy in (7.3) that employs a learned, per-part softness coefficient β_p. We compare per-part softness coefficients to three alternatives: *(i)* average pooling, wherein $\forall p, \beta_p = 0$; *(ii)* max pooling, which is equivalent to $\forall p, \beta_p \to \infty$; and *(iii)* a cross-validated softness coefficient that is constant for all parts, $\forall p, \beta_p = \beta$. In all three of these alternatives, we run the complete SPLeaP optimization process discussed in Section 7.4. As illustrated in the table, using our proposed learned, per-part softness coefficient yields the best performance, with an improvement of close to 4 mAP points over average pooling, 1.5 mAP points over max pooling, and 0.4 mAP points over a cross-validated but constant softness coefficient. Note that allowing the algorithm to choose β_p during the optimization process eliminates the need for a costly cross validation of the β_p.

Effect of number of weak learners K.

In Fig. 7.1 we evaluate the effect of the number K of weak learners per part by plotting mAP as a function of the number of training epochs. Note that, for a fixed number of learning iterations, adding more weak learners results in higher performance. We have tried the effect of other design choices such as averaging K weak learners in contrast to greedily adding the weak learners. We obtain slight improvement *i.e.*we obtain 84.78 mAP for $K = 3$. We also compared adding weak learners to dropout, which is known to behave as averaging multiple thinned networks, and obtained a reduction in mAP of 0.5% (83.98 mAP with 50 % dropout).

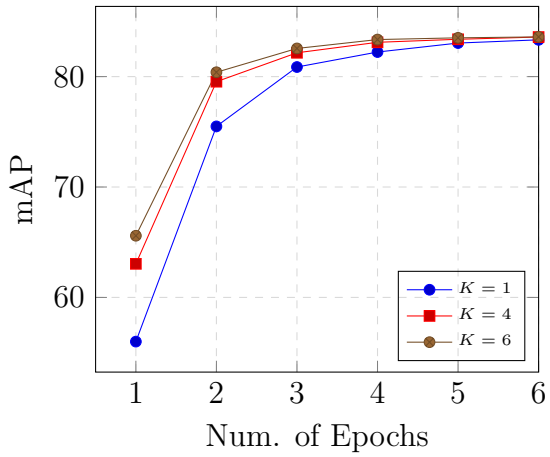


Figure 7.1 – Plot of test mAP versus number of training epochs.

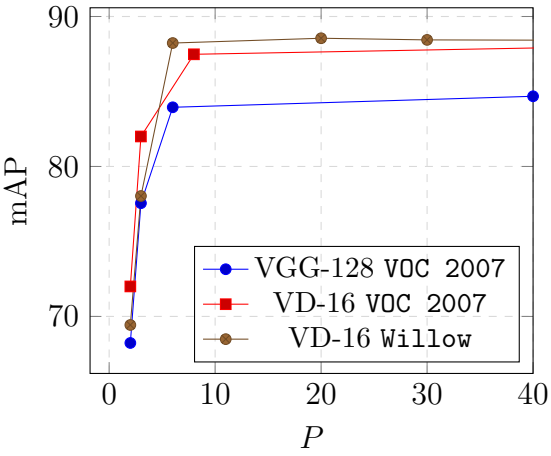


Figure 7.2 – Plot of test mAP versus the number of parts P.

Table 7.3 – Comparison of results on `Pascal-VOC-2007` dataset ($P = 40$ parts per class, $K = 1$) using CNN features extracted from (*left)* Krizhevsky-like [59] and (*right)* very deep architectures [106]

Methods	*mAP*
VGG-G	75.35
Oquab *et al.* [79]	77.31
Li *et al.* [70]	77.90
Cimpoi *et al.* [13]	79.50
CNN-S fine tuned [10]	82.42
SPP [46]	82.44
SPLeaP-VGG-128 (*SS+ext. aug*)	**84.68**

Methods	*mAP*
VD-16-G [106]	81.73
VD-16 (*dense_evaluation*)	84.67
VD-16-sum (*SS+ext. aug*)	82.58
Cimpoi *et al.* [13]	85.10
SPLeaP-VD-16 (*SS+ext. aug*)	**88.01**

7.5.3 Parameters/Design related choices

Per-category parts and number P of parts.

When learning SPLeaP for the `MIT-Indoor-67` dataset, we learn P part classifiers that are common to all 67 categories using the multi-class objective described in Section 7.3. For `Willow` and `Pascal-VOC-2007`, on the other hand, we learn P different part classifiers for each category, using a one-vs-rest strategy to learn each SPLeaP model independently for each class.

In Fig. 7.2 we evaluate mAP on `Pascal-VOC-2007` as a function of the number of parts P. We show that even with a small number of parts $P = 6$ per class, we obtain a very good mAP of 83.94.

7.5.4 Comparisons with state-of-the-art

Pascal-VOC-2007.

In Table 7.3 we compare SPLeaP to various existing state-of-the-art methods on the `Pascal-VOC-2007` dataset.

Methods employing Krizhevsky-type architectures. On the left side of Table 7.3, we compare against Krizhevsky's original 13-layer architecture [59] and variants thereof such as VGG-128 [10]. In particular, the architectures of [10, 79] were first learned on `ImageNet` and subsequently fine-tuned specifically for `Pascal-VOC-2007`.

Note that, when using architectures derived from [59], including architectures fine-tuned specifically for `Pascal-VOC-2007`, our method outperforms all of these baselines by at least 3 absolute mAP points, despite using the 128-dimensional VGG-128 feature that is not fine-tuned for `Pascal-VOC-2007`. In particular, our method outperforms the recent, part-based representation of [70], which is a state-of-the-art part-based method employing association rule mining to discover discriminative patterns/regions. In Table 7.3 we present their results based on features from [59].

Methods employing very deep architectures. On the right side of Table 7.3, we compare against the deep pipelines of Simonyan *et al.*[106], using the pre-computed models provided by the authors in [52] to reproduce the baselines. We use the state-of-the-art VD-16 feature to reproduce three different baselines using our own implementations.

The first one (VD-16-G) uses a global VD-16 feature by feeding the entire image to the CNN architecture.

The second one, VD-16 *dense_evaluation*, follows [106] in employing their CNN architecture as a fully convolutional architecture by treating the weights of the last two fully connected layers as 7×7 and 1×1 convolutional kernels, respectively. This enables them to process images of arbitrary size. The approach further employs scaling, cropping and flipping to effectively produce a pool of close to 500 region proposals that are subsequently average-pooled. The resulting descriptor is ℓ_2 normalized and used to compute linear SVMs, and achieves state-of-the-art results on `Pascal-VOC-2007`.[1]

For completeness, we further explore a third baseline that employs the extended augmentation (*ext. aug.*) strategy employed by [116], which effectively produces 144 crops per image, as opposed to the 10 crops of the *aug* strategy discussed above. We further extend this region proposals by the selective search region proposals and employ sum pooling.

The results, summarized in Table 7.3, show that proposed SPLeaP system outperforms all three baselines, and further outperforms a very recent baseline [13] relying on a hybrid bag-of-words / CNN scheme.

Willow action dataset.

Our best results on `Willow` (Table 7.4 left) likewise outperforms VD-16-G by 3.35 mAP points and VD-16 *dense_evaluation* (Table 7.4 left) by 2.8 mAP points. For completeness, we have included several, previously-published results. To our knowledge, our approach outperforms the highest published results on this dataset.

MIT-Indoor-67.

In Table 7.4, we present results on the `MIT-Indoor-67` dataset. For this dataset, we represent candidate regions using the Hybrid CNN model of [137], which is learned on a training set obtained by merging `ImageNet` and the `Places` database [137] and is better suited for scene recognition. Given the large size (4096) of these features, we

[1] Our own implementation of this method achieves results below those reported in [106].

Table 7.4 – Comparison of results on the `Willow` dataset ($P = 7$ parts per-class, $K = 1$) *(left)* and the `MIT-Indoor-67` dataset ($P = 500$ parts, common to all classes, $K = 2$) *(right)*

Methods	*mAP*
Khan *et. al*[57]	70.10
Sharma *et. al*[102]	65.90
Sharma *et. al*[103]	67.60
Sicre *et. al*[104]	81.90
VD-16-G	85.12
VD-16 (dense_evaluation)	85.67
SPLeaP (*SS*+*aug*)	**88.47**

Methods	*mAP*
Orderless[40]	68.80
MLPM[70]	69.69
HybridCNN-G[137]	72.54
HybridCNN-sum[137]	70.36
Parizi *et. al*[85]	73.30
SPLeaP-PCA160 (*SS*)	**73.45**

reduce them to size 160 using PCA, similarly to the approach of [85]. Note that our method outperforms all other methods in Table 7.4. Unlike our reported results, those of [85] use a spatial pyramid with two scales and five cells (1×1, 2×2), as well as a different number of parts and PCA-reduction factor, resulting in features that are 3.73 times bigger than ours.

7.6 Qualitative Analysis

We now present qualitative results to illustrate the response of our learned part classifiers on `Pascal-VOC-2007` test examples.

In Fig. 7.3 we demonstrate the selectivity of our part detectors by presenting image triplets consisting, in order, of *(i)* the image with candidate region bounding boxes superposed, *(ii)* the original image, and *(iii)* heatmaps for the part responses of each candidate region. Note in particular the selectivity of our part detectors: in all examples, the actual object occupies but a small fraction of the image area.

In Fig. 7.4, we illustrate the highest ranking candidate regions from all images for the part classifiers associated to the largest entries in the corresponding weight vector $\mathbf{w}_c$, with each row of each group of images corresponding to a different part classifier. Note that the part classifiers all become specialized to different object parts or poses.

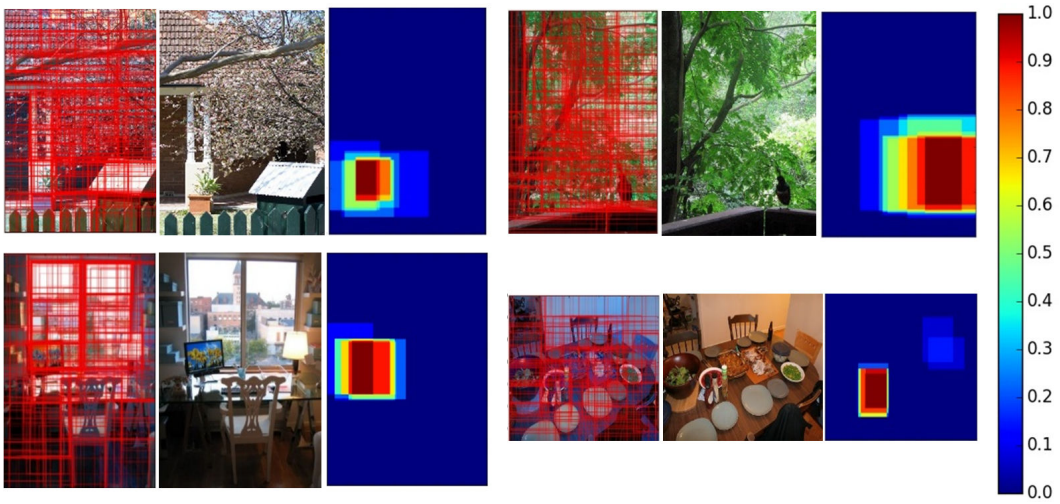


Figure 7.3 – Heatmaps for images `Pascal-VOC-2007` of classes *(clockwise from top-left)* "potted plant", "bird", "bottle" and "TV monitor".

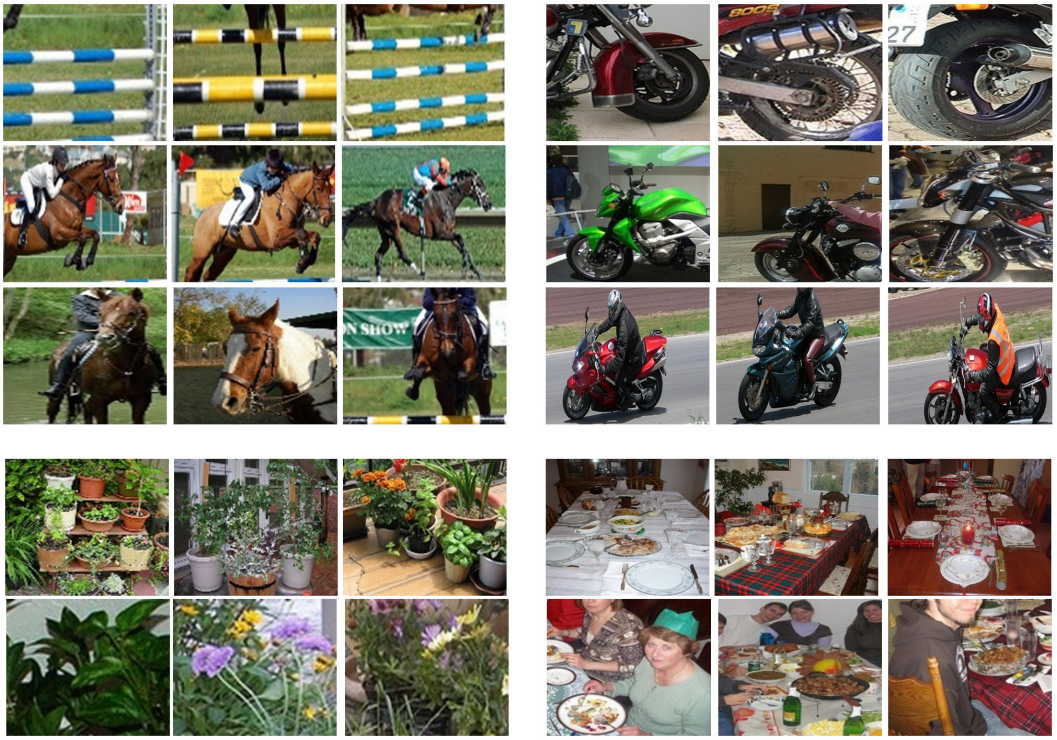


Figure 7.4 – Discriminative parts for the four `Pascal-VOC-2007` classes *(clockwise from top-left)* "horse", "motorbike", "dining table", and "potted plant".

7.7 Conclusions

We introduce SPLeaP, a novel part-based model for image classification. Based on non-linear part classifiers combined with part-dependent soft pooling – both being trained jointly with the image classifier – this new image model consistently surpasses standard pooling approaches and part-based models on several challenging classification tasks. In addition, we have experimentally observed that the proposed method does not need any particular initialization of the parts, contrarily to most of the recent part-based models which require a first step for selecting a few regions candidates from the training images before they actually start learning the parts.

Chapter 8

Soft Pooling of Learned Parts with Per-Part Latent Scale Selection

Contents

8.1 Introduction

This chapter is an extension of the work presented in the previous chapter. The SPleaP algorithm (presented in the previous chapter) depends on a region proposal obtained with the Selective Search (SS) [120] algorithm to obtain an initial set of regions. We know from the last chapter that the algorithm requires computation of CNN descriptors for each region which makes this algorithm costly considering that there is a huge number of regions to operate on. In this chapter, we would like to address this issue of slowness by employing a dense multi-scale sampling to derive region proposals unlike SS.

It is important to note that the slowness in SS is due to the repetitive network computation needed for each region proposal. Therefore, the extraction of descriptors from each of these regions creates a bottle-neck for the overall architecture. It should be noted that the slowness is not due to the region proposal method (SS) itself. In a dense sampling method unlike SS, the description computation is shared across proposals within CNN framework. This is because the dense region proposal scheme can be easily realized using primitive operation such as convolution. This operation can be accelerated by many order using specifically designed GPU based libraries. Therefore, one can easily integrate the dense sampling into our proposed CNN framework which includes PBMs (Part-Based Models). Sermanet *et al.* [98] were the first to propose the usage of multi-scale dense sampling scheme embedded in a CNN feature extraction method for the localization task. Their idea was to apply CNN in a sliding window fashion at both multiple locations per image and multiple scales. In practice, Sermanet *et al.* [98] treat the last fully connected layers as a fully convolutional layers. This allows the whole network to operate on an image of arbitrary size in a sliding window fashion.

As a first contribution, we propose to combine the method proposed by Sermanet with our state-of-the-art method described in the previous chapter to achieve computation acceleration. The part classifiers of our model operates on these densely sampled regions and their responses are then soft pooled to obtain a representation which we refer to as "part-based representation" in the rest of the chapter.

In our second contribution, we propose to use a per-part latent-scale selection method to further introduce scale invariance with respect to the absolute scale of regions in an input image. We apply the method described in the previous paragraph in parallel on multiple scales of input image to obtain a part-based representation for each scale. The resulting scale-dependent part-based representations are then soft-pooled across scales to obtain the final representation. This final representation is then fed to a linear classifier. During the learning process, both the linear classifiers and the part classifiers are jointly optimized in order to minimize the final classification loss. Due to the soft-pooling of part-based representations over multiple scales, the final part-classifiers after optimization become scale invariant in terms of absolute scale of regions in an original image.

The remaining part of this chapter is organized as follows: In Section 8.2 we discuss the related works which used various region proposals schemes. In Section 8.3 we discuss our proposed approach. In Section 8.4 we describe the algorithm used to optimize the parameters in our model. Then, in Section 8.5 we first present the experimental setup for evaluating our proposed approach and subsequently study the effect of different design choices and compare our approach against state-of-the-art methods. Note that we assess our method and design choices against `Pascal-VOC-2007`, `MIT-Indoor-67` and `Willow` datasets.

8.2 Related Work

Recent advances in computer vision tasks such as object detection and classification are obtained by applying CNN on multiple regions per image. The RCNN [38] obtained a best detection performance by applying the CNN on multiple regions (nearly 2000 proposals per image) proposed by SS. This SS-based region proposal is popular because it relies on computationally inexpensive low-level descriptors. Yet RCNN requires huge training time and space because it repeatedly applies expensive CNN feature extraction pipeline on thousands of region proposals, and the resulting region descriptors have to be saved on disk.

SPPnet [46] proposed to address the above difficulty of RCNN [38] by applying the convolutional layers of the network on an entire image only once to obtain feature maps. Then they use a Spatial Pyramid Pooling (SPP) (popularly employed in Bag-of-Words) over these feature maps resulting in a fixed sized representation irrespective of the input image size/scale. Then each fixed size representation can be fed to a sequence of fully connected layers. The authors showed a comparable performance and at the same time increasing the speed by a large margin *w.r.t.* RCNN method.

Another method which attempts to address the difficulty posed by RCNN is the Fast-RCNN [37]. Similar to SPPnet, authors of Fast-RCNN method, first apply convolutional layers on an entire image to obtain feature maps. Next, for each object proposal proposed by SS, they select the corresponding region in the feature map referred by Region of Interest (RoI) in Fast-RCNN [37]. Note that during training time, they select 25 % of the RoIs that overlap with a ground truth bounding box by 50 percent. Finally, they apply a pooling layer per RoI to obtain a fixed length feature vector. Next, each feature vector is fed to the subsequent fully connected layers.

One step further, Faster-RCNN [91] proposed to eliminate the usage of SS by employing Region Pooling Networks (RPN). Here the authors fine tune the RPN to propose the set of object proposals. Next, they use these object proposals instead of SS for training Fast-RCNN. It is important to note here that the costly ground truth bounding box annotations are used to train the RPN.

In our method we use the region proposal scheme similar to Sermanet *et al.* [98] because their approach does not require costly bounding box annotations for generating object proposals. The PBMs automatically select a subset of the proposed candidate regions, and this region selection is driven by the classification loss which in turn requires only image-level annotations. In addition, all the above described methods attempt to show some degree of scale invariance by letting the learning (for e.g., RPNs) to deal with it, instead of addressing it directly. We propose a novel method to achieve scale invariance by using per-part latent scale selection.

A closely related approach by Oquab *et al.* [80], which is based on a sliding window like region proposals, focuses on a localization task. Similar to us they treat the last two dense layers in a CNN as convolutional layers. They then apply a CNN on the images at multiple scales to obtain C feature maps where C indicates the number of classes. They do an explicit search for the object's location within the feature map via max pooling. Our method on the contrary does explicit search for discriminative regions via soft-pooling over intermediate part classifiers' responses. In addition, we also do per-part latent scale selection. In this work we build the architecture similar to [80] and compare it with ours.

8.3 Proposed Approach

In the previous chapter we have seen that multiple regions are extracted per image using SS. Then, we run CNN on each region to obtain a region descriptor. Then, PBMs operate on these region descriptors to automatically identify discriminative regions. Unlike previous chapter, in this chapter we rely on a dense sliding window based region proposal. However, by employing dense proposals per image one does not know beforehand if the given region (*i.e.* constituent fragment of object of interest) occupies the size of the receptive field of the CNN (*i.e.* 224×224). To address this issue we rescale the input image into multiple scales. Then, we run the CNN densely for different scales in parallel to obtain the region descriptors.

An input image is rescaled into the S different scales. We then extract from each rescaled image a bag of regions labeler $r = [1, \ldots, R]$, each one equipped with a descriptor $\mathbf{x}_r^s \in \mathbb{R}^D$, where $s = [1, \ldots, S]$ indicates the scale from which the regions are sampled. The number of regions R per rescaled version is scale-dependent in general even if we assume it is not for notational convenience.

Images in training data span C categories. P "part" classifiers will be learned, each as a weighted sum of K base classifiers applied to a region's descriptor (K chosen by cross-validation). These part classifiers are applied to each region descriptor collected from each scale s. The score of the p-th part classifier for a given descriptor $\mathbf{x}$ is defined as:

$$H_p(\mathbf{x}; \boldsymbol{\theta}_p) = \sum_{k=1}^{K} a_k^p \sigma(\mathbf{x}^\top \mathbf{u}_k^p + b_k^p), \tag{8.1}$$

where σ is the sigmoid function, a_k^p is the weight of the k-th base classifier, $\mathbf{u}_k^p \in \mathbb{R}^D$ and $b_k^p \in \mathbb{R}$ are its parameters and $\boldsymbol{\theta}_p = \text{vec}(a_{1:K}^p, \mathbf{u}_{1:K}^p, b_{1:K}^p) \in \mathbb{R}^{K(D+2)}$ is the vector of all the parameters that define the part classifier.

Let us assume that the collection of R descriptors extracted from the R regions of an image at scale s is $\mathcal{X}_s = \{\mathbf{x}_r^s\}_{r=1}^R$. The score is aggregated over the pool of R regions:

$$g_p^s(\mathcal{X}_s) = \sum_{r=1}^{R} \eta_r^p H_p(\mathbf{x}_{r,s}; \boldsymbol{\theta}_p), \tag{8.2}$$

where part-dependent normalized weight is defined as

$$\eta_r^p \propto \exp\left(\beta_p' H_p(\mathbf{x}_{r,s}; \boldsymbol{\theta}_p)\right), \ \sum_{r=1}^{R} \eta_r^p = 1, \tag{8.3}$$

We then aggregate over scales for the p-th part as follows

$$f^p(\mathcal{X}) = \sum_{s=1}^{S} \alpha_p^s g_p^s(\mathcal{X}_s), \tag{8.4}$$

where scale-dependent normalized weight is defined as

$$\alpha_p^s \propto \exp\left(\beta_s'' g_p^s(\mathcal{X}_s)\right), \ \sum_{s=1}^{S} \alpha_p^s = 1, \tag{8.5}$$

with β_p' and β_s'' are part and scale dependent "pooling" parameters respectively. For large values of this parameter the scores are max-pooled, while they are averaged for very small values.

Given a set of part classifiers with parameter $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1|\cdots|\boldsymbol{\theta}_P]$ with associated $\boldsymbol{\beta'} = [\beta'_p]_{p=1}^P$ and $\boldsymbol{\beta''} = [\beta''_s]_{s=1}^S$ pooling parameters, the bag of region descriptors $\mathcal{X} = \{x_r^s\}_{r,s=1}^{R,S}$ per input image is turned into a final representation:

$$\mathbf{f}(\mathcal{X}; \boldsymbol{\Theta}, \boldsymbol{\beta'}, \boldsymbol{\beta''}) = [f_p(\mathcal{X})]_{p=1}^P . \tag{8.6}$$

On this representation we treat the classification problem at hand by learning independently a binary classifier per class using on-vs-all strategy. By using Support Vector Machines (SVM), we aim at learning P-dimensional vectors, $\mathbf{w}_c = [w_1^c \cdots w_P^c]^\top \in \mathbb{R}^P$, one per class c. We learn the parameters in our model from annotated training set comprised of N images $\mathcal{X}_n$, $n = 1, \ldots, N$, each with class label $y_n^c \in \{-1, 1\}$, $c = 1, \ldots, C$ indicating whether image n belongs to class c or not. Given such training data, our approach is to minimize following objective function with respect to the parameters:

$$\min_{\boldsymbol{\Theta}, \boldsymbol{\beta'}, \boldsymbol{\beta''}, \mathbf{W}} - \sum_{n=1}^{N} \sum_{c=1}^{C} \ell(y_n^k \mathbf{w}_c^\top \mathbf{f}(\mathcal{X}; \boldsymbol{\Theta}, \boldsymbol{\beta'}, \boldsymbol{\beta''})) + \mu \|\boldsymbol{\Theta}\|_F^2 + \delta \|\mathbf{W}\|_F^2, \tag{8.7}$$

where $\mathbf{W} = [\mathbf{w}_1|\cdots|\mathbf{w}_C]$.

The two regularization weights μ and δ, the number P of part classifiers and the number K of base learners in each part are set by cross-validation. Learning is done by block-wise stochastic gradient descent, as explained next into more details.

Latent-scale selection In this work we introduce soft-pooling in two subsequent steps. The first pooling using (8.3) forces the p-th part classifier to select the highest scoring discriminative region at relative scale s. The second pooling step using (8.3) enables the p-part classifier to select the best relative scale. By employing this approach, learnt part classifiers are invariant to absolute scale of an object in an image.

8.4 Optimization specific details

In the current contribution, we learn another parameter called scale-specific pooling parameter $\boldsymbol{\beta''}$ in addition to the parameters such as final category-level classifiers (defined by $\mathbf{W}$), the part classifiers (defined by $\boldsymbol{\Theta}$) and the part-dependent pooling coefficients in $\boldsymbol{\beta'}$ which are learned using the method described in previous chapter Section 7.4. The method used to learn parameter $\boldsymbol{\beta''}$ is similar to $\boldsymbol{\beta'}$.

Algorithm 2 summarizes the different steps of the optimization. In Alg. 2, we denote $\boldsymbol{\theta}_{(k)}$ the vector of parameters associated to k-th base classifiers in matrix $\boldsymbol{\Theta}$, that is $\boldsymbol{\theta}_{(k)} = \text{vec}(a_k^{1:P}, \mathbf{u}_k^{1:P}, b_k^{1:P})$ and $\mathcal{L}_n = \sum_{c=1}^C \ell(y_n^k \mathbf{w}_c^\top \mathbf{f}(\mathcal{X}_n; \boldsymbol{\Theta}, \boldsymbol{\beta'}, \boldsymbol{\beta''})) + \mu \|\boldsymbol{\Theta}\|_F^2 + \delta \|\mathbf{W}\|_F^2$ is the loss function associated with n-th training pair $(\mathcal{X}_n, y_n)$. In the loss function $\mathcal{L}_n$, we have used $\ell(x)$ to denote the hinge loss, given by $\max(0, 1 - x)$.

Algorithm 2 Training: joint part-category classifier learning

procedure LEARN($\mathcal{T}$)
  parameters: P, K, μ, δ, S, γ_W, γ_θ, $\gamma_{\beta'}$, $\gamma_{\beta''}$
  $W \leftarrow 0$
  $\boldsymbol{\theta}_{(1)} \leftarrow$ `rand()`
  $\boldsymbol{\theta}_{(2:K)} \leftarrow 0$
  $\boldsymbol{\beta}' \leftarrow$ `rand()`
  $\boldsymbol{\beta}'' \leftarrow$ `rand()`
  $k \leftarrow 1$
  for $e = 1$ to $E = K + S - 1$ **do**
    $\mathcal{T} \leftarrow RandomShuffle(\mathcal{T})$
    for $n = 1$ to N **do**
      $\mathbf{W} \leftarrow (1 - \gamma_\mathbf{w})\mathbf{W} + \gamma_\mathbf{w} \sum_{(\mathcal{X}_n, y_n) \in \mathcal{T}} \nabla_\mathbf{W} \mathcal{L}_n$
      $\boldsymbol{\theta}_{(k)} \leftarrow (1 - \gamma_\theta)\boldsymbol{\theta}_{(k)} + \gamma_\theta \sum_{(\mathcal{X}_n, y_n) \in \mathcal{T}} \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_n$
      $\boldsymbol{\beta}' \leftarrow \boldsymbol{\beta}' - \gamma'_\beta \sum_{(\mathcal{X}_n, y_n) \in \mathcal{T}} \nabla_{\beta'} \mathcal{L}_n$
      $\boldsymbol{\beta}'' \leftarrow \boldsymbol{\beta}'' - \gamma''_\beta \sum_{(\mathcal{X}_n, y_n) \in \mathcal{T}} \nabla_{\beta''} \mathcal{L}_n$
    end for
    if $e > S$ **then**
      $k \leftarrow k + 1$
    end if
  end for
  Return $\mathbf{W}, \boldsymbol{\Theta}, \boldsymbol{\beta}', \boldsymbol{\beta}''$
end procedure

8.5 Results

In this section, we validate our main contributions presented in this chapter: 1) PBMs operating on a sliding window based region proposals yield good classification performance at reduced computational complexity; 2) Per-part latent-scale selection improves the classification performance. Here, we first introduce the sliding window based region proposals. Then we describe how we use VGG-128/VD16 to represent each region proposal. Next, we discuss the effect of per-part latent scale selection. Then, we discuss the effect of soft pooling coefficients $\boldsymbol{\beta}'$ and $\boldsymbol{\beta}''$. Finally, we compare our method against the state-of-the art results. We use `Pascal-VOC-2007` dataset and mean Average Precision (mAP) performance measure to validate our proposed approach.

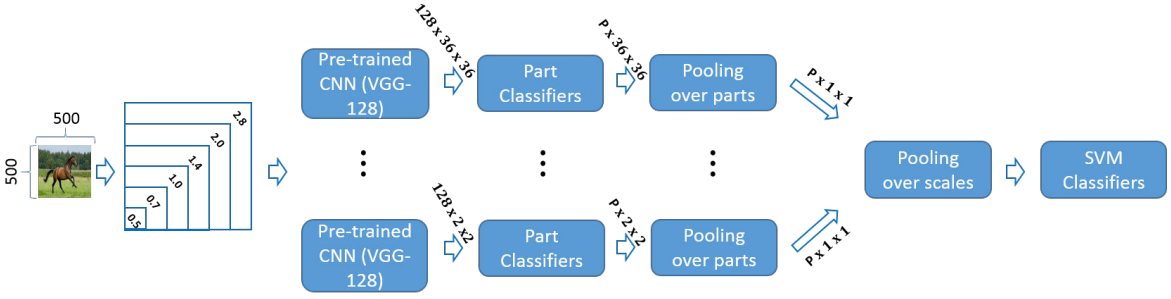


Figure 8.1 – Block diagram of image classification system.

A. Sliding window based region proposals (dense). At both test and training time, we utilize a sliding window procedure to extract regions from the input image and from scaled and mirrored versions of the input image. We follow the procedure described in [80]: each input image is first resized to 500×500 by scaling the longest side of an image to 500 and extending the shortest side to 500 by padding it with zeros. Next the resized image is rescaled by a factor of 6 different scales $\{0.5, 0.7, 1, 1.4, 2.0, 2.8\}$. Note that we repeat the above process with the mirrored version of the input images, thus leading to a total of 12 samples per image. These samples are then fed to a pre-trained CNN in parallel as illustrated in Fig. 8.1. The pre-trained CNN architecture is modified such that it accepts arbitrary sized input image and extracts regions from these samples in a sliding window manner.

B. Feature Extraction. The size of the input image that can be fed into VGG-128/VD-16 is dependent on a receptive field of fully connected layers. We convert the VD-16/VGG-128 into a fully convolutional architecture by treating the weights of the last two layers as 7×7 and 1×1 convolutional kernels, respectively. This enables these CNN architectures to densely extract regions, with a size of each region 256×256 and with a stride of 32 pixels. For example in Fig. 8.1, VGG-128 operating on a resized image (500×500) scaled by the factor of $s = 2.8$ results into 128 feature maps of size 36×36. For arbirary scale s, the size of feature map can be calculated as:

$$size\ of\ feature\ map = \frac{500 \times s - 256}{32} + 1 \tag{8.8}$$

C. Cross-validation of hyperparameters. We use the Stochastic Gradient Descent (SGD) method to train the parameters in our model. We use the method described in Section 7.5.1 of previous chapter to cross validate hyper parameters such as the learning rates ($\gamma_{\mathbf{w}}$, γ_{θ}, $\gamma_{\beta'}$ and $\gamma_{\beta''}$), the number of part-classifiers P, the number of weak learners K, the number of epochs E and the penalty terms (μ and δ).

D. Effect of latent-scale selection. In this paragraph, we evaluate our proposed approach discussed in Section 8.3 that employs latent-scale selection against two different models which do not leverage the benefit of scale selection: 1) In a first model *classifs*, we replace the per-part pooling over scales in Fig. 8.1 with the SVM classifiers which directly operate on aggregated part scores per scale. Next, both at test and train time, we average the prediction scores across samples (refer to Appendix A.1 for more details). 2) In a second model *weaksup*, we adapt the methodology described in [80] in our setup, wherein we remove both per-part and per-scale pooling layers and replace it with C filters (C equal to the number of class categories) with kernel size 1×1. These C filters operate on the previous layer's output (*i.e.* the part classifiers' response maps) to output C feature maps, each one corresponding to the specific class. We then introduce global soft-pooling operation on C feature maps to obtain a single image-level score per class.

In Table 8.1 we compare the three methods described above. For simplicity, we denote the method in this chapter by *SPLeaPS*. We use VGG-128 to extract the features for each region. We keep the parameters such as the number of part classifiers $P = 500$ and the number of weak learners $K = 1$ fixed in all the three cases. However we cross-validate the μ, learning rates and the δ separately in each case.

Table 8.1 – Comparison of our approach with two different models.

weaksup	*classifs*	*SPLeaPS*
81.57	82.73	**83.9**

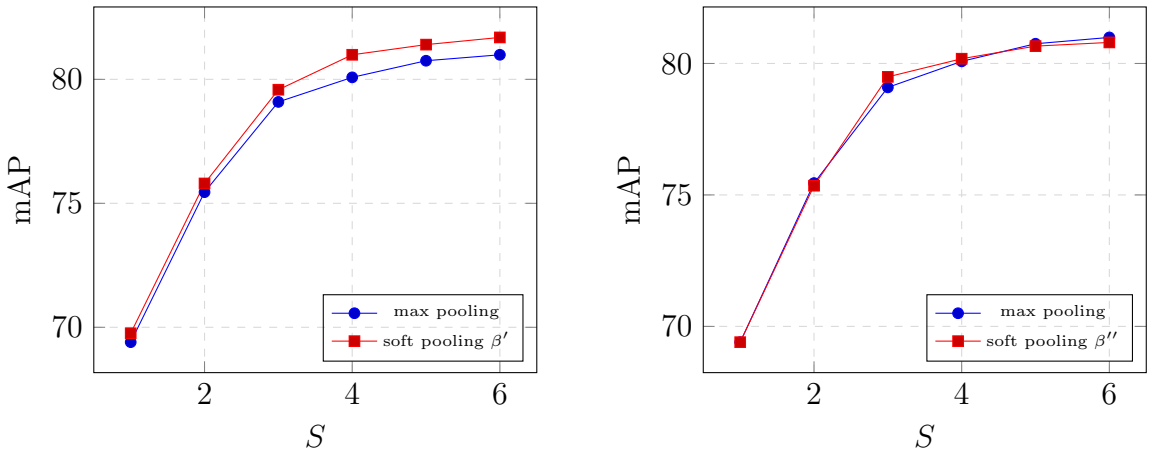


Figure 8.2 – Plot of test mAP versus number of rescaled samples per image S. We evaluate the effect of the spatial pooling coefficients β' (left) and β'' (right) against standard max pooling.

E. Effect of soft-pooling coefficients. We evaluate the effect of soft-pooling coefficients β' and β'' versus the standard max-pooling approach. Note that we fix the β'' to some large value and do not learn it in an optimization process while we evaluate the effect of β'. We fix β' to a large value while evaluating the effect of β''. In Fig. 8.2, y-axis represents mAP over test data that is plotted against the various subsets of $S = 6$ scales $\{0.5, 0.7, 1, 1.4, 2.0, 2.8\}$. That is we vary the number of scales S along x-axis from $S = 1$ (leftmost) to $S = 6$ (rightmost).

In Fig. 8.2 (left), we see that per-part soft pooling consistently outperforms the standard max-pooling. One can observe that for higher values of S the difference is significant. This is because when the number of regions per image increases, per-part soft-pooling leverages the benefits of different aggregation schemes (because each part classifier is equipped with varying $\boldsymbol{\beta}'_p$ values) at work. We can observe from Fig. 8.2 (right) that optimal mAP is obtained for large values of $\boldsymbol{\beta}''_p$, thus performing similar to max-pooling over scales. Note that experiments in Fig. 8.2 are conducted by setting $P = 130$, $E = 3$ and $K = 1$.

F. Comparison with state-of-the-art methods. The simulation environment is similar to Section 7.5.4 in terms of datasets (`Pascal-VOC-2007`, `MIT-Indoor-67` and `Willow`) and CNN architectures (VD-16 and VGG-128). In Table 8.2 and Table 8.3, we compare the method proposed in this chapter (SPLeaPS) with the method described in previous chapter (SPLeaP). It is important to note that we have cross-validated the parameters such as P, K, E and learning rates to optimize the performance of SPLeaPS algorithm.

We establish from Table 8.2 and Table 8.3 that SPLeaPS consistently outperforms

Table 8.2 – Comparison of results on `Pascal-VOC-2007` dataset ($P = 500$ part classifiers, $K = 1$) using CNN features extracted from *(left)* Krizhevsky-like [59] and *(right)* very deep architectures [106]

Methods	*mAP*
VGG-G	75.35
SPLeaP-VGG-128 (*SS+ext. aug*)	**84.68**
SPLeaPS-VGG-128 (*dense*)	83.9

Methods	*mAP*
VD-16-G [106]	81.73
SPLeaP-VD-16 (*SS+ext. aug*)	88.01
SPLeaPS-VD-16 (*dense*)	**88.68**

Table 8.3 – Comparison of results on the `Willow` dataset ($P = 50$ parts, $K = 1$) *(left)* and the `MIT-Indoor-67` dataset ($P = 500$ parts, common to all classes, $K = 1$) *(right)*

Methods	*mAP*
VD-16-G	85.12
SPLeaP-VD-16 (*SS+aug*)	88.47
SPLeaPS-VD-16 (*dense*)	**90.23**

Methods	*mAP*
HybridCNN-G[137]	72.54
SPLeaP-PCA160 (*SS*)	73.45
SPLeaPS-PCA160 (*dense*)	**74.88**

SPLeaP in terms of accuracy in most of the cases. It also processes images, both at training and test time, 300 and 150 times faster than SPLeaP respectively. This is because *dense* region proposals that is employed in SPLeaPS completely eliminates multiple runs through CNN for describing each region proposed by *SS*. In contrast, the computations in the *dense* are being shared across region proposals.

Chapter 9

Summary and Conclusion

Contents

9.1 Summary

In this section we present the chapter wise summary wherein we highlight important contributions made in each chapter and we also present possible future directions. We then present global concluding remarks for the thesis.

9.1.1 Transfer Learning via Attributes

Retrieving images from an unannotated image collection given an arbitrary user query, provided in textual form, is a challenging problem. A recently proposed on-the-fly classification method addresses this by constructing a visual classifier with images returned by an internet image search engine, based on the user query, as positive images while using a fixed pool of negative images. In practice, not all of the images obtained from the internet image search are always pertinent to the query; some might contain abstract or artistic representation of the content and some might have artifacts. Such images degrade the performance of on-the-fly constructed classifier.

In Chapter 3, we proposed a method for improving the performance of on-the-fly classifiers by using transfer learning via attributes. We first mapped the textual query to a set of known attributes and then used those attributes to prune the set of images downloaded from the internet. This pruning step can be seen as zero-shot learning of the visual classifier for the textual user query, which transfers knowledge from the attribute domain to the query domain. We also used the attributes along with the on-the-fly classifier to score the database images and obtain a hybrid ranking. We showed interesting qualitative results and demonstrated by experiments with standard datasets that the proposed method improves upon the baseline on-the-fly classification system.

Future Work We have used a limited set of attributes to demonstrate the above method. The attribute database can be automatically extended using the two steps below:

1. A user query is fed to a textual search engine to extract attributes that are relevant to a given user query.

2. The system can then launch an image search on the internet for both the queried visual concept and the related attributes. Thus, given the relevant images for each attribute from the internet one can build the corresponding attribute classifier and extend the attribute database.

9.1.2 Hybrid multi-layer CNN/Aggregator

Convolutional Neural Networks (CNNs) have established a remarkable performance benchmark in the field of image classification, replacing classical approaches based on hand-tailored aggregations of local descriptors. Yet CNNs impose high computational burdens both at training and at testing time, and training them requires collecting and annotating large amounts of training data. Supervised adaptation methods have been proposed in the literature that partially re-learn a transferred CNN structure from a new target dataset. Yet these require expensive bounding-box annotations and are still computationally expensive to learn. In Chapter 4, we addressed these shortcomings

of CNN adaptation schemes by proposing a hybrid approach that combines conventional, unsupervised aggregators such as Bag-of-Words (BoW) with the CNN pipeline by treating the output of intermediate layers as densely extracted local descriptors.

We tested a variant of our approach that uses only intermediate CNN layers on the standard PASCAL VOC 2007 dataset and showed that performance is significantly higher than the standard BoW model and comparable to Fisher vector aggregation but with a feature that is 150 times smaller. A second variant of our approach that includes the fully connected CNN layers significantly outperformed Fisher vector schemes and performed comparably to CNN approaches adapted to Pascal VOC 2007, yet at only a small fraction of the training and testing cost.

Future Work In Chapter 4, we have aggregated intermediate CNN descriptors by applying a CNN on the entire image. One extension might be to apply a CNN at multiple locations per input image and also at multiple scales to obtain a higher number of CNN descriptors per image. Then, subsequently BoW might be used to aggregate these descriptors.

9.1.3 Max-Margin, Single-Layer Adaptation

In Chapter 5, we proposed to learn a single adaptation layer on top of a pre-trained CNN network jointly with linear classifiers under a max-margin framework. We showed that our learning process takes only a few minutes when compared to other state-of-the art adaptation schemes. In addition, we obtained very good results even with image representation sizes as small as 20.

9.1.4 Learning the Structure of Deep Architecture

In Chapter 6, we presented a method that formulates the selection of the structure of a deep architecture as a penalized, discriminative learning problem. Prior to our work, the structure of deep architectures has been fixed by hand, and only the weights are learned using discriminative learning. Our work was a first attempt towards a more formal method of deep structure selection. We considered architectures consisting only of fully-connected layers, and our approach relies on diagonal matrices inserted between subsequent layers. By including an ℓ_1 norm of the diagonal entries of said matrices as a regularization penalty, we forced the diagonals to be sparse, accordingly selecting the effective number of rows (respectively, columns) of the corresponding layer's (resp. next layer's) weights matrix. We carried out experiments on a standard dataset and showed that our method succeeds in selecting the structure of deep architectures of multiple layers. One variant of our architecture resulted in a feature vector of size as little as 36 that resulted, nonetheless, in competitive image classification performance.

Future Work In the above method we have used an ℓ_1 based regularizer on diagonal matrices to encourage sparsity. We have applied it to a small scale problem such as a network consisting only of fully-connected layers. It would be worth to explore the benefits by applying the same method on an entire CNN, *i.e.* even between the convolutional layers of a CNN.

One interesting extension to our work was proposed by Srinivas *et al.* [110], where they proposed to impose binary constraint on the elements in diagonal matrices. This will make these induced matrices behave as actual selectors of number of rows in

weights matrix (of previous layer's) when compared to real valued non-zero elements that happen when using ℓ_1 regularized diagonal matrices.

9.1.5 Soft Pooling of Learned Parts (SPLeaP)

The aggregation of image statistics – the so-called pooling step of image classification algorithms – as well as the construction of Part-Based Models (PBMs), are two distinct and well-studied topics in the literature. The former aims at leveraging a whole set of local descriptors that an image can contain (through spatial pyramids or Fisher vectors for instance) while the latter argues that only a few of the regions an image contains are actually useful for its classification. Chapter 7 bridges the two worlds by proposing a new pooling framework based on the discovery of useful parts involved in the pooling of local representations. The key contribution lies in a model integrating a boosted non-linear part classifier as well as a parametric soft-max pooling component, both trained jointly with the image classifier. The experimental validation showed that the proposed model not only consistently surpasses standard pooling approaches but also improves over state-of-the-art part-based models, on several different and challenging classification tasks.

9.1.6 SPLeaP with per-part latent scale selection

In Chapter 7, PBMs operate on an initial set of region proposals obtained with Selective Search (SS) algorithm. This makes the SPLeaP algorithm very slow because one has to repeatedly run an entire CNN network for all region proposals. In Chapter 8, we used an alternative dense region proposal method which speeds up computations of an entire SPLeaP algorithm both at train and test-time by a factor of 300x and 150x respectively. Further, in this Chapter, we proposed to operate part-classifiers in PBMs in parallel on multiple scales of input image. The resulting scale-dependent aggregated part classifiers' responses are then soft-pooled across scales. Thus, we introduced per-part latent scale selection in the proposed architecture. We outperformed the results obtained in the previous chapter in most of the cases by a slight margin.

Future Work The part-based models that we employed in Chapters 7 and 8 pool the local regions represented using only the activation coefficients of the penultimate layer of a CNN. One possible future direction is to design a strategy to pool representations from multiple layers of the CNN. We have observed from our work in Chapter 4 that features extracted from multiple layers of a CNN are complimentary. So one can benefit by employing part-based models to find optimal representations across the layers of a CNN.

9.2 Conclusion

Since [59] demonstrated the outstanding results that can be obtained in image classification by using Convolutional Neural Networks (CNNs), the popularity of CNNs has exploded. CNN methods have indeed resulted in very large increases in performance on a wide range of image classification datasets, including large scale datasets such as ImageNet [59], and, by means of knowledge transfer, smaller datasets such as `Pascal-VOC-2007`/2012 [79], SUN397 [40] and `MIT-Indoor-67` datasets [40, 13]. The

emergence of the knowledge transfer technique has shown how one can stretch the limits of classification accuracy on smaller datasets.

We have observed that in day-to-day life we encounter lot of emerging and small size datasets for which it is extremely difficult to train the massive CNN networks containing millions of parameters. In the future, one of the critical things to have is knowledge transfer techniques which will address the issue of training CNNs. In this thesis work we have mainly address the problem of image classification particularly in smaller and challenging datasets via the knowledge transfer technique.

To achieve these goals, we have proposed methods which extend the ability of state-of-the-art CNNs by leveraging knowledge transfer technique. We have proposed several effective techniques to reduce the training and test-time computational burden associated to CNNs by exploiting either a hybrid method to combine conventional low cost BoWs with CNNs, or novel pooling methods within a CNN framework along with non-linear part-based models. In addition, we also proposed a novel method to learn the structure of weights in deep neural networks. We have established our proposed approach by conducting experiments on challenging datasets. We have also compared our method against state-of-the art methods to this date. We also showed how our methods generalize to different tasks in image classification such as object, scene and action classification in still images.

We hope our contribution has advanced the ability of the image classification task in computer vision and has opened new doors for further exploration.

Annexes

Appendix A

Annexes

A.1 Appendix for Chapter 8

In this section we provide details about one of the models that we use to compare against the latent scale selection in Chapter 8. We derive our expressions starting from (8.2) in Section 8.3.

The aggregated score over the pool of R descriptors per image at scale s is given by below expression from Section 8.3:

$$g_p^s(\mathcal{X}_s) = \sum_{r=1}^{R} \eta_r^p H_p(\mathbf{x}_{r,s}; \boldsymbol{\theta}_p), \tag{A.1}$$

where normalized weights are defined as

$$\eta_r^p \propto \exp\left(\beta'_p H_p(\mathbf{x}_{r,s}; \boldsymbol{\theta}_p)\right), \ \sum_{r=1}^{R} \eta_r^p = 1, \tag{A.2}$$

with β'_p are part-dependent "pooling" parameters respectively.

Given a set of part classifiers with parameter $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1|\cdots|\boldsymbol{\theta}_P]$ with associated $\boldsymbol{\beta'} = [\beta'_p]_{p=1}^P$ pooling parameters, the bag of region descriptors $\mathcal{X} = \{x_r^s\}_{r,s=1}^{R,S}$ per input image is turned into a part-based description:

$$\mathbf{f}^s(\mathcal{X}; \boldsymbol{\Theta}, \boldsymbol{\beta'}) = [g_p^s(\mathcal{X}_s)]_{p=1}^P. \tag{A.3}$$

On this representation we treat the classification problem at hand by learning independently binary classifier per class, per-scale using one-vs-all strategy. By using Support Vector Machines (SVM), we aim at learning P-dimensional vectors, $\mathbf{w}_{c,s} = [w_1^{c,s} \cdots w_P^{c,s}]^\top \in \mathbb{R}^P$, one per class and per scale. We will learn these variables from an annotated training set comprised of N images $\mathcal{X}_n$, $n = 1, \ldots, N$, each with class label $y_n^c \in \{-1, 1\}$, $c = 1, \ldots, C$ indicating whether image n belongs to class c or not. Given such training data, our approach is to minimize following objective function with respect to above variables:

$$\min_{\boldsymbol{\Theta}, \boldsymbol{\beta'}, \boldsymbol{\beta''}, \mathbf{W}^s} - \sum_{n=1}^{N} \sum_{c=1}^{C} \ell(y_n^k \mathbf{w}_{s,c}^\top \mathbf{f}^s(\mathcal{X}; \boldsymbol{\Theta}, \boldsymbol{\beta'})) + \mu \|\boldsymbol{\Theta}\|_F^2 + \delta \|\mathbf{W}^s\|_F^2 \quad \forall s \in 1, \ldots, S, \tag{A.4}$$

where $\mathbf{W}^s = [\mathbf{w}_{s,1}|\cdots|\mathbf{w}_{s,C}]$.

Finally, predictions at test time are obtained by averaging the dot of product of image representation with weight vectors over S scales:

$$\frac{1}{S} \sum_{s=1}^{S} \mathbf{W}^s \mathbf{f}^s(\mathcal{X}; \boldsymbol{\Theta}, \boldsymbol{\beta'}). \tag{A.5}$$

A.2 Publications and Patents

The set of publications, workshop and patents accepted in major Computer Vision conferences are:

Publications

1. **Kulkarni, P.**, Jurie, F., Zepeda, J., Perez, P., and Chevallier, L. "SPLeaP: Soft Pooling of Learned Parts for Image Classification". In 2016 IEEE 4th European Conference on Computer Vision (ECCV).

2. **Kulkarni, P.**, Zepeda, J., Jurie, F., Perez, P., and Chevallier, L. "Learning the structure of deep architectures using l1 regularization". In 2015 IEEE British Machine Vision Conference (BMVC).

3. **Kulkarni, P.**, Zepeda, J., Jurie, F., Perez, P., and Chevallier, L. "Hybrid multi-layer deep CNN/aggregator feature for image classification". In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

4. **Kulkarni, P.**, Sharma, G., Zepeda, J., and Chevallier, L. "Transfer learning via attributes for improved on-the-fly classification". In 2014 IEEE Winter Conference on Applications of Computer Vision (WACV).

Workshop

1. **Kulkarni, P.**, Zepeda J., Jurie F., Perez P., Chevallier L. "Max-Margin, Single-Layer Adaptation of Transferred Image Features", In 2015 IEEE CVPR Big Vision Workshop.

Patents filed

1. Zepeda, J., **Kulkarni, P.**, "Method and apparatus for encoding image features using a differentiable bag-of-words encoder.", in US, Patent application no. 14/920,367.

2. **Kulkarni, P.**, Zepeda, J., Jurie, F. "Method and apparatus for image classification with joint feature adaptation and classifier learning.", in US, Patent application no. 14/942302.

3. **Kulkarni, P.**, Sharma, G., Zepeda, J., and Chevallier, L. "Transfer Learning via Attributes for Improved On-the-fly Classification.", in Technicolor (2014).

4. Claire-Helene, D., **Kulkarni, P.**, Sharma, G., Zepeda, J., and Chevallier, L. "User tracking for intelligent capture.", in Technicolor (2015).

5. **Kulkarni, P.**, Claire-Helene, D., Sharma, G., Zepeda, J., and Chevallier, L. "Real time camera calibration from artistic images to enhance captured scene.", in Technicolor (2015).

Bibliography

[1] Arandjelovic, R., and Zisserman, A. Three things everyone should know to improve object retrieval. *CVPR* (2012). 41

[2] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. Speeded-up robust features (surf). *Computer vision and image understanding 110*, 3 (2008), 346–359. 8

[3] Bolle, R. M., Connell, J. H., Hampapur, A., and Senior, A. W. System and method for automatically setting image acquisition controls, Oct. 9 2001. US Patent 6,301,440. 6

[4] Bosch, A., Zisserman, A., and Munoz, X. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval* (2007), ACM, pp. 401–408. 38, 41

[5] Bottou, L. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, G. Montavon, G. Orr, and K.-R. Müller, Eds., 2 ed., vol. 1. Springer, 2012. 55

[6] Boureau, Y.-L., Bach, F., LeCun, Y., and Ponce, J. Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 2559–2566. 17

[7] Boureau, Y.-L., Ponce, J., and LeCun, Y. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), pp. 111–118. 15

[8] Chatfield, K., Arandjelović, R., Parkhi, O., and Zisserman, A. On-the-fly learning for visual search of large-scale image and video datasets. *International journal of multimedia information retrieval 4*, 2 (2015), 75–93. 5, 10

[9] Chatfield, K., Lempitsky, V., Vedaldi, A., and Zisserman, A. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC* (2011), no. 1, pp. 76.1–76.12. 38, 41, 44

[10] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *British Machine Vision Conference* (2014). 9, 46, 47, 48, 51, 52, 53, 56, 59, 62, 66, 67, 69, 70

[11] Chatfield, K., and Zisserman, A. Visor: Towards on-the-fly large-scale object category retrieval. In *Asian Conference on Computer Vision* (2012), Springer, pp. 432–446. iv, 26, 28, 29, 30, 31, 32, 35

[12] Chavali, N., Agrawal, H., Mahendru, A., and Batra, D. Object-proposal evaluation protocol is 'gameable'. In *arXiv:1505.05836* (2015). 66

[13] Cimpoi, M., Maji, S., and Vedaldi, A. Deep filter banks for texture recognition and segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2015). 17, 63, 69, 70, 85

[14] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297. 8, 19

[15] Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV* (2004), vol. 1, Prague, pp. 1–2. 28, 37

[16] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (2005), vol. 1, IEEE, pp. 886–893. 8

[17] De Souza, F. D., Chavez, G. C., do Valle Jr, E. A., and Araújo, A. d. A. Violence detection in video using spatio-temporal features. In *2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images* (2010), IEEE, pp. 224–230. 6

[18] Delaitre, V., Laptev, I., and Sivic, J. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *British Machine Vision Conference* (2010). 9, 20

[19] Delhumeau, J., Gosselin, P.-H., Jégou, H., and Pérez, P. Revisiting the VLAD image representation. In *Proceedings of ACM International Conference on Multimedia* (New York, New York, USA, 2013), vol. 21, ACM Press, pp. 653–656. 37, 41

[20] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 248–255. 9

[21] Denoyer, L., and Gallinari, P. A ranking based model for automatic image annotation in a social network. In *ICWSM* (2010). 5

[22] Doersch, C., Gupta, A., and Efros, A. A. Mid-level visual element discovery as discriminative mode seeking. In *Advances in neural information processing systems* (2013), pp. 494–502. 8, 17, 61, 63

[23] Doersch, C., Singh, S., Gupta, A., Sivic, J., and Efros, A. What makes paris look like paris? *ACM Transactions on Graphics 31*, 4 (2012). 61, 63

[24] Duan, L., Tsang, I. W., Xu, D., and Chua, T.-S. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 289–296. 28

[25] Duan, L., Tsang, I. W., Xu, D., and Maybank, S. J. Domain transfer svm for video concept detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 1375–1381. 29

[26] Duan, L.-Y., Xu, M., Chua, T.-S., Tian, Q., and Xu, C.-S. A mid-level representation framework for semantic sports video analysis. In *Proceedings of the eleventh ACM international conference on Multimedia* (2003), ACM, pp. 33–44. 17

[27] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 9, 20, 26, 28, 30, 41, 62

[28] Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. Describing objects by their attributes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 1778–1785. 26, 28

[29] Farhadi, A., Tabrizi, M. K., Endres, I., and Forsyth, D. A latent model of discriminative aspect. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 948–955. 9

[30] Fathi, A., and Mori, G. Action recognition by learning mid-level motion features. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8. 17

[31] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence 32*, 9 (2010), 1627–1645. 8, 61, 63

[32] Fernando, B., Fromont, E., and Tuytelaars, T. Mining mid-level features for image classification. *International Journal of Computer Vision 108*, 3 (2014), 186–203. 17

[33] Fischler, M. A., and Elschlager, R. A. The Representation and Matching of Pictorial Structures. *IEEE Trans. Computers 22*, 1 (1973), 67–92. 61

[34] Friedman, J., Hastie, T., Tibshirani, R., et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics 28*, 2 (2000), 337–407. 61

[35] Gatys, L. A., Ecker, A. S., and Bethge, M. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2414–2423. 9

[36] George, M., and Floerkemeier, C. Recognizing products: A per-exemplar multi-label image classification approach. In *European Conference on Computer Vision* (2014), Springer, pp. 440–455. 5

[37] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1440–1448. 11, 75

[38] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on* (2014), IEEE, pp. 580–587. 52, 75

[39] Goh, H., Thome, N., and Cord, M. Top-Down Regularization of Deep Belief Networks. 1–9. 52

[40] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV 2014*. Springer, 2014, pp. 392–407. 17, 37, 43, 51, 52, 63, 71, 85

[41] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. Multi-scale Orderless Pooling of Deep Convolutional Activation Features. In *European Conference on Computer Vision* (Mar. 2014). 52

[42] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. Maxout networks. *ICML (3) 28* (2013), 1319–1327. 63

[43] Gulcehre, C., Cho, K., Pascanu, R., and Bengio, Y. Learned-norm pooling for deep feedforward and recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2014), Springer, pp. 530–546. 63

[44] Hafemann, L. G., Oliveira, L. S., Cavalin, P. R., and Sabourin, R. Transfer learning between texture classification tasks using convolutional neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)* (2015), IEEE, pp. 1–7. 9

[45] Hansen, T., Olkkonen, M., Walter, S., and Gegenfurtner, K. R. Memory modulates color appearance. *Nature neuroscience 9*, 11 (2006), 1367–1368. 28

[46] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 37*, 9 (2015), 1904–1916. 52, 69, 75

[47] Hoai, M., and Zisserman, A. Improving human action recognition using score distribution and ranking. In *Asian Conference on Computer Vision*. 2014. 61

[48] Idris, F., and Panchanathan, S. Review of image and video indexing techniques. *Journal of visual communication and image representation 8*, 2 (1997), 146–166. 6

[49] Jaakkola, T. S., Haussler, D., et al. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems* (1999), 487–493. 8, 15

[50] Jégou, H., Douze, M., Schmid, C., and Pérez, P. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 3304–3311. 8, 63

[51] Jégou, H., and Zisserman, A. Triangulation embedding and democratic aggregation for image search. In *CVPR* (2014). 41

[52] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia* (2014). 51, 67, 70

[53] Juneja, M., Vedaldi, A., Jawahar, C., and Zisserman, A. Blocks that shout: Distinctive parts for scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 923–930. 8, 61, 63

[54] Jurie, F., and Triggs, B. Creating efficient codebooks for visual recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (2005), vol. 1, IEEE, pp. 604–610. 15

[55] Ke, Y., and Sukthankar, R. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (2004), vol. 2, IEEE, pp. II–506. 14

[56] Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. Learning systems of concepts with an infinite relational model. In *AAAI* (2006), vol. 3, p. 5. 31

[57] Khan, F. S., Anwer, R. M., van de Weijer, J., Bagdanov, A. D., Lopez, A. M., and Felsberg, M. Coloring action recognition in still images. *International journal of computer vision 105*, 3 (2013), 205–221. 71

[58] Khosla, A., Zhou, T., Malisiewicz, T., Efros, A. A., and Torralba, A. Undoing the damage of dataset bias. In *European Conference on Computer Vision* (2012), Springer, pp. 158–171. 9

[59] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105. iv, 1, 9, 17, 38, 39, 51, 53, 62, 67, 69, 70, 81, 85

[60] Kulkarni, P., Zepeda, J., Jurie, F., Perez, P., and Chevallier, L. Hybrid multi-layer deep cnn/aggregator feature for image classification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing* (2015). 52, 63

[61] Kulkarni, P., Zepeda, J., Jurie, F., Perez, P., and Chevallier, L. Max-Margin, Single-Layer Adaptation of Transferred Image Features. In *BigVision Workshop, Computer Vision and Pattern Recognition* (2015). 51, 52, 53, 67

[62] Kumar, N., Berg, A. C., Belhumeur, P. N., and Nayar, S. K. Attribute and simile classifiers for face verification. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 365–372. 28

[63] Lampert, C. H., Nickisch, H., and Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 951–958. 8, 26, 28, 29, 30, 31

[64] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), vol. 2, IEEE, pp. 2169–2178. 16, 26, 28, 32, 38, 41, 52

[65] LeCun, Y., Bottou, L., Orr, G., and Muller, K.-R. Efficient BackProp. In *Neural Networks: Tricks of the Trade*. 2002, pp. 9–50. 55

[66] Lee, C.-Y., Gallagher, P. W., and Tu, Z. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *International Conference on Artificial Intelligence and Statistics* (2016). 63

[67] Li, L.-J., Su, H., Fei-Fei, L., and Xing, E. P. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in neural information processing systems* (2010), pp. 1378–1386. 28

[68] Li, M., Zang, S., Zhang, B., Li, S., Wu, C., et al. A review of remote sensing image classification techniques: The role of spatio-contextual information. *European Journal of Remote Sensing 47* (2014), 389–411. 5

[69] Li, Q., Wu, J., and Tu, Z. Harvesting mid-level visual concepts from large-scale internet images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 851–858. 17

[70] Li, Y., Liu, L., Shen, C., and van den Hengel, A. Mid-level deep pattern mining. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2015). 63, 69, 70, 71

[71] Lindstaedt, S., Pammer, V., Mörzinger, R., Kern, R., Mülner, H., and Wagner, C. Recommending tags for pictures based on text, visual content and user context. In *Internet and Web Applications and Services, 2008. ICIW'08. Third International Conference on* (2008), IEEE, pp. 506–511. 5

[72] Liu, C., Yuen, J., Torralba, A., Sivic, J., and Freeman, W. T. Sift flow: Dense correspondence across different scenes. In *European conference on computer vision* (2008), Springer, pp. 28–42. 14

[73] Lobel, H., Vidal, R., and Soto, A. Hierarchical joint Max-Margin learning of mid and top level representations for visual recognition. In *IEEE International Conference on Computer Vision* (2013). 61, 63

[74] Lowe, D. G. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on* (1999), vol. 2, Ieee, pp. 1150–1157. 28

[75] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110. iii, 8, 15, 37

[76] Malisiewicz, T., Gupta, A., and Efros, A. A. Ensemble of exemplar-svms for object detection and beyond. In *2011 International Conference on Computer Vision* (2011), IEEE, pp. 89–96. iii, 26, 27

[77] Mason, L., Baxter, J., Bartlett, P., and Frean, M. Boosting algorithms as gradient descent in function space. NIPS. 61

[78] Muja, M., and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1) 2*, 331-340 (2009), 2. 15

[79] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 1717–1724. 1, 9, 37, 40, 43, 44, 46, 47, 48, 52, 53, 59, 62, 69, 70, 85

[80] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 685–694. 75, 79

[81] Orabona, F., Castellini, C., Caputo, B., Fiorilla, A. E., and Sandini, G. Model adaptation with least-squares svm for adaptive hand prosthetics. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 2897–2903. 9

[82] Osherson, D., Smith, E. E., Myers, T. S., Shafir, E., and Stob, M. Extrapolating human probability judgment. *Theory and Decision 36*, 2 (1994), 103–129. 28

[83] Osherson, D. N., Stern, J., Wilkie, O., Stob, M., and Smith, E. E. Default probability. *Cognitive Science 15*, 2 (1991), 251–269. 28

[84] Pan, S. J., and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering 22*, 10 (2010), 1345–1359. 9

[85] Parizi, S. N., Vedaldi, A., Zisserman, A., and Felzenszwalb, P. Automatic Discovery and Optimization of Parts for Image Classification. In *International Conference on Learning Representations* (2015). 61, 63, 71

[86] Parkhi, O. M., Vedaldi, A., and Zisserman, A. On-the-fly specific person retrieval. In *Image Analysis for Multimedia Interactive Services (WIAMIS), 2012 13th International Workshop on* (2012), IEEE, pp. 1–4. 38

[87] Perronnin, F., and Dance, C. Fisher kernels on visual vocabularies for image categorization. In *CVPR* (2007), pp. 1–8. 37

[88] Perronnin, F., Sánchez, J., and Mensink, T. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision* (2010), Springer, pp. 143–156. 8, 15, 19, 38, 42, 44, 52

[89] Quattoni, A., and Torralba, A. Recognizing indoor scenes. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2009). 9, 20

[90] Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops* (2014). 62

[91] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99. 75

[92] Ro, Y. M., Kim, M., Kang, H. K., Manjunath, B., and Kim, J. Mpeg-7 homogeneous texture descriptor. *ETRI journal 23*, 2 (2001), 41–51. 7

[93] Russakovsky, O., and Fei-Fei, L. Attribute learning in large-scale datasets. In *Trends and Topics in Computer Vision*. Springer, 2012, pp. 1–14. 29

[94] Saad, D. Online algorithms and stochastic approximations. *Online Learning*. 16

[95] Sabzmeydani, P., and Mori, G. Detecting pedestrians by learning shapelet features. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), IEEE, pp. 1–8. 17

[96] Sanchez, J., and Perronnin, F. High-dimensional signature compression for large-scale image classification. *CVPR* (June 2011), 1665–1672. 39

[97] Sawant, N., Datta, R., Li, J., and Wang, J. Z. Quest for relevant tags using local interaction networks and visual content. In *Proceedings of the international conference on Multimedia information retrieval* (2010), ACM, pp. 231–240. 5

[98] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229* (2013). 74, 75

[99] Shalev-shwartz, S., and Srebro, N. Pegasos : Primal Estimated sub-GrAdient SOlver for SVM. 55

[100] Sharif, A., Hossein, R., Josephine, A., Stefan, S., and Royal, K. T. H. CNN Features off-the-shelf : an Astounding Baseline for Recognition. 59

[101] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2014), pp. 806–813. 9, 17, 46, 51, 52, 59, 62

[102] Sharma, G., Jurie, F., and Schmid, C. Discriminative spatial saliency for image classification. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2012). 28, 71

[103] Sharma, G., Jurie, F., and Schmid, C. Expanded parts model for human attribute and action recognition in still images. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2013). 71

[104] Sicre, R., and Jurie, F. Discovering and aligning discriminative mid-level features for image classification. In *International Conference on Pattern Recognition* (2014). 63, 71

[105] Simonyan, K., Vedaldi, A., and Zisserman, A. Deep fisher networks for large-scale image classification. In *NIPS* (2013), pp. 163–171. 37

[106] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014). 1, 67, 69, 70, 81

[107] Singh, S., Gupta, A., and Efros, A. A. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–ECCV 2012*. Springer, 2012, pp. 73–86. 8, 17, 61, 63

[108] Sivic, J., and Zisserman, A. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (2003), IEEE, pp. 1470–1477. 8, 15, 28, 38, 52

[109] Sloman, S. A. Feature-based induction. *Cognitive psychology 25*, 2 (1993), 231–280. 28

[110] Srinivas, S., and Babu, R. V. Learning the architecture of deep neural networks. *CoRR abs/1511.05497* (2015). 84

[111] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research 15* (2014), 1929–1958. 51

[112] Stone, Z., Zickler, T., and Darrell, T. Autotagging facebook: Social network context improves photo annotation. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on* (2008), IEEE, pp. 1–8. 5

[113] Su, Y., and Jurie, F. Improving image classification using semantic attributes. *International journal of computer vision 100*, 1 (2012), 59–77. 26, 28

[114] Swain, M. J., and Ballard, D. H. Color indexing. *International journal of computer vision 7*, 1 (1991), 11–32. 7

[115] Sydorov, V., Sakurada, M., and Lampert, C. H. Deep fisher kernels–end to end learning of the fisher kernel gmm parameters. In *CVPR* (2014), pp. 1402–1409. 37, 54

[116] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2015). 70

[117] Torresani, L., Szummer, M., and Fitzgibbon, A. Efficient object category recognition using classemes. In *European conference on computer vision* (2010), Springer, pp. 776–789. 28

[118] Ullman, S., Sali, E., and Vidal-Naquet, M. A Fragment-Based Approach to Object Representation and Classification. In *Visual Form 2001*. 2001. 61

[119] Van De Sande, K., Gevers, T., and Snoek, C. Evaluating color descriptors for object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence 32*, 9 (2010), 1582–1596. 16

[120] van de Sande, K. E. A., Uijlings, J. R. R., Gevers, T., and Smeulders, A. W. M. Segmentation as selective search for object recognition. In *IEEE International Conference on Computer Vision* (2011). 66, 74

[121] van Gemert, J. C., Geusebroek, J.-M., Veenman, C. J., and Smeulders, A. W. Kernel codebooks for scene categorization. In *ECCV*. Springer, 2008, pp. 696–709. 37

[122] Van Gemert, J. C., Veenman, C. J., Smeulders, A. W., and Geusebroek, J.-M. Visual word ambiguity. *IEEE transactions on pattern analysis and machine intelligence 32*, 7 (2010), 1271–1283. 8

[123] Vedaldi, A., and Fulkerson, B. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia* (2010), ACM, pp. 1469–1472. 32

[124] Vogel, J., and Schiele, B. Semantic modeling of natural scenes for content-based image retrieval. *International Journal of Computer Vision 72*, 2 (2007), 133–157. 28

[125] Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. Regularization of Neural Networks using DropConnect. In *International Conference of Machine Learning* (2013). 52

[126] Wang, G., Hoiem, D., and Forsyth, D. Learning image similarity from flickr groups using stochastic intersection kernel machines. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 428–435. 28

[127] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. Locality-constrained linear coding for image classification. In *CVPR* (2010), pp. 3360–3367. 37

[128] Weber, M., Welling, M., and Perona, P. Towards automatic discovery of object categories. 61

[129] Wu, P., and Dietterich, T. G. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the twenty-first international conference on Machine learning* (2004), ACM, p. 110. 29

[130] Xue, S., Agarwala, A., Dorsey, J., and Rushmeier, H. Learning and applying color styles from feature films. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 255–264. 6

[131] Yang, J., Yan, R., and Hauptmann, A. G. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia* (2007), ACM, pp. 188–197. 28

[132] Yang, J., Yu, K., Gong, Y., and Huang, T. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 1794–1801. 15

[133] Yu, X., and Aloimonos, Y. Attribute-based transfer learning for object categorization with zero/one training example. In *European conference on computer vision* (2010), Springer, pp. 127–140. 28

[134] Zeiler, M. D., and Fergus, R. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision* (2014), Springer, pp. 818–833. 16

[135] Zhang, D., and Lu, G. Generic fourier descriptor for shape-based image retrieval. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on* (2002), vol. 1, IEEE, pp. 425–428. 7

[136] Zhang, J., Marszałek, M., Lazebnik, S., and Schmid, C. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision 73*, 2 (2007), 213–238. 16

[137] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems* (2014), pp. 487–495. 9, 67, 70, 71, 81

[138] Zhou, X., Yu, K., Zhang, T., and Huang, T. S. Image classification using super-vector coding of local image descriptors. In *European conference on computer vision* (2010), Springer, pp. 141–154. 8, 37