



HAL
open science

Of Learning Visual Representations Robust to Invariances for Image Classification and Retrieval

Mattis Paulin

► **To cite this version:**

Mattis Paulin. Of Learning Visual Representations Robust to Invariances for Image Classification and Retrieval. Computer Vision and Pattern Recognition [cs.CV]. Université Grenoble Alpes, 2017. English. NNT: . tel-01677852v1

HAL Id: tel-01677852

<https://inria.hal.science/tel-01677852v1>

Submitted on 8 Jan 2018 (v1), last revised 11 Jan 2018 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques, sciences et technologies de l'information**

Arrêté ministériel : 7 aout 2006

Présentée par

Mattis Paulin

Thèse dirigée par **Cordelia Schmid**
et codirigée par **Zaid Harchaoui**

préparée au sein

et de l'école doctorale **MSTII : Mathématiques, Sciences et Technologies de l'Information, Informatique**

De l'Apprentissage de représentations visuelles robustes aux invariances pour la classification et la recherche d'images

Of Learning Visual Representations Robust to Invariances for Image Classification and Retrieval

Thèse soutenue publiquement le **6 février 2017**,
devant le jury composé de :

Pr. Matthieu Cord

Université Pierre et Marie Curie, Paris, France, Rapporteur

Dr. Josef Sivic

Ecole Normale Supérieure, Paris, France, Rapporteur

Pr. Vincent Lepetit

Technische Universität Graz, Austria, Examineur

Dr. Christian Wolf

Institut National des Sciences Appliquées, Lyon, France, Examineur

Dr. Florent Perronnin

Xerox Research Center Europe, Montbonnot, France, Examineur

Dr. Julien Mairal

Inria Grenoble, Montbonnot, France, Examineur

Dr. Cordelia Schmid

Inria Grenoble, Montbonnot, France, Directeur de thèse

Dr. Zaid Harchaoui

University of Washington, Seattle, WA, USA, Co-Directeur de thèse



Abstract

This dissertation focuses on designing image recognition systems which are robust to geometric variability. Image understanding is a difficult problem, as images are two-dimensional projections of 3D objects, and representations that must fall into the same category, for instance objects of the same class in classification can display significant differences. Our goal is to make systems robust to the right amount of deformations, this amount being automatically determined from data. Our contributions are twofolds. We show how to use virtual examples to enforce robustness in image classification systems and we propose a framework to learn robust low-level descriptors for image retrieval.

We first focus on virtual examples, as transformation of real ones. One image generates a set of descriptors –one for each transformation– and we show that data augmentation, ie considering them all as iid samples, is the best performing method to use them, provided a voting stage with the transformed descriptors is conducted at test time. Because transformations have various levels of information, can be redundant, and can even be harmful to performance, we propose a new algorithm able to select a set of transformations, while maximizing classification accuracy. We show that a small amount of transformations is enough to considerably improve performance for this task. We also show how virtual examples can replace real ones for a reduced annotation cost. We report good performance on standard fine-grained classification datasets.

In a second part, we aim at improving the local region descriptors used in image retrieval and in particular to propose an alternative to the popular SIFT descriptor. We propose new convolutional descriptors, called patch-CKN, which are learned without supervision. We introduce a linked patch- and image-retrieval dataset based on structure from motion of web-crawled images, and design a method to accurately test the performance of local descriptors at patch and image levels. Our approach outperforms both SIFT and all tested approaches with convolutional architectures on our patch and image benchmarks, as well as several state-of-the-art datasets.

Résumé

Ce mémoire de thèse porte sur l'élaboration de systèmes de reconnaissance d'image qui sont robustes à la variabilité géométrique. La compréhension d'une image est un problème difficile, de par le fait qu'elles sont des projections en deux dimensions d'objets 3D. Par ailleurs, des représentations qui doivent appartenir à la même catégorie, par exemple des objets de la même classe en classification, peuvent être visuellement très différentes. Notre but est de rendre ces systèmes robustes à la juste quantité de déformations, celle-ci étant automatiquement déterminée à partir des données. Nos deux contributions sont les suivantes. Nous montrons tout d'abord comment utiliser des exemples virtuels pour rendre les systèmes de classification d'images robustes et nous proposons ensuite une méthodologie pour apprendre des descripteurs de bas niveau robustes, pour la recherche d'image.

Nous étudions tout d'abord les exemples virtuels, en tant que transformations de vrais exemples. En représentant une image en tant que sac de descripteurs transformés, nous montrons que l'augmentation de données, c'est-à-dire le fait de les considérer comme de nouveaux exemples iid, est la meilleure manière de les utiliser, pourvu qu'une étape de vote avec les descripteurs transformés soit opérée lors du test. Du fait que les transformations apportent différents niveaux d'information, peuvent être redondants, voire nuire à la performance, nous proposons un nouvel algorithme capable de sélectionner un petit nombre d'entre elles, en maximisant la justesse de classification. Nous montrons par ailleurs comment remplacer de vrais exemples par des virtuels, pour alléger les coûts d'annotation. Nous rapportons de bons résultats sur des bancs d'essai de classification.

Notre seconde contribution vise à améliorer les descripteurs de régions locales utilisés en recherche d'image, et en particulier nous proposons une alternative au populaire descripteur SIFT. Nous proposons un nouveau descripteur, appelé patch-CKN, appris sans supervision. Nous introduisons un nouvel ensemble de données liant les images et les imagerie, construit à partir de reconstruction 3D automatique d'images récupérées sur Internet. Nous définissons une méthode pour tester précisément la performance des descripteurs locaux au niveau de l'imagerie et de l'image. Notre approche dépasse SIFT et les autres approches à base d'architectures convolutionnelles sur notre banc d'essai, et d'autres couramment utilisés dans la littérature.

Contents

Contents	iv
List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Goals	4
1.2 Context	10
1.3 Contributions	14
2 Selecting Virtual Examples for Image Classification	18
2.1 Introduction	19
2.2 Related Work	20
2.3 Virtual Examples: Generation and Selection	23
2.4 Experiments	31
3 Patch Description with Convolutional Kernel Networks	44
3.1 Introduction	45
3.2 Related Work	46
3.3 Convolutional Descriptors	49
3.4 Convolutional Kernel Descriptors	52
3.5 Image and Patch Retrieval Datasets	61
3.6 Experiments	63
4 Conclusion	80
4.1 Summary of Contributions	81
4.2 Future Directions	82
A Publications	85
B Released Software	87

CONTENTS

v

B.1 JSGD	87
B.2 ITP	90
B.3 Patch-CKN	91

Bibliography

92

List of Figures

1.1	The Allegory of the Cave, according to Cornelis van Haarlem, 1604 . . .	3
1.2	Examples of semantic and geometric variability	5
1.3	Example of four different classes of matching objects in the Rome dataset, introduced in this thesis [Paulin et al., 2016]	6
1.4	Examples and statistics of the state-of-the-art retrieval datasets	7
1.5	Images of two different classification tasks.	8
1.6	The process of collecting the PASCAL VOC and ImageNet datasets . .	10
1.7	Evolution of the best performing method on the ImageNet dataset. . .	13
1.8	Evolution of the best performing methods on the Holidays dataset . . .	14
1.9	Our approach to boost classification performances with virtual examples	15
1.10	Our approach for image retrieval with local convolutional descriptors .	16
2.1	Illustrations of image transformations, on simple icons and on an image from the CUB dataset.	21
2.2	Four of the eight homographies we use. The four others we consider are similar, but with a broader window.	24
2.3	Illustration of the different training and test strategies.	34
2.4	Evolution of the test accuracy with respect to the number of selected transformations.	36
2.5	Accuracy on CUB after adding up to five transformations with ITP, with random selection (5 trials averaged, error bars shown), or with just flip in addition to the original images (\mathcal{T}_1).	37
2.6	Evolution of the test accuracy on CUB as a function of the number of transformations selected by ITP, or its cheaper TR variant.	37
2.7	First five transformations, as selected by ITP.	38
2.8	Overlaying the different crops selected by ITP.	38
2.9	Test accuracy as a function of the number of SGD iterations on CUB (left) with SIFT-Fisher.	39
2.10	Equivalence in terms of accuracy of images with virtual examples and lone examples on CUB (30 images per class is the whole dataset) with ITP. Left: SIFT-Fisher. Right: Color-Fisher.	40

2.11	Evolution of the accuracy with an increased number of images per class on the CUB dataset with transformations selected by ITP. Left: SIFT-Fisher. Right: Color-Fisher.	40
2.12	Histogram of the average distance of a transformed example to its original	42
2.13	All transformations used in this article, applied to a CUB image. . . .	43
3.1	Typical organization of two successive layers of a CNN.	50
3.2	Construction of the sequence of feature maps	55
3.3	Vizualization of the possible CKN inputs	61
3.4	Examples of patches matched under our procedure. We observe significant changes in lighting, but smaller changes in rotation and skew. . .	64
3.5	Example of classes of the image retrieval dataset of Rome. Each class consists of a particular location. Some bundle display significant view-point changes (extreme left and right), while others have little variation in appearance (middle). Best viewed in color.	64
3.6	Example of matching points in two different images.	65
3.7	All 512 kernels of the first layer of our CKN-raw architecture.	68
3.8	mAP results on the train set of RomePatches with CKN-grad, whose hyper-parameters have been changed one by one around the optimal point	70
3.9	Influence of dimensionality reduction on patch retrieval performance . .	71
3.10	First convolutional filters of the PhilippNet learned with surrogate classes (left), 10K Rome classes (middle) and 100K Rome classes. Best viewed in color.	74
3.11	Impact of various transformations on CKN descriptors	75

List of Tables

2.1	Comparison between ITP and ITW for CNN descriptors.	34
2.2	Comparison of different training and test scenarios for $T = 6$, i.e. $T - 1$ corresponds to the number of transformations used for training and test. Score aggregation is performed by averaging. Transformations selected with ITP.	35
2.3	Comparison of score aggregation schemes.	35
2.4	Comparison of different schemes for using virtual examples at training time. All transformations were selected by ITP.	35
2.5	Number of training samples (in thousands) that the SGD has to process to reach a given accuracy on the test set, for a varying number of transformations T for Fisher-Fusion features. The 'X' sign means that the target accuracy cannot be reached at all.	39
2.6	Performance of our method compared to the state of the art. Our method holds state-of-the-art performance for off-the-shelf features. . .	41
3.1	For each layer we indicate the sub-patch size, the subsampling factor and the number of filters. For the gradient network, the value 16 corresponds to the number of orientations.	69
3.2	Results of convolutional architectures for patch retrieval.	71
3.3	Evaluation of the deepcompare architectures on RomePatches in mAP (%). Networks were trained on the three subsets of the Multi-view Stereo Correspondence Dataset: Yosemite (Y), Notre-Dame (ND) and Statue of Liberty (L). The network notations are as in the original paper [Zagoruyko and Komodakis, 2015].	72
3.4	Experiments on a set of pairs of the Liberty dataset. Measure is false positive rate @95% recall, and therefore the lower the better. The third layer of AlexNet is used, as it provides the best results.	73

3.5	Impact of supervision on patch retrieval. PhilippNet is trained on surrogate classes, while PhilippNet-Rome is trained on a larger set of RomePatches-Train, containing either 10K or 100K classes. We see that retraining improves performance provided enough data is given. Supervised CNNs are still below the SIFT baseline, as well as the unsupervised CKNs.	74
3.6	Image retrieval results. Results are in mAP except for UKB where we measure the average number of true positives in the first 4 results. CKN-mix is the result of the concatenation of the VLAD descriptors for the three channels.	76
3.7	Influence of the pretraining dataset on image retrieval. With the same architecture (AlexNet), training is conducted either on ILSVRC'12 data or on the Landmarks dataset. Using semantic information related to buildings and places yields improvements for Oxford, but not for Holidays.	76
3.8	Impact of dimensionality reduction by PCA+whitening on the best channel for each dataset. PCA to 4096 dimensions.	77
3.9	Dense results on Holidays. Right hand side of the table are CKN descriptors. "Same parameters" correspond to CKNs that have the same parameters as Hessian-Affine ones, yet learned on densely extracted patches. "Changed pooling" have pooling size increased by one at each layer (CKN-raw is unchanged as it only has one layer, and already gives good performances).	78
3.10	Dense keypoints on the Oxford dataset. "Crop" indicates the protocol where queries are cropped to a small bounding-box containing the relevant object, while "no crop" takes the full image as a query. For CKN, parameters have increased pooling sizes for dense keypoints, as on Holidays.	78
3.11	Comparison with state-of-the-art image retrieval results for global descriptors.	79

Chapter 1

Introduction

1.1	Goals	4
1.2	Context	10
1.3	Contributions	14

Et j'ai vu quelquefois ce que l'homme a cru voir !
Arthur Robot

The ancient thinkers used to call μίμησις (mimesis) every form of expression which had the goal to imitate reality, such as paintings and other artistic works. Our modern images fall into that category. The difficult translation of the word – between *mimickery* and *representation*– has started quarrels between philosophers, that still perdure today. It revolves around the following questions: how can we make sense of the real world from mere visualizations? Are representations a degradation of reality, or do they offer accurate knowledge from which one can extract insights of concepts? As it turns out, most computer vision tasks face a very similar problem. Images are two-dimensional projections of a three-dimensional world, and several semantically similar concepts can have different expressions, such as different models and colors for cars or species and genders for animals. Plato was among the first to denounce the dangers of false representations, distinguishing the μίμησις εἰκαστική (mimesis eikastike), true to its model, from the μίμησις φανταστική (mimesis phantastike), which distorts reality. The task of understanding concepts in images can also suffer from multiple hindering factors, such as mislabelings or unplausible situations.

Do you not perceive how far images are from possessing the same qualities as the originals which they imitate?

Plato, Cratylus 432d, Trans. H. N. Folwer.

In image understanding, very early works had the Platonic goal of defining constraining models that would underlie representations, and forced these priors onto observations. A notorious example is the constellation models of Fischler and Elschlager [1973], which cast the human face into semi-rigid models with spring constraints. Models from two different faces are then matched together to determine if the faces are from the same person.

More recently, image recognition systems have still not been exempt of design choices that incorporate robustness to some specific variability. As an example, using only grey-level images can be useful to recognize buildings, for which color is usually non-discriminative. This strategy can however be harmful for animal recognition, whose notable differentiating features can be solely based on colorful parts. Very similarly to the prisoners of Plato's well known Allegory of the Cave (Fig. 1.1), learning frameworks require the incorporation of some external knowledge of the problem provided by the scientist (the philosopher who returns to the cave after escaping) to correctly figure out visualizations (the shadows), in the form of well-designed robustness in their architecture.

On the other hand, Plato's most famous disciple and part-time contradictor Aristotle was not so vindictive as to the deceptive nature of representations. He postulates that many relevant characteristics can be learned from observation, and



Figure 1.1: The Allegory of the Cave, according to Cornelis van Haarlem, 1604, Albertina, Vienna. Learning systems must figure out concepts from projections.

indeed his whole approach to science is more about observing Nature, rather than shaping it.

The reason why we enjoy seeing likenesses [images] is that, as we look, we learn and infer what each is, for instance, “that is so and so.”

Aristotle, Poetics, 1448b, Trans. W. H. Fyfe.

Similarly, modern computer vision has looked into reducing the level of rigidity in the models, having architectures with millions of learnable parameters that are directly determined from data. Collecting this data is however not always an easy task. It requires human labor which is usually long and expensive, and raw images which are not always available, as for instance for rare species of animals, or copyrighted material. For some tasks such as image retrieval (see section 1.2 for details), where the goal is to learn to match similar objects, it is desirable to have no a priori knowledge of the various instances to retrieve. This way, for instance, a building that would be constructed after the release of the program could still be properly handled.

When data is scarce, one must rely on general knowledge of the task and enforce robustness to appropriate variability, in order to ensure decent recognition performance. While many works optimize the robustness of their models by hand, the purpose of this thesis is to show that it can also be inferred from data.

1.1 Goals

The goals of this thesis are twofolds. First, we design systems that are robust to some geometric transformations on the input image (e.g., crops and rotations). We propose a framework to automatically determine to which transformations and by what amount the system must be robust to maximize performance. Our second goal is to reduce one of the perduring human cost in computer vision: annotation. Indeed, using carefully designed learning methods that make use of unsupervised or weakly-supervised data is able to significantly reduce the number of required labeled examples.

1.1.1 Identifying and Enforcing Robustness

We define the robustness of a system as its capacity to predict a similar output under small variation of input. Typical image recognition systems need to encompass two forms of robustness: to *geometric* and to *semantic* variability. The first one models all expected changes in the mode of capture of the 3D scene or object. Indeed, different images of the same entity are subject to differ from viewpoint, camera model, illumination conditions, occlusions, and background changes. The second source of variability is defined by the size of the *class* of entities that are allowed to correctly match. For instance, for a coarse classification task where “bird” defines one class, the system must learn to match together all bird species. Even for very narrow subclasses such as “lion”, sexual dimorphism can cause significant visual changes in appearance for instances of the same class. See Figure 1.2 for examples of expected variability in images.

A learning system that has enough capacity (usually measured in terms of VC dimension [Vapnik, 1998]: the largest number of points a classifier can arbitrarily separate) and provided with unlimited independent and identically distributed data can learn to be discriminative even in presence of these variabilities. Yet, annotated images are costly and systems are limited in the number of images they can reasonably handle. To reduce the number of labeled images, one can look into directly enforcing specific robustness into vision systems. Semantic variability is usually dependent of the class and cannot in general be learned without examples. A notable exception is the field of attributes (e.g., Lampert et al. [2009]), which looks into finding shared semantic characteristics between different class instances, and can be used for zero-shot learning [Lampert et al., 2013, Akata et al., 2015], which is the task of classifying images without having a single example available. This is, however, not the focus of our work and we choose to instead look into geometric variability, which does not depend of the class as it is only linked to acquisition parameters. It is possible to design architectures that are robust to certain geometric transformations. The simplest example is to preprocess images

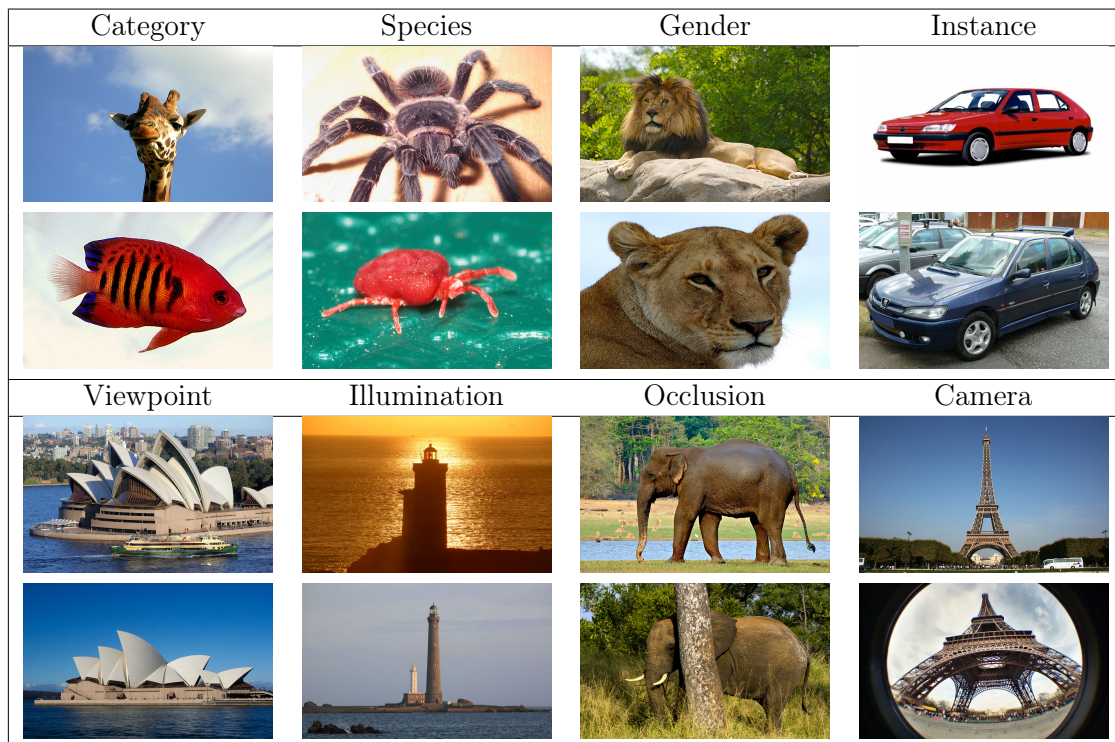


Figure 1.2: Semantic variability (top). A same class can contain several sub-categories (e.g., “animals” on the left), species (e.g., “spiders” middle-left), genders (“lions” middle-right) or even different expressions of the exact same model (“Peugeot 306” on the right). Geometric variability (bottom): even the same objects can significantly vary in appearance due to acquisition parameters. From left to right: viewpoint variation, lighting changes, occlusions, and camera model. Images from the ImageNet dataset.

into grayscale ones to make them invariant to hue and saturation changes. Manually enforcing the necessary level of robustness is not practical, and this thesis looks into methods to directly learn the levels of geometric robustness that are required by the task. We study two such tasks, image retrieval and classification, which we briefly introduce in the following.

1.1.2 Robustness in the Context of Image Retrieval

The first task on which we focus is called *instance-level retrieval*. It consists in finding in a large dataset instances of an object that is known from a reference image called *query*. One usage scenario could be a photo-tourist who forgot the name



Figure 1.3: Example of four different classes of matching objects in the Rome dataset, introduced in this thesis [Paulin et al., 2016]

of a building he captured and wanted to know its name in webpages containing the same instance, or who needed more images of the same element to enrich his collection. Only images of the exact same object are supposed to match; buildings of a similar architectural design, for instance, do not fall into that category. This differs from work on *semantic retrieval* or context retrieval, where semantic groups such as “birds” can match, and which have been studied for instance by Chatfield and Zisserman [2012]. In contrast to classification, no knowledge of the classes or in this case objects is required beforehand. This allows for instance the system to be able to find instances of objects that were created after the retrieval program was deployed. Since classes consist of the same objects, the only variability to which the system must be robust is of the geometric kind. Two sorts of retrieval datasets currently exist. Ones for which acquisition of different images are sequential and with similar settings. This is the case for instance for the popular Holidays dataset [Jégou et al., 2008] and University of Kentucky Benchmark (UKB, Nister and Stewenius [2006]). These datasets usually display significant viewpoint variability. The other category of datasets consists of images crawled from popular image hosting websites, such as Flickr¹. Popular examples include Oxford and Paris [Philbin et al., 2007], and are usually more difficult as variability to camera model and illumination due to time of day is added to the viewpoint changes. See Figure 1.4. Our work (see Chapter 3) introduces a new dataset of the latter category, with images from the city of Rome. See Figure 1.3 for examples of matching images in this dataset, illustrating the large visual variation between images of the same object in instance-level retrieval.

A retrieval system is evaluated on three different criteria: accuracy (usually measured in terms of mean average precision: mAP), speed, and memory footprint. There is no unbiased global measure that takes into account these three points.

¹www.flickr.com



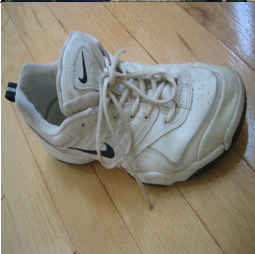


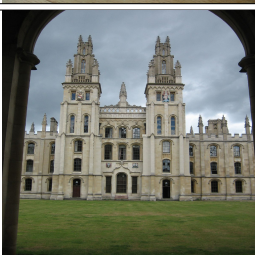

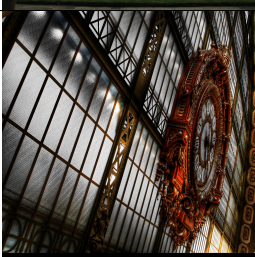
Dataset	Images	Avg ratio Query:Target	Query Examples	Target Examples
Holidays	1491	1:3		
UKB	10200	1:1		
Oxford	5063	1:10		
Paris	6392	1:10		

Figure 1.4: Examples and statistics of the state-of-the-art retrieval datasets: Holidays [Jégou et al., 2008], UKB [Nister and Stewenius, 2006], Oxford [Philbin et al., 2007] and Paris [Philbin et al., 2008].

The usage is therefore to fix speed and memory footprint and compare accuracy only for methods that are similar in terms of those. Thus, methods that work by local descriptor search are not directly compared to methods that rely on global descriptors (see Section 1.2 for more details). Our work on image retrieval only uses the latter: global image descriptors. The goal is to find a robust image representation which allows for two images of the same instance to be close in their induced space. Then, simple nearest neighbor search on the query should return its target.

1.1.3 Robustness in the Context of Image Classification

The second task on which we focus is classification. The goal is to learn to discriminate images into classes, which are known from examples in a training dataset.

In a first stage called *training*, the algorithm or *classifier* is given a set of examples with their class labels. During a second *testing* stage, it must attribute to a set of unknown examples their expected labels. Examples of two popular datasets: Caltech University Birds [Wah et al., 2011] and ImageNet [Deng et al., 2009] are illustrated in Figure 1.5. They illustrate the two popular subtasks of classification. The first is fine-grained classification, which aims at separating truly close concepts such as bird species or car models. The second is large-scale classification, where the sheer number of different classes and examples is in itself a challenge.

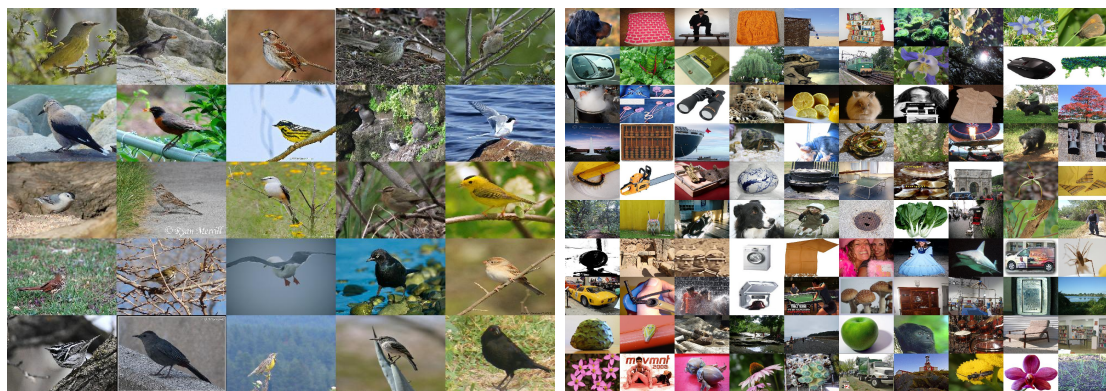


Figure 1.5: Images of two different classification tasks. Left: Birds species categorization. Right: Broad large-scale classification.

Apart from datasets which only have synthetic examples that can be generated at very low cost (see e.g. Dosovitskiy et al. [2014]), training sets are finite. This is a problem as optimizing a classifier with large enough capacity leads to the *overfitting* phenomenon where the system learns examples by heart (large accuracy on the train set), and is unable to give new images a relevant label (low accuracy

on the test set). There are two main possibilities to solve this problem: using more examples or adding *regularization*, as a constraint on w that forces a lower accuracy on the train set. The former is non-trivial in the case of real data, and the latter is often of less quality, because it has no tangible link to the problem (for instance L_2 regularization). One intermediate approach, is to create synthetic new examples using known real ones [Sietsma and Dow, 1991a]. These *virtual* examples, often generated as perturbations (crops, flips, blur, rotations, etc.) of real ones, are used in classification to complement the training set and enforce specific robustness to some transformations. Data augmentation implicates higher training costs, and one of the goals of this thesis is to show how fine-grained classification can use techniques from large-scale methods (such as stochastic gradient descent [Bottou and Bousquet, 2007]) to scale to large numbers of examples and improve performance. Another way to reduce training costs, both in time and memory, is to minimize the number of selected virtual examples, to maximize efficiency. These techniques are further discussed in Chapter 2.

1.1.4 Towards Reducing Annotation Costs

An important goal of computer vision is to develop fast and light methods that are able to scale to large datasets without requiring great amounts of computing power or storage. Implementing algorithms on GPU (e.g. SIFT [Wu, 2007]), and Product Quantization [Jégou et al., 2011] are related examples for speed and memory optimization respectively.

But looking at computer vision as a whole, one important cost that must not be overlooked is the cost of human annotation. This directly impacts the performance of the whole process and is not on the same level as other performance issues. In many tasks, a preliminary phase of costly manual labor prior to learning is required, in order to generate training and testing data for computer vision algorithms. To do so, scientists often resort to crowdsourcing through websites such as Amazon Mechanical Turk, or undergraduate students. Figure 1.6 shows how two of the most popular benchmarks for image classification were gathered, and illustrate the high cost in resources (both monetary and human) they required. The advent of large-scale learning especially with deep architectures has recently dramatically increased the need for manually annotated examples.

This is indeed unsatisfactory due to the variability of results, the required offline time, and the monetary cost. For the worker, this is also excruciatingly boring. One goal of this thesis is to show that we can reduce the amount of required labor through careful design.

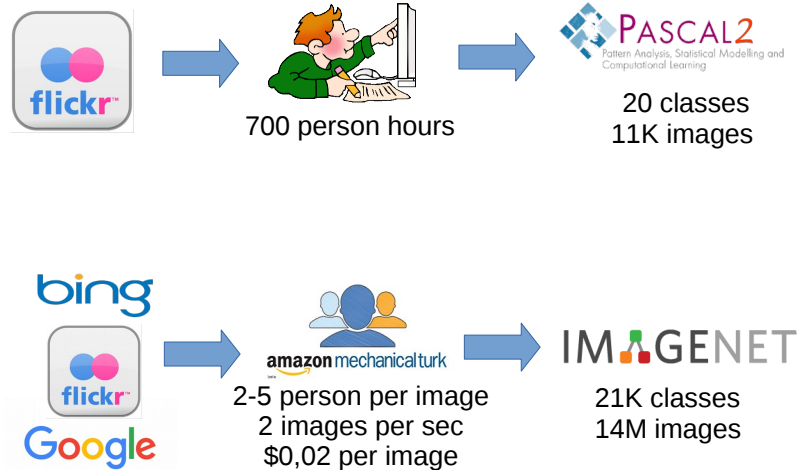


Figure 1.6: The process of collecting the PASCAL VOC and ImageNet datasets. While the early PASCAL VOC Challenge relied on computer scientists to annotate data, the need for large scale benchmarks such as ImageNet with thousands of classes and millions of examples required crowdsourcing.

The implements of labour, in the form of machinery, necessitate the substitution of natural forces for human force, and the conscious application of science, instead of rule of thumb.

K. Marx, Das Kapital. Trans. Samuel Moore and Edward Aveling.

We achieve this by using virtual examples (see Chapter 2) or unsupervised learning—a method which requires no groundtruth (see Chapter 3).

1.2 Context

To motivate the goals of this thesis, we give a brief overview of the evolution of robustness in image classification and retrieval. One of the main difficulty of computer vision is expressed in the Burns theorem [Burns et al., 1993]: “there is no feature of n projected points for any n that is both a general-case view invariant and nontrivial”. It is therefore impossible to find truly invariant image representations, and this does not even concern the image classification problem

and semantic variability. 3D-view invariants do not exist, but it is possible to design quasi-invariants that are robust to small changes in viewpoint.

For instance Binford and Levitt [1993] show that homography invariants at image level are quasi-invariants for perspective transformations. Most typical retrieval problems, however, cannot only be solved by quasi-invariants to viewpoint changes, and robustness to many other factors such as illumination or occlusions must be enforced. In the following section, we give an overview of methods that explored which other invariances could be attained, especially at patch level. We then describe how methods evolved to robust descriptors and methods, and finish by the most recent development in computer vision which are modern neural networks, and the level of robustness they brought.

1.2.1 Strong Invariances

The early days of image recognition were marked by the desire to find representations that were invariant to some transformations. Due to acquisition methods, as well as the wish to be invariant to hue and saturation, images were almost exclusively pre-processed into grey-level versions. A large field of study was devoted to find reproducible locations in images, called *interest points*. These points are distinctive enough so that all projections of a 3D location whose projection in one view is an interest point, are also interest points. This was achieved with several key processes. Such an interest point can only lie on a textured surface, so as not to be mistaken with its close neighbors. Two specific characteristics for attaining translation invariance can be exploited in images: *corners*, as intersection of edges with a significant angle, and *blobs*, as small and bounded uniform regions. Examples of the former include the Harris [Harris and Stephens, 1988] and Hessian [Lindeberg, 1998] detectors. The latter is typically used with a Difference- or Laplacian-of-Gaussian [Lowe, 1999] detectors or Maximally Stable Extremal Regions [Kadir et al., 2004]. To make region extraction invariant to zooms, algorithms use a scale selection method such as the one of Lindeberg [1998]. Last, robustness to local perspective changes can be obtained by the affine normalization approach of Mikolajczyk and Schmid [2002].

Once these points are computed, small regions (between 10 and 100 pixels square areas called *patches*) are extracted around them. The goal is to design invariant *local descriptors*, defined as mappings between these regions and a certain Euclidean space, with the property of being identical under some transformations. Usually relying on image gradients, which are naturally invariant to uniform illumination changes, and higher order moments, these local features are used to match the aforementioned interest points detected in different views. Typical invariant representations include the differential invariants of Florack et al. [1992]

and their extension, the steerable filters of Freeman and Adelson [1991]. Both representations are rotation-invariant.

The design and proofs of these invariants relies on theoretical results related to Lie groups, but their hypotheses are not always respected on real data, a fact that is partly due to pixelization.

1.2.2 Pooled Invariances

One of the major breakthroughs in image description was the invention of Lowe [2004] with the Scale-Invariant Feature Transform (SIFT). It started a whole new field of research with the introduction of *pooled* invariants. On top of invariant features such as gradient orientations, a pooling step –spatially aggregating features into one that represents an area, taking e.g. their sum or their max– has the effect of no longer having invariant features, but rather *robust* ones that keep closeness to their origin under small transformations.

Many descriptors that work in a similar fashion to SIFT were crafted, e.g., GLOH [Mikolajczyk and Schmid, 2005], SURF [Bay et al., 2006], or Daisy [Tola et al., 2010]. At the same time, realistic benchmarks to test performance of all local descriptors were designed such as the one of Mikolajczyk and Schmid [2005]. Its notable conclusion is the outstanding experimental performance of SIFT compared to the previously described *invariant* descriptors.

The success of these pooled descriptors in pairwise point matching leads to works exploring efficient ways to design similarity measures between images. Building on sets of local descriptors such as SIFTs, typical global representations aggregate or *pool* them using Bag-of-Words [Sivic and Zisserman, 2003], VLAD [Jégou et al., 2010] or Fisher Vectors [Sánchez et al., 2013]. They were used for both retrieval [Jégou et al., 2012] and classification [Perronnin and Dance, 2007] tasks.

1.2.3 CNNs: Learning Robustness at Different Scales

The recent evolution of computer vision can be read in the results to the ImageNet Large Scale Visual Recognition Challenge, plotted in Figure 1.7. The year 2012 is marked by a large increase in performance, operated by a paradigm shift in methodology. While previous approaches were based, as described in the previous paragraphs, on local descriptor aggregation, the best performing ones, called *deep learning*, used Convolutional Neural Networks (CNN) to perform classification, a pioneering work in this field being the one of Krizhevsky et al. [2012].

The changes with respect to the previous approach were numerous. First, the architecture in itself, consisting of stacked layers of convolutions interleaved with pooling, although reminiscent of previous models, was radically different. Fully-connected classifiers were also shown to perform better than linear ones, even for

Fisher Vectors [Perronnin and Larlus, 2015]. Note that both these claims were invalidated by the success of Residual Nets [He et al., 2015] which used neither pooling layers, nor fully-connected classifiers. Other major changes to typical neural networks architectures included rectified linear units, dropout [Hinton et al., 2012], and data augmentation.

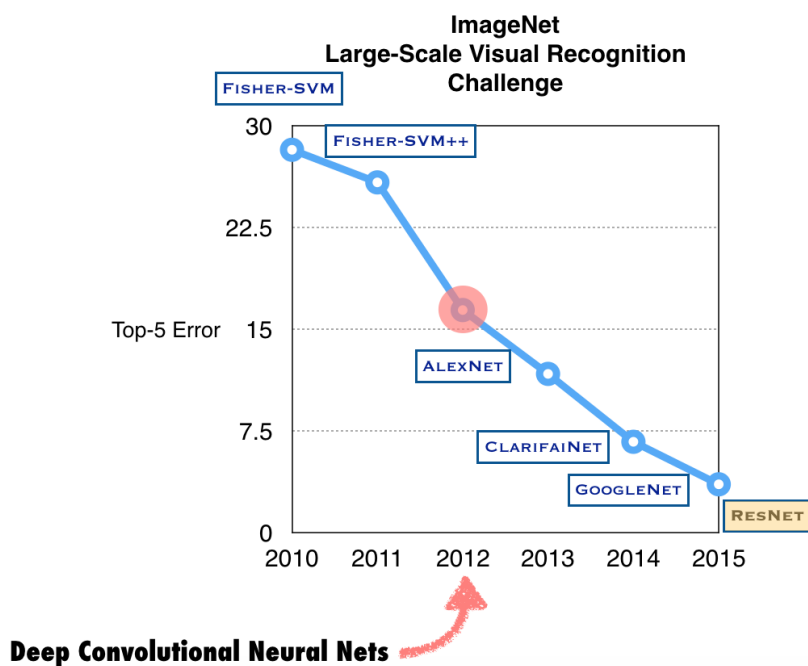


Figure 1.7: Evolution of the best performing method on the ImageNet dataset.

Although not fundamentally new and relying on methods that were developed several decades before [LeCun et al., 1998], it was the first time they were applied to realistic image recognition problems.

The success of deep learning in image classification was however not directly transferred to the task of image retrieval, a fact that can be explained by two factors. First, global image representations give suboptimal (albeit faster) results compared to approaches that utilize local descriptor indexing. Second, the absence of training data forces the use of transfer techniques and do not give state-of-the-art results. Two notable exceptions is the work of Gong et al. [2014] which holds the state of the art for global descriptors for 2014 by pooling CNN-encoded local image regions, with a mAP of 80.2%, and the work of Perronnin and Larlus [2015] with 84.7%, which solely takes from deep learning the multi-layer fully-connected classifier. The evolution of the best performing methods on the Holidays dataset is given in Figure 1.8.

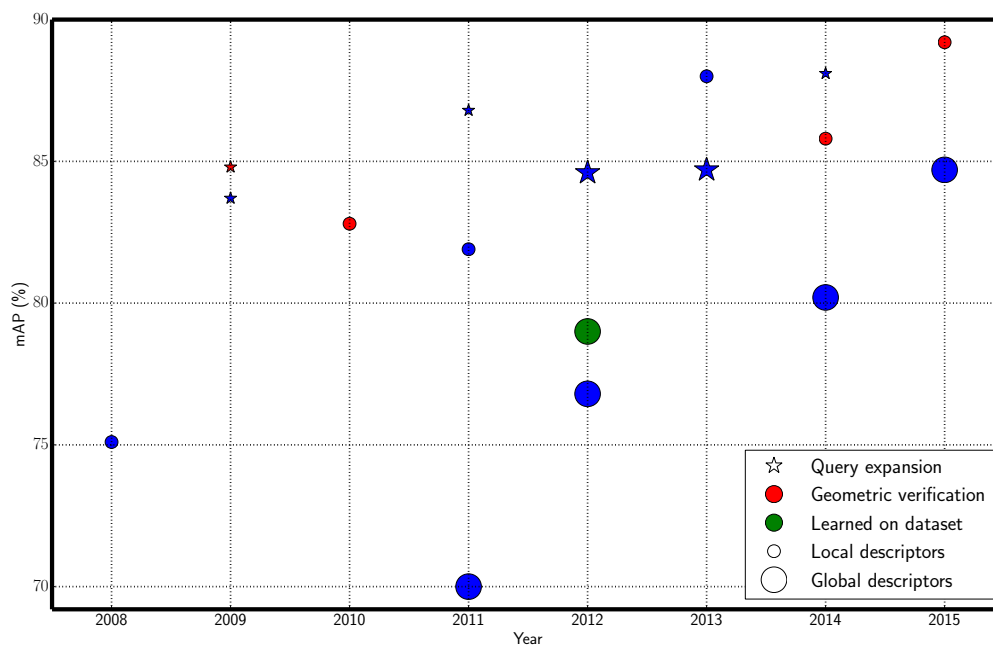


Figure 1.8: Evolution of the best performing methods on the Holidays dataset. Regular methods are marked with circles. Small markers are methods that rely on sets of local descriptors (more costly), where large markers are for global descriptors. Stars represent methods that assume the database may have several target instances and use query expansion to utilize them. Red markers use geometric verification posterior to retrieval. Green markers allow methods to perform unsupervised learning on the dataset itself, effectively preventing them to generalize to brand new objects.

In this work, we focus on the first and last novelties of the CNN: the particular convolutional architecture in Chapter 3 and data augmentation in Chapter 2. We believe that breaking down the individual components can help better understand the success of the method. We now give a brief overview of these two techniques and our contributions for them.

1.3 Contributions

This thesis addresses the problem of robustness in image classification and retrieval systems. Here, we briefly describe the two main contributions that will be detailed in Chapter 2 and Chapter 3.

1.3.1 First Contribution: Selecting Virtual Examples

When training data is scarce, a popular technique is to generate small perturbations in the forms of plausible deformations (crops, flips, rotations, etc) and use them to teach robustness to the learning system. This process is called data augmentation, and the new data “virtual examples”. We focus Chapter 2 on the study and optimal selection of transformations that lead to virtual examples that are beneficial to classification. This work was published in CVPR 2014. Our contributions are threefold:

- We show how to select a small set of transformations to create virtual examples while maximizing performance. Selecting a small number of virtual examples also allows to reduce memory footprint and training time. An overview of our method is given in Figure 1.9. Our algorithm, called Image Transformation Pursuit returns a quasi-optimal set of candidate transformations, while staying tractable. This effectively allows us to design a system that is robust to geometric transformations, which are automatically determined from data and require no manual selection.

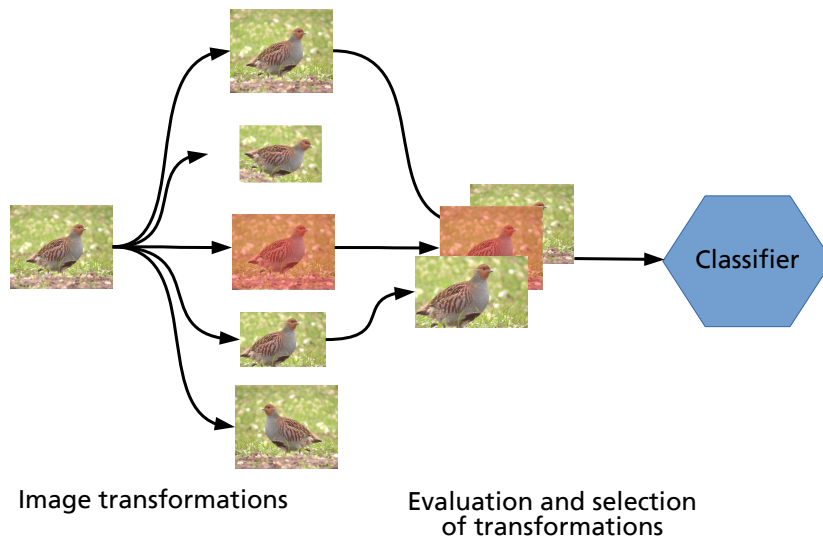


Figure 1.9: Our approach to boost classification performances with virtual examples. Each image is transformed through several geometric transformations. Some specific transformations are selected to maximize performance. In a last stage, the classifier is trained with all selected transformations.

- We show that jittering examples with transformations improves performance by a fair margin provided it is used both at train and test time. While mostly used at train time, Krizhevsky et al. [2012] were the first to also use it at test time, but did not give details on its contribution.
- We report state-of-the-art performance on several widely used fine-grained categorization datasets such as Caltech University Birds, Aircrafts and Cars. At the time of publication, our method also achieved state of the art with Fisher Vectors on ImageNet.

1.3.2 Second Contribution: Convolutional Architectures for Patch Description

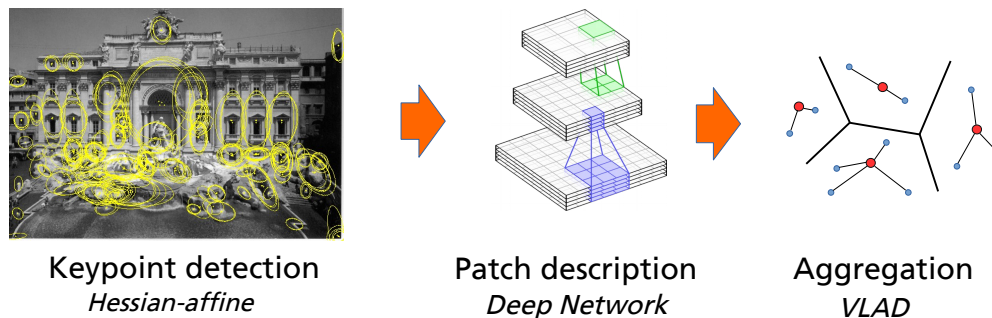


Figure 1.10: Our approach for image retrieval with local convolutional descriptors. In a typical pipeline with keypoint detection, region description, and aggregation, we propose to replace the commonly used SIFT descriptor with deep architectures for patch description.

We use the following three steps for our image retrieval procedure: interest point detection, local region description and aggregation. While the second step is usually achieved with handcrafted descriptors such as SIFT, we propose in this work to replace them with a descriptor build with the CNN framework [Mairal et al., 2014b]. Figure 1.10 gives an overview of our approach. Our contributions are the following:

- We propose a new patch descriptor based on the Convolutional Kernel Networks of Mairal et al. [2014b]. These descriptors do not require manually labeled data, as their parameters are entirely determined through unsupervised learning. Their level of robustness is directly determined from the patch matching dataset.

- To do so, we introduce a new patch/image dataset, named RomePatches, on which we are able to link performances for both tasks. This dataset is automatically generated using 3D reconstruction, effectively introducing a general methodology for benchmarking patch description in the context of image retrieval.
- Most works that focus on learning patch descriptors only perform evaluation on patch retrieval tasks. We show that better performance in patch matching leads to better performance in image retrieval as well.
- At the time of publication, our descriptors outperformed the state of the art on our dataset, as well as on standard patch retrieval and image retrieval benchmarks. We especially show that they outperform SIFT, transferred off-the-shelf CNN features, as well as supervised CNNs. Compared to the latter, they require an order of magnitude less training time, and no labelled data.

Chapter 2

Selecting Virtual Examples for Image Classification

2.1	Introduction	19
2.2	Related Work	20
2.3	Virtual Examples: Generation and Selection	23
2.4	Experiments	31

SOCRATES: *Let us take any common instance; there are beds and tables in the world –plenty of them, are there not?*

GLAUCON: *Yes.*

SOCRATES: *But there are only two ideas or forms of them –one the idea of a bed, the other of a table.*

Plato, The Republic, book X. Trans. B. Jowett.

2.1 Introduction

The focus of this chapter is image classification. This is a very challenging problem because objects of the same class may exhibit large variations in appearance. Such intra-class variations fall into two categories. The *intrinsic* variability corresponds to the fact that two instances of the same object class can be visually different, even when viewed under similar conditions (*e.g.*, different zebras have different patterns). The *extrinsic* variability refers to those differences in appearance that are not specific to the object class (*e.g.*, different viewpoints, lighting conditions, image compression).

In what follows, we use the term “invariant” in a loose manner, implying that a learning system is invariant if, for a given object, the predicted label remains unchanged for all possible variations of images of the class. Following the taxonomy of Bishop [1995b], there are three approaches to building invariant learning systems from finite training sets: (i) generating virtual training examples by applying transformations to the original samples to account for the variations that are expected at test time [LeCun et al., 1998, DeCoste and Burl, 2000, Decoste and Schölkopf, 2002, Simard et al., 2003, Krizhevsky et al., 2012]; (ii) designing a feature representation which is inherently invariant to the variations to be expected [Tuytelaars and Mikolajczyk, 2007]; (iii) embedding the invariance in the structure of the learning system, a popular example being convolutional nets [Bengio, 2009, Sohn and Lee, 2012]. Examples of (ii) include invariant kernels for support vector machines [Schölkopf and Smola, 2002] which are limited to invariance to transformations satisfying a particular Lie group structure. On the other hand, examples of (iii) usually correspond to learning architectures intended to mimic biological visual systems [Bengio, 2009] that exhibit attractive invariance properties; it remains unclear however how to reverse-engineer such architectures to build-in relevant invariances for a particular task.

We propose here to explore virtual example generation (option (i)), by explicitly generating virtual examples at training and at test time. While it might be difficult to generate virtual samples that reflect intrinsic class variations (except for specific object classes such as pedestrians [Marín et al., 2010, Pishchulin et al., 2011]), it is possible to generate virtual samples that simulate extrinsic variations by applying simple geometric and colorimetric transformations. In this work, we focus on such transformations, as they have been shown to increase classification accuracy, especially on simple tasks such as digit recognition [LeCun et al., 1998, DeCoste and Burl, 2000, Decoste and Schölkopf, 2002, Yaeger et al., 1996], and, more recently, on ImageNet [Krizhevsky et al., 2012].

We believe that the selection and weighting of an optimal set of transformations is essential for the success of the approach. Indeed, transformations that are too conservative (*e.g.*, removing the first column of pixels) have little if any impact,

whereas they come with significant computational overhead both at training and testing time. On the other hand, transformations that are too extreme (*e.g.* for digits, a strong rotation that would transform a 6 into a 9) will lead to unlikely images and may decrease classification accuracy. We are not aware of any principled and automatic approach to weighting a set of transformations that lead to higher accuracy, except manually by trial and error. This might be acceptable when the number of possible transformations is limited, for instance when dealing with small (*e.g.* 256 pixels) black-and-white images of digits. However, such a manual selection process is not applicable when there is a large number of possible transformations, which is a must when dealing with realistic datasets.

In our proposal, each image can be represented by a set of descriptors, each corresponding to the features extracted for the transformed version of the image. We are faced with a classical computer vision problem: how to aggregate a bag of descriptors into a single image-level representation. We propose a principled approach for utilizing transformations, termed *Image Transformation Weighting* (ITW). This approach computes an optimal set of weights for each transformation by modelling them as latent variables. When the dataset size or the feature dimension are too high for such an approach, we further constrain this model to select weights in $\{0, 1\}$, and derive an approximation called *Image Transformation Pursuit* (ITP). ITP computes a set of optimal transformations from a large “dictionary” of transformations (see Fig. 2.1), by iteratively and greedily selecting one transformation at a time. ITP is reminiscent of pursuit algorithms such as matching pursuit or basis pursuit [Mallat, 2008], which compute a signal approximation from a dictionary by iteratively selecting one atomic element at a time from the dictionary. We show that ITP gives similar results to ITW, at a fraction of the cost.

We report results for ITP on three public benchmarks for fine-grained classification: the CUB dataset of Birds [Wah et al., 2011] as well as on the Aircrafts dataset and the Cars dataset that were both featured in the FGComp’13 challenge¹. On all datasets we report significant improvements for Fisher as well as deep convnet features. Our method holds state-of-the-art for non-finetuned bilinear features. One important conclusion of our work is that it is crucial to *apply transformations both at training and test time for best performance*.

2.2 Related Work

Augmentation of the training set by adding virtual examples has received considerable interest. In the context of image classification, data augmentation is an attractive empirical approach to learn invariances. There are two main strategies:

¹<https://sites.google.com/site/fgcomp2013/>

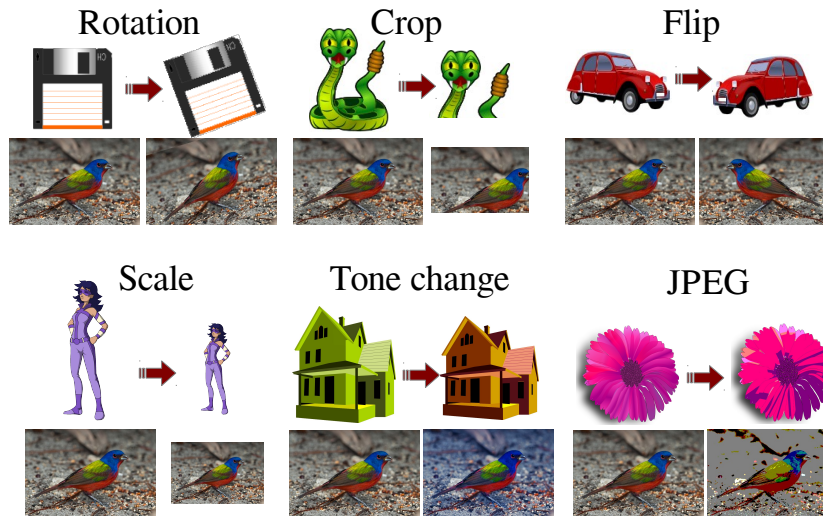


Figure 2.1: Illustrations of image transformations, on simple icons and on an image from the CUB dataset.

i) generating virtual examples by transforming the visual descriptor; ii) generating virtual examples by transforming the original image, prior to visual descriptor extraction. We also review previous works that enforce geometric priors in classification, without resorting to data augmentation.

2.2.1 Corrupting features with noise

The first approach consists in adding noise to descriptors according to particular probability distributions. Originally introduced by Sietsma and Dow [1991b], this approach can be shown to be equivalent, up to first-order, to adding a particular regularization term to the empirical risk objective. For instance, adding Gaussian noise with mean 0 and variance σ^2 to least-square regression is equivalent to ridge regression with penalty term σ^2 . Bishop [1995a] shows that noise injection with a least-squares penalty is first-order equivalent to adding a Tikhonov regularizer to the objective. An [1996] suggest that noise injection can induce smoothness of the prediction function with respect to the input data. More recently, marginalized auto-encoders [Chen et al., 2012] rely on a similar idea. Maaten et al. [2013] give an analytical expression of the noise injection effect for several noise probability distributions as well as several losses, in particular the one corresponding to drop-out noise [Hinton et al., 2012, Krizhevsky et al., 2012, Baldi and Sadowski, 2013, Wan et al., 2013].

2.2.2 Transforming images prior to description

The analytic noise injection strategy requires little to no a-priori knowledge on the nature of the examples, and is therefore easy to implement. Yet, it can be difficult to interpret from a computer vision point of view, as there is no unique mapping from the descriptor space to the image space (such a mapping might not even exist!) and the image corresponding to the corrupted descriptor is thus usually untractable. Therefore the choice of the noise probability distribution can be unclear. The second strategy works directly at the image level, using geometric transformations to generate perturbations of an image. For instance, LeCun et al. [1998], Decoste and Schölkopf [2002], Niyogi et al. [1998], Krizhevsky et al. [2012] generate virtual images from the original ones using plausible transformations such as crop and flip. How to use these virtual images to help classification remains an open problem. While we focus our work on three simple strategies: data augmentation, feature concatenation and match-kernel, many other techniques exist. For instance, Abu-Mostafa [1995] proposes to add to the classifier a penalty that enforces similar decisions for images and their transformations, an idea which is similar to the transductive inference of Vapnik [1998] or the auto-encoder of Vincent et al. [2008]. In a similar fashion, these invariances can be learned during training, using tangent propagation [Bishop, 1995b], or embedded into kernels through jittering [DeCoste and Burl, 2000]. Standard approaches for equipping learning architectures with invariance is tangent distance and its cousins [Simard et al., 1992, Keysers et al., 2007, DeCoste and Burl, 2000, Huang et al., 2012], which correspond to a manifold assumption on the input data.

In contrast to the virtual examples strategy, this strategy is more intuitive and easier to interpret, as one works directly with realistic transformations. Yet, generating virtual images and extracting their features is significantly more expensive than adding noise directly on the features. An exception is the digit recognition task, where computing elastic deformation fields can be performed in an elegant and computationally efficient manner [Loosli et al., 2007]. Our approach applies the virtual images strategy by greedily selecting a limited number of transformations from a large dictionary, hence boosting performance without compromising scalability. Furthermore, it requires potentially no prior knowledge, as it can work with very large sets of transformations.

Finally, it is worth mentioning that most previous works perform only transformations on the training set [LeCun et al., 1998, Decoste and Schölkopf, 2002, Krizhevsky et al., 2012]. A few approaches also considered transforming the test images [DeCoste and Burl, 2000, Krizhevsky et al., 2012].

2.2.3 Geometric priors for classification

Generating transformations of images and using them as virtual examples can be seen as a way to enforce robustness to these transformations. Several works have also focused on enforcing geometric robustness, without using virtual examples. Early work on visual features focused on explicitly enforcing invariances to rotation, translation, or other Lie-group transformations [Tuytelaars and Mikolajczyk, 2007, Szeliski, 2010], later culminating with the SIFT patch descriptor [Lowe, 2004]. Invariant features can also be emulated using invariant kernels [Schölkopf and Smola, 2002, Decoste and Schölkopf, 2002, DeCoste and Burl, 2000], but training and testing with non-linear kernel SVMs is challenging on large-scale image datasets, except for specific kernels that admit explicit embeddings. Kumar et al. [2010] assumes that each image has a unique representation in the set of its transformed versions, and model it as a latent variable. In this work however, all our transformations are shared between images, to simplify selection as well as testing. In a similar fashion, Jaderberg et al. [2015] introduce a Spatial Transformer layer in a CNN that explicitly applies a transformation to a feature map. Its parameters are directly learned on data. In contrast to SPNs, our approach can leverage a large class of image transformations, potentially enforcing a larger class of invariances.

2.3 Virtual Examples: Generation and Selection

We start by describing the family of transformations we apply to train and test images (Section 2.3.1). We follow by introducing several schemes to make use of the virtual examples (Section 2.3.2). We then explain the proposed Image Transformation Weighting (ITW) algorithm and its approximation: Image Transformation Pursuit. Finally, we discuss the score aggregation stage when transformed images are generated at test time (Section 2.3.3).

2.3.1 Image Transformations

We distinguish two types of transformations, geometric ones, that transform the initial image prior to descriptor extraction, and analytic ones, that are applied as noise on the features. We considered 7 families of geometric transformations (for a total of 40 possible transformations).

Flip. This transformation horizontally mirrors an image. It encodes the natural symmetry of most scenes and objects (text is an exception). Many approaches use flipping to increase their training set size without prior knowledge, see for instance Krizhevsky et al. [2012], Gavves et al. [2013].

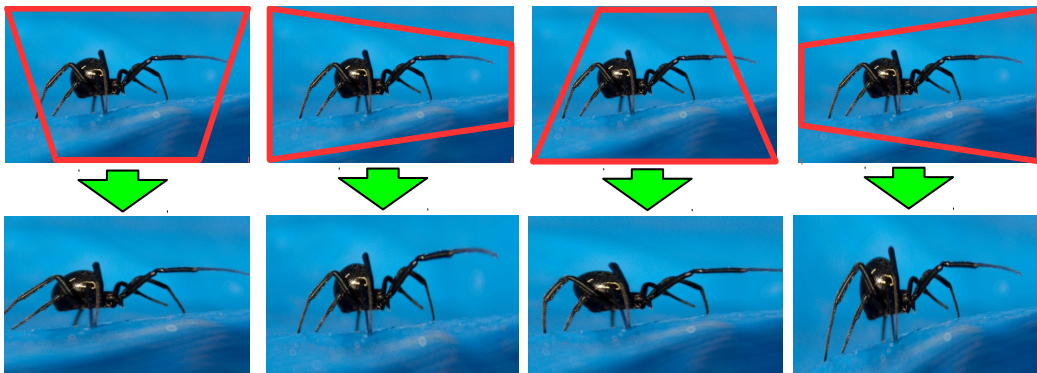


Figure 2.2: Four of the eight homographies we use. The four others we consider are similar, but with a broader window.

Crop. We consider the restriction of an image to a specific sub-window. We use 10 different crops, defined relatively to the image size. Prior to training/testing, we randomly draw 10 sub-windows (x_0, y_0, x_1, y_1) with $(x_0, y_0) \in [0, 0.25]^2$ and $(x_1, y_1) \in [0.75, 1]^2$.

Homography. To model viewpoint changes, we consider homographic transformations of the initial images. We restrict ourselves to horizontal and vertical pannings, and select 8 homographies (see Fig. 2.2).

Scale. We scale down images using bilinear interpolation. We consider 5 downscaling factors, respectively $(\sqrt{1.5})^n$ with $n \in \{1, \dots, 5\}$.

Colorimetry. We consider colorimetric transformations. In the same fashion as Krizhevsky et al. [2012], we compute the covariance matrix of the RGB components on the whole dataset. Denoting $\lambda_1, \lambda_2, \lambda_3$ and p_1, p_2, p_3 respectively its eigen-values and -vectors, we add to each pixel $p \in [0, 255]^3$ a quantity $\varepsilon_1 \lambda_1 p_1 + \varepsilon_2 \lambda_2 p_2 + \varepsilon_3 \lambda_3 p_3$. We generate three different random triplets $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ sampled from a normal distribution $\mathcal{N}(0; 0.1)$.

JPEG Compression. Despite being designed to minimize the change observable by a human, strong JPEG compression can have a dramatic effect on descriptors. To account for variations in image encoding, we consider three different levels of JPEG compression: 30, 50 and 70.

Rotation. To be robust to camera orientation, we rotate images around their centers. To avoid introducing blank corners or cropping the image, we adopt the following method. Rotated images are incrementally pasted onto one another, by steps of 1 degree. The resulting images therefore contain parts of the original image in their corners. We consider 10 rotations of $\{-15, -12, \dots, -3, +3, \dots, +12, +15\}$ degrees.

2.3.2 Virtual Examples

In the following chapter we denote by $(x, y) \in \mathcal{X} \times \mathcal{Y}$ a pair of (image, label) and $\mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T\}$ a set of T possible transformations. For simplicity, we assume \mathcal{T}_1 to be the identity. For an image x , we denote $\phi(x)$ its encoding in an arbitrary descriptor space (for instance Fisher Vectors or CNN features) and therefore $\phi(\mathcal{T}_p(x))$ is the p -th virtual example of image x . Each image is thus represented as a bag of descriptors of its transformations, and the goal of this chapter is to explore the traditional problem of aggregating them to maximize performance.

The traditional classification framework relies on the definition of a loss function $\ell(x, y, w)$ and a regularizer $\Omega(w)$, and aims to solve the following problem:

$$\text{Minimize}_w \mathbb{E}_{x,y} [\ell(x, y, w)] + \Omega(w), \quad (2.1)$$

where w represents the weights of the classifier.

The expectation is then approximated by an empirical risk on a training set \mathcal{Z} of n pairs (x, y) :

$$\text{Minimize}_w \frac{1}{n} \sum_{(x,y) \in \mathcal{Z}} \ell(x, y, w) + \Omega(w). \quad (2.2)$$

Typical choices of losses ℓ can be

- The hinge loss [Vapnik, 1998], leading to the one-versus-rest SVM formulation:

$$\ell_{\text{OVR}}(x, y, w) = \sum_{j=1}^k \max\{0, 1 - y_j w_j^\top \phi(x)\},$$

with (2.3)

$$y_j = \begin{cases} 1 & \text{if } y = j \\ -1 & \text{if } y \neq j. \end{cases}$$

- The multiclass hinge loss:

$$\ell_{\text{MUL}}(x, y, w) = -w_y^\top \phi(x) + \max_j \{w_j^\top \phi(x) + \Delta(y, j)\}.$$

with (2.4)

$$\Delta(y, j) = \begin{cases} 0 & \text{if } y = j \\ 1 & \text{if } y \neq j \end{cases}$$

- The multinomial logistic loss:

$$\ell_{\text{LOG}}(x, y, w) = -w_y^\top \phi(x) + \log \sum_{j=1}^K \exp(w_j^\top \phi(x)). \quad (2.5)$$

We refer the reader to [Akata et al., 2013] for examples of other losses for image classification, including ranking losses. For all previously described losses except the OVR-SVM, we observe that the prediction function that maps an unknown example x to its guessed class \hat{y} can be written as:

$$\hat{y} = \arg \min_y \ell(x, y, w). \quad (2.6)$$

Note that this assumption holds true for the OVR-SVM loss when at least one classifier w_j fires a non-negative prediction $w_j^\top \phi(x)$. We restrict ourselves in the rest of this chapter to losses that verify this property.

In the following, to ease notations, we discard the regularization term Ω . It is straightforward to derive regularized versions of our equations.

To use virtual examples in this classification framework, we introduce weighting variables $\{\mu_1, \mu_2, \dots, \mu_p\}$ in $[0, 1]$ which represent the relative performance of each transformation, and we denote by $\mathcal{L}(x, y, w, \mu)$ an aggregated loss over all transformed examples. We propose several such losses, detailed in the following paragraphs.

2.3.2.1 Feature Concatenation

The first and simplest scheme consists in embedding the selected transformations directly in feature space by concatenating the transformed descriptors. The resulting aggregated loss writes as:

$$\mathcal{L}(x, y, w, \mu) = \ell([\mu_1 \phi(\mathcal{T}_1(x)), \dots, \mu_T \phi(\mathcal{T}_T(x))], y, w). \quad (2.7)$$

Note that in this case, w is T times higher dimensional than the original problem, which can be prohibitive for large-scale problems.

2.3.2.2 I.i.d. Data Augmentation

The second scheme simply adds the generated examples to the training set as if they were new iid samples drawn from the data distribution with the same label as the original example. To incorporate the μ weightings, we assume they correspond to importance sampling factors that only depend on the transformation (and not

the example). The aggregated loss writes as:

$$\mathcal{L}(x, y, w, \mu) = \sum_{p=1}^T \mu_p \ell(\phi(\mathcal{T}_p(x)), y, w). \quad (2.8)$$

When $\mu_p \in \{0, 1\}$, this corresponds to adding or not the examples transformed by \mathcal{T}_p to the total training set.

2.3.2.3 Match Kernel

The last scheme embeds invariance directly into the kernel by considering an instance as a bag of transformed examples, and comparing two examples as the sum of pairwise products between all transformations. The scheme relies on the following kernel:

$$k(x, x') = \sum_{p=1}^T \sum_{p'=1}^T \mu_p \mu_{p'} \phi(\mathcal{T}_p(x))^\top \phi(\mathcal{T}_{p'}(x')). \quad (2.9)$$

This corresponds to summing transformed examples in feature space. The corresponding aggregated loss is:

$$\mathcal{L}(x, y, w, \mu) = \ell \left(\sum_{p=1}^T \mu_p \phi(\mathcal{T}_p(x)), y, w \right). \quad (2.10)$$

2.3.3 Score Aggregation

Each aggregation method detailed in section 2.3.2, comes with its preferred label prediction:

$$\tilde{y} = \underset{y}{\operatorname{argmin}} \mathcal{L}(x, y, w, \mu). \quad (2.11)$$

With linear models as considered in this work, this corresponds to averaging the individual transformation scores: $\sum_{p=1}^T w^\top \phi(\mathcal{T}_p(x))$ in the case of data augmentation and match-kernel. For feature concatenation, this is only the standard score on stacked vectors. However, other aggregations schemes are possible at test time for virtual examples, and we investigate specifically three of them.

2.3.3.1 Averaging.

We use the scores of all virtual examples as independent measures and average them to get a consensus similarly to Krizhevsky et al. [2012]. Denoting by $s_t^{(k)}$ the score given by the classifier for class k to the t -th transformation ($t \leq T$), we

define: $s_{\text{avg}}^{(k)} := \sum_{t=1}^T s_t^{(k)}$. Note that in the case of logistic regression, we use the scores prior to softmax, as the latter is treated separately.

2.3.3.2 Maximum.

We use the transformation that yields the best score: $s_{\text{max}}^{(k)} := \max_t s_t^{(k)}$. It returns the prediction for which the classifier is the most confident.

2.3.3.3 Softmax Aggregation.

The previous maximum scheme can suffer from the presence of outliers, while averaging takes into account transformations with low confidence. A soft-max strategy $s_{\text{softmax}}^{(k)} := \log \sum_{t=1}^T \exp s_t^{(k)}$, provides an intermediate solution.

2.3.4 The Image Transformation Weighting algorithm

We now introduce our algorithm, called Image Transformation Weighting. Its goal is to jointly determine μ and w for any loss that satisfies postulate (2.6). Note that the unconstrained minimization over μ leads to trivial solutions where μ is zero everywhere. We fix this problem by assuming μ lies in the simplex:

$$\sum_{p=1}^T \mu_p = 1. \quad (2.12)$$

Yet, in the case of data augmentation, the optimal solution to direct optimization leads to a solution where only the best performing transformation has a non-zero μ_p weight. To solve these issues, we start by defining the aggregated prediction function of an example x as:

$$\mathcal{P}(x, w, \mu) = \arg \min_y \mathcal{L}(x, y, w, \mu). \quad (2.13)$$

In a similar fashion to structured output learning theory [Nowozin et al., 2015], we handle this problem by using a surrogate loss functional, and solving the following formulation:

$$\begin{aligned} & \text{Minimize}_{w, \mu} \sum_{(x, y) \in \mathcal{Z}} \mathcal{L}(x, y, w, \mu) - \max_j \{ \mathcal{L}(x, j, w, \mu) + \Delta(y, j) \} \\ & \text{s.t.} \quad \sum_{p=1}^T \mu_p = 1 \\ & \text{with} \quad \Delta(y, j) = \begin{cases} 0 & \text{if } y = j \\ 1 & \text{if } y \neq j. \end{cases} \end{aligned} \quad (2.14)$$

As is common practice, we use alternating descent minimization [Bertsekas, 1999] on w and μ . When optimizing on μ , we use a conditional gradient descent algorithm, with linesearch, described in Alg. 1. For w , we could directly optimize the problem in (2.14), which would be equivalent to solving (2.4), but we chose instead to directly minimize $\mathcal{L}(x, y, w, \mu)$ over w , which is more natural.

Because problem (2.14) is non-convex, it cannot be solved exactly by alternating descent, but instead will converge to a local minimum. Following common practice when dealing with latent variables [Cinbis et al., 2014], we prevent harmful auto-reinforcement by splitting 50%/50% our training data ($\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$) at each alternating iteration, and optimizing w and μ on a different set. The resulting algorithm is presented in Alg. 2.

Algorithm 1 Conditional Gradient Descent for ITW

INPUTS: Dataset \mathcal{Z} . Set \mathbb{T} of transformations. Current estimate of w . Initial μ_1 .

$$P(\mu) := \sum_{(x,y) \in \mathcal{Z}} \mathcal{L}(x, y, w, \mu) - \max_j \{\mathcal{L}(x, j, w, \mu) + \Delta(y, j)\}.$$

For $i = 1$ to N **do**

- Compute $s = \arg \min s^\top \nabla_\mu P$ s.t. $\sum_{p=1}^T s_p = 1$.
- Define $\gamma = 2/(2 + k)$
- **While** $P(\mu_i + \gamma(s - \mu_i)) > P(\mu_i)$ **do** $\gamma \leftarrow \gamma/2$ **done**
- Update $\mu_{i+1} \leftarrow \mu_i + \gamma(s - \mu_i)$.

done

OUTPUT: μ_N

Algorithm 2 Image Transformation Weighting

INPUTS: Dataset \mathcal{Z} . Set \mathbb{T} of transformations.

Initialize w_1 to 0 and μ_1 to $(1, 0, 0, \dots, 0)$.

For $i = 1$ to N **do**

- Split \mathcal{Z} 50/50 into $(\mathcal{Z}_1, \mathcal{Z}_2)$.
- $w_i \leftarrow \operatorname{argmin} \sum_{(x,y) \in \mathcal{Z}_1} \mathcal{L}(x, y, w, \mu_i)$
- Optimize μ_i with Alg. 1 over \mathcal{Z}_2 .

Set $\mu = \mu_N$ and $w = \operatorname{argmin} \sum_{(x,y) \in \mathcal{Z}} \mathcal{L}(x, y, w, \mu)$

OUTPUT: (w, μ)

2.3.5 Image Transformation Pursuit (ITP)

The ITW algorithm is appropriate for low-dimensional features but requires loading in memory all transformed versions of all examples. It is therefore not scalable to

very large decriptors or datasets. We propose an alternative, which is called Image Transformation Pursuit (ITP), able to only select transformations one at a time, thus dramatically reducing the selection process. We now detail this algorithm.

Our proposed transformations selection algorithm is based on a greedy search in \mathbb{T} . We start from an initial set $\mu^{(0)}$ of transformations, which can either be empty, only contain the original images ($\mu_k^{(0)} = 1$ if $k = 1$ else 0), or contain some transformations determined a priori to be beneficial. As before, to prevent auto-reinforcement, the training dataset \mathcal{Z} is split into a train \mathcal{Z}_1 and validation \mathcal{Z}_2 set. We then maintain a set of current transformations and monitor the gain on the risk obtained by adding a new transformation p (setting $\mu_p = 1$). The optimal weight is thus:

$$w_p = \operatorname{argmin}_w \sum_{x,y \in \mathcal{Z}_1} \mathcal{L}(x, y, w, \mu \cup \{p\}) + \Omega(w). \quad (2.15)$$

The transformation p is scored using the loss on the validation set:

$$\sum_{x,y \in \mathcal{Z}_2} \mathcal{L}(x, y, w_p, \mu \cup \{p\}). \quad (2.16)$$

As an alternative, and because we no longer require a differentiable loss, we propose to directly evaluate the classification accuracy:

$$A_p = \sum_{x,y \in \mathcal{Z}_2} \mathbb{1} \left[y = \arg \min_k \mathcal{L}(x, k, w_p, \mu \cup \{p\}) \right]. \quad (2.17)$$

The algorithm stops when S transformations have been selected. We detail the process in Algorithm 3.

2.3.5.1 Ranking Alternative.

We also propose a simple one-step alternative to ITP which we refer to as Transformation Ranking or TR. As is the case for ITP, we start with the original samples and quantify the gain that we would get by adding the S possible transformations. We rank the transformations based on this gain and select the top S transformations. Note that in this setting, the redundancy between the transformations is not taken into account (only the gain with respect to the original images).

2.3.5.2 Implementation Details.

We consider linear SVM classifiers, which are trained with Stochastic Gradient Descent (SGD) [Bottou, 2012, Akata et al., 2013]. At each outer iteration of ITP, we estimate the additional gain offered by a specific transformation at a low cost using

Algorithm 3 Image Transformation Pursuit (ITP)

INPUTS: Dataset \mathcal{Z} . Set of T transformations. Number S of transformations to select.

Initialize $\mu^{(0)} = (1, 0, 0, \dots, 0)$.

For $i = 1$ to S **do**

- split \mathcal{Z} into \mathcal{Z}_1 and \mathcal{Z}_2 .
- **For** $p = 1$ to T **do**
 - Set $\nu_k = 1/(i + 1)$ if $\mu_k^{(i-1)} \neq 0$ or $k = p$ else 0.
 - Train:

$$w_p \leftarrow \operatorname{argmin}_w \sum_{x,y \in \mathcal{Z}_1} \mathcal{L}(x, y, w_p, \nu) + \Omega(w).$$
 - Validate:

$$A_p \leftarrow \sum_{x,y \in \mathcal{Z}_2} \mathbb{1}[y = \operatorname{argmin}_k \mathcal{L}(x, k, w_p, \nu)]$$
- Keep best transformation: $p^* \leftarrow \operatorname{argmax}_p A_p$.
- Set $\mu_k^{(i)} = 1/(i + 1)$ if $\mu_k^{(i-1)} \neq 0$ or $k = p^*$ else 0.

OUTPUT: $\mu^{(S)}$

the following strategy. Indeed, at iteration k (i.e. k transformations are already selected), a classifier w_k is learned using SGD on half of the training set augmented with these k transformations. For each $(k+1)$ -th candidate transformation, SGD is run using w_k as a warm start and few epochs. We determine the initial learning rate η_0 following the heuristic given in Bottou [2012]. We also cross-validate the regularization parameter λ using values around the best parameter selected at the previous main iteration. In our experiments, however, this optimal value never changed across the main iterations.

2.4 Experiments

We first describe the datasets and the experimental setup. We then study the impact of different design choices for using transformed descriptors to improve performance. We also quantitatively compare the ITW, ITP and TR algorithms and several baselines. Finally, we compare our method against the state of the art on fine-grained image classification datasets.

2.4.1 Datasets

We report results on three challenging fine-grained classification benchmarks which aim to classify respectively birds species, aircrafts and car models. For all those

datasets, the evaluation metric is the top-1 classification accuracy.

2.4.1.1 CUB

The Caltech-UCSB-Birds-200-2011 dataset [Wah et al., 2011] is a fine-grained classification benchmark consisting of 200 bird species and approximately 12,000 images. We use the provided training/test split: there are approximately 30 images per class at both training and test time.

2.4.1.2 Aircrafts

The Fine-Grained Visual Classification of Aircraft (FGVC-Aircraft) [Maji et al., 2013] dataset consists of 10,000 images of 100 classes of flying vehicles. The dataset is split uniformly in a train, validation and test set. Following common practice, we use the two first splits for training and report results on the third split. This dataset was part of the FGComp’13 challenge.

2.4.1.3 Cars

The third fine-grained classification dataset we use is the one of Krause et al. [2013], whose goal is to differentiate between 196 car models observed in 16,185 images. This dataset was also part of the FGComp’13 challenge. We use the provided train/test split.

2.4.2 Image descriptors

To show that our method is able to improve on a different types of descriptors, we investigate three of them.

2.4.2.1 Fisher Vectors

Fisher Vector feature representations [Sánchez et al., 2013] are attractive for fine-grained classification [Gosselin et al., 2014]. Following common practice, we use a combination of SIFT [Lowe, 2004] and color [Clinchant et al., 2007] local descriptors, extracted on a dense grid at multiple scales. We append the patch location (x, y) and scale σ to the 61-dim PCA-projected patch descriptors, thus resulting in 64-dim descriptors, following Sánchez et al. [2012] Fisher vectors are separately evaluated on both of these features, and we also investigate the result of late-fusion (summing class scores after prediction) of the two resulting global descriptors. We denote these approaches resp. as “SIFT-Fisher”, “Color-Fisher” and “Fisher-fusion”. We use a Gaussian mixture model (GMM) with 256 Gaussians, resulting in 32K-dim representations.

2.4.2.2 Convolutional Neural Networks

We investigate the benefits of our method with off-the-shelf CNN features. We use the output of the penultimate layer (the one before the last fully-connected layer) of the very deep network of Simonyan and Zisserman [2014], and the parameters that are available online². We use the 16-layer model “D” network as it was shown to give similar performances to “E” while requiring slightly less memory. This is also the protocol of Lin et al. [2015]. The final descriptors have a relatively low dimension of 4096.

2.4.2.3 Bilinear CNN features

To get state-of-the-art results, we use the powerful bilinear features of Lin et al. [2015], specifically the $[D, D]$ setting, which has shown outstanding performances on the three datasets on which we report results. These features are obtained as sum-pooled outer-products of the last convolutional features of the previous “D” network. As in Lin et al. [2015], we use power and L2 normalization. The resulting features have the huge dimension of $512 \times 512 = 262K$.

2.4.3 Training procedure

We use the J-SGD stochastic gradient descent package³. Following Akata et al. [2013], we use One-Versus-Rest SVM as a classifier, for which three parameters have to be tuned: the learning rate, the regularization parameter λ and the number of epochs. In a first offline stage, we determine these parameters independently for all datasets and features. Five-fold cross-validation is used for λ and the number of epochs. The learning rate is automatically tuned using the trick of Bottou [2012]. The same parameters are kept throughout the remaining experiments. In particular, we verified that these parameters stay optimal when adding transformed examples.

2.4.4 ITW

We start by reporting results with the full ITW algorithm, which allows to weight all transformations in a holistic approach. Note that because it requires storing all descriptors with all possible transformations, we restrict ourselves to the 4096 dimensional CNN features. The results are displayed in Table 2.1, and show that ITW gives performance that is on par with ITP. Because of the simplicity of the latter, we choose to use this approximation in the remainder of this chapter.

²http://www.robots.ox.ac.uk/~vgg/research/very_deep/

³lear.inrialpes.fr/src/jsgd

	CUB	Aircrafts	Cars
\emptyset	61.3	40.9	36.7
ITP	69.7	51.7	53.5
ITW	69.5	53.7	56.3

Table 2.1: Comparison between ITP and ITW for CNN descriptors.

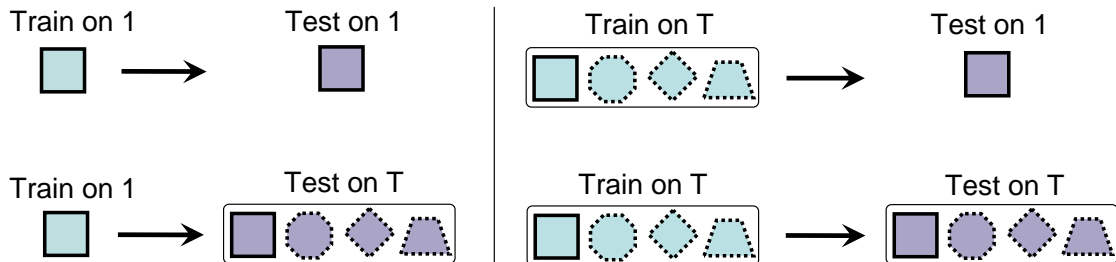


Figure 2.3: Illustration of the different training and test strategies. Both training and test can be done either on the original single image or on the set of all transformed images.

2.4.5 Impact of score aggregation

We evaluate different dataset augmentation strategies. We assume that, after the selection pass, we have $T - 1$ transformations per image, in addition to the original one, so that each image has T representatives. We investigate the following scenarios (Fig 2.3).

- Train 1/ Test 1: training and test are done only on the original images (baseline).
- Train T / Test 1: training is performed using the original as well as the transformed images, and test is done on original test images.
- Train 1/ Test T : test is done on the transformed images, and their scores are averaged, while training is done on the original images.
- Train T / Test T : training and test are done on the transformed images.

Table 2.2 shows that the “Train T / Test T ” scheme outperforms by far the other ones. This demonstrates the *importance of applying transformations at both training and test time*. In what follows, we use this scheme for all experiments unless stated otherwise.

#trans		CUB				Aircrafts				Cars			
#train	#test	SIFT	color	fusion	CNN	SIFT	color	fusion	CNN	SIFT	color	fusion	CNN
1	1	15.4	22.3	27.3	61.3	60.6	39.1	62.5	40.9	52.2	31.8	60.9	36.7
1	T	15.4	22.3	25.4	61.3	60.6	39.1	57.4	40.9	59.9	31.8	56.2	38.3
T	1	20.7	27.6	32.8	63.3	65.7	41.9	62.5	41.6	62.4	37.6	61.4	39.6
T	T	27.5	36.1	41.3	69.7	69.6	49.0	66.8	51.7	70.6	44.9	69.1	53.5

Table 2.2: Comparison of different training and test scenarios for $T = 6$, i.e. $T - 1$ corresponds to the number of transformations used for training and test. Score aggregation is performed by averaging. Transformations selected with ITP.

Scheme scheme	CUB				Aircrafts				Cars			
	SIFT	color	fusion	CNN	SIFT	color	fusion	CNN	SIFT	color	fusion	CNN
avg	27.5	36.1	41.3	69.7	69.6	49.0	66.8	51.7	70.6	44.9	69.1	53.5
max	23.2	33.0	38.6	69.2	69.0	48.2	66.0	47.9	69.7	43.7	68.5	46.5
soft-max	27.5	36.1	41.3	69.8	69.5	49.2	66.6	51.7	70.8	45.0	69.2	53.5

Table 2.3: Comparison of score aggregation schemes.

Scheme scheme	CUB				Aircrafts				Cars			
	SIFT	color	fusion	CNN	SIFT	color	fusion	CNN	SIFT	color	fusion	CNN
Data Aug	27.5	36.1	41.3	69.7	69.6	49.0	66.8	51.7	70.6	44.9	69.1	53.5
MK.	18.4	27.7	32.8	67.0	66.5	43.8	66.6	48.5	66.3	40.7	69.3	47.1
Concat.	19.9	30.0	30.8	68.5	65.5	45.1	63.2	57.7	62.3	39.0	63.0	52.1

Table 2.4: Comparison of different schemes for using virtual examples at training time. All transformations were selected by ITP.

2.4.5.1 Score aggregation scheme

We compare the different score aggregation schemes at test time (Section 2.3.3) and provide results in Table 2.3. We observe that all schemes yield similar results, with “max” slightly below. We therefore decide to use the simple average in the rest of the experiments.

2.4.5.2 Usage of transformed examples

As detailed in Section 2.3.2, there are several possible ways to use virtual examples during training time. Match-kernel (feature averaging), data augmentation and feature concatenation. Table 2.4 clearly shows that of all schemes, data augmentation performs better.

2.4.6 Transformation selection experiments

We now evaluate ITP, algorithm 3 of Section 2.3. At each iteration, the algorithm evaluates the gain in accuracy obtained for each candidate transformation indepen-

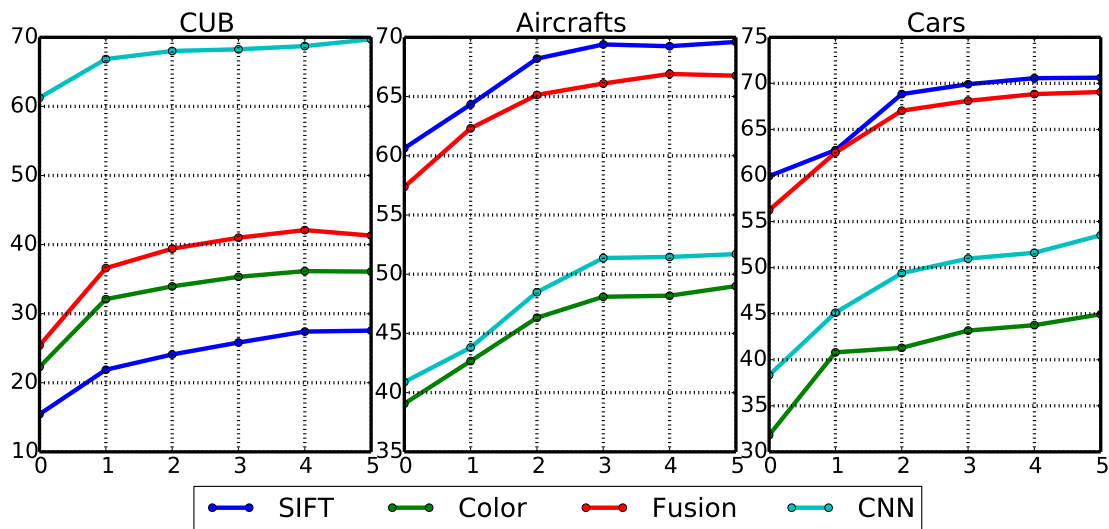


Figure 2.4: Evolution of the test accuracy with respect to the number of selected transformations.

dently on a held-out validation set. We plot the evolution of the validation score across iterations for the selected transformations, along with the accuracy computed on the full test set, in Fig. 2.4. The validation accuracy, although computed on a smaller set and without waiting for the SGD algorithm to fully converge, is varying consistently with the full test accuracy in our experiments. This validates empirically the core idea of our algorithm.

Next, we compare ITP to several other selection strategies. We first consider the choice of just adding the ‘flip’ transformation, a rather common practice [Krizhevsky et al., 2012, Gavves et al., 2013, Lin et al., 2015]. The second choice to which we compare is a random selection of transformations. This is to ensure that we do not get a similar increase in accuracy with any subset of transformations. Finally, we compare with the TR variant of our algorithm, which is a cheaper version of ITP. Results are presented in Fig. 2.5 and 2.6. As can be observed, the selection obtained by ITP significantly outperforms the first two baselines (flip and random). ITP itself performs at least on par with the TR variant, and sometimes significantly better. This is due to the fact that the selected transformations are individually beneficial, but heavily redundant (different crops for instance).

Fig. 2.7 (left) shows the first five transformations selected by ITP. We can see that crops are the most frequently selected transformations. We stack in Fig. 2.7 (right) the selected crops on the CUB dataset, thus leading to a saliency

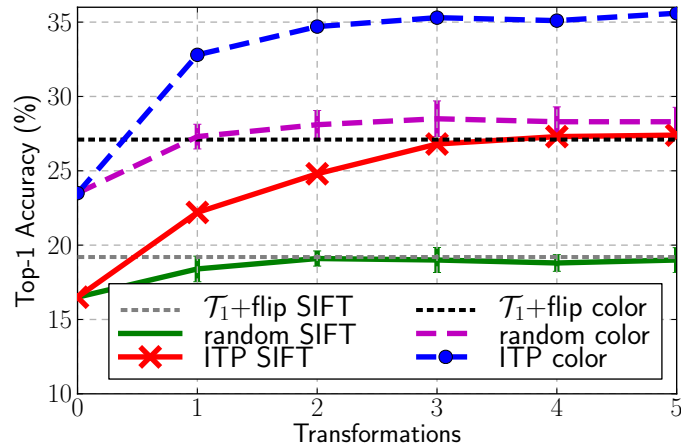


Figure 2.5: Accuracy on CUB after adding up to five transformations with ITP, with random selection (5 trials averaged, error bars shown), or with just flip in addition to the original images (\mathcal{T}_1).

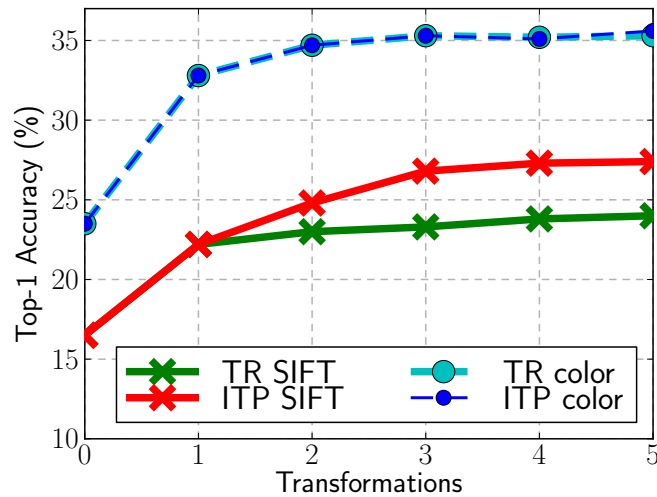


Figure 2.6: Evolution of the test accuracy on CUB as a function of the number of transformations selected by ITP, or its cheaper TR variant.

map. They clearly focus on the center of the image. One might argue from this observation that, by applying crops at training and test time, we are just learning a prior on the object location (*ie* a saliency map). To test this hypothesis, we perform the following experiment: for each training and test image, we extract local descriptors from all its transformed versions and aggregate them in a single FV. The patches which are present in multiple croppings are, therefore, weighted

	CUB			Aircrafts			Cars		
	SIFT	Color	CNN	SIFT	Color	CNN	SIFT	Color	CNN
1	crop5	crop1	crop1	flip	color0	flip	crop5	crop5	crop9
2	flip	crop5	crop8	homo3	crop3	crop1	flip	crop0	crop2
3	crop6	flip	homo7	homo7	crop5	crop2	crop2	flip	crop7
4	crop1	crop6	flip	homo2	crop8	crop8	crop3	crop2	homo7
5	rot-9	crop2	crop5	crop3	color1	crop5	crop0	color2	crop5

Figure 2.7: First five transformations, as selected by ITP.



Figure 2.8: Overlaying the different crops selected by ITP.

more than patches which occur in few or no crops. This is equivalent to weighting patches with the saliency map Sánchez et al. [2012]. Note that this approach is different from the match-kernel approach “MK” previously described, because of Fisher Vector normalizations. On CUB for Fisher-Fusion features, we get 35.9% which is a significant improvement over the 27.3% baseline – see Table 2.2. Yet, this is only half of the improvement that we get with ITP (41.3%). This result confirms that our approach is not merely learning a prior on the object location, but also invariances against extrinsic variability.

2.4.7 Total training time vs. target accuracy

We plot the test accuracy as a function of the number of iterations performed by the SGD in Fig. 2.9 (one SGD iteration corresponds to processing a single training image). Interestingly, we observe that the learning process is faster for higher numbers of transformations T . To better exhibit this phenomenon, we show in Table 2.5 the number of training samples that the SGD algorithm has to process

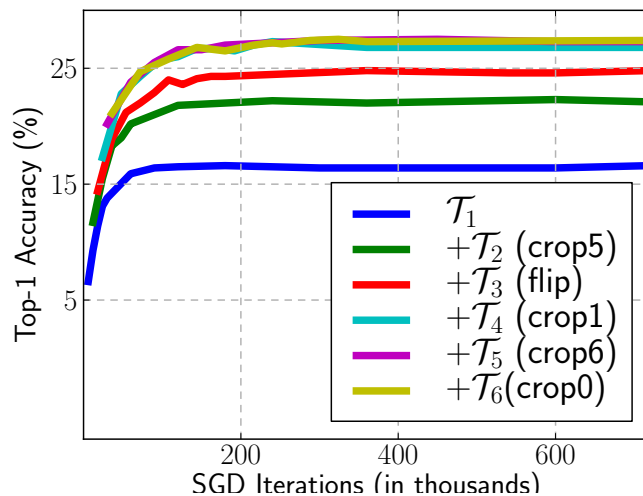


Figure 2.9: Test accuracy as a function of the number of SGD iterations on CUB (left) with SIFT-Fisher.

T	CUB acc target			
	25%	30%	35%	40%
1	42	X	X	X
2	29	48	121	X
3	31	41	79	250
5	33	44	66	141

Table 2.5: Number of training samples (in thousands) that the SGD has to process to reach a given accuracy on the test set, for a varying number of transformations T for Fisher-Fusion features. The 'X' sign means that the target accuracy cannot be reached at all.

to reach a certain accuracy on the test set. It can be observed that it takes *less time* to reach a certain accuracy when T is larger (i.e. in spite of a larger training set). For instance, on CUB with Fisher-Fusion descriptors, reaching a 35% accuracy is achieved for $T = 5$ transformations after examining about 2 times less training samples than for $T = 2$. This result is consistent with the work of Shalev-Shwartz and Srebro [2008].

2.4.8 Comparison between virtual and real examples

Because of costly manual labelling, it is far more interesting to use virtual examples than real ones. Yet, because of their statistical dependency to their original image, they bring less performance. In Fig. 2.10, we show how many true examples are required with added transformations to obtain the same performance as without transformations. For instance, with five added transformations ($T = 6$), one can get a higher accuracy with 5 original examples and their transformations than with 20 lone labeled examples.

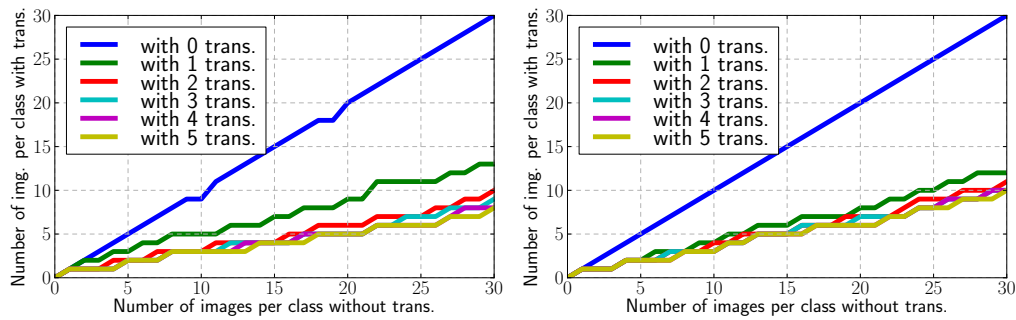


Figure 2.10: Equivalence in terms of accuracy of images with virtual examples and lone examples on CUB (30 images per class is the whole dataset) with ITP. Left: SIFT-Fisher. Right: Color-Fisher.

Alternatively, we show in Fig. 2.11 the number of labeled examples one needs to reach a target accuracy, for varying number of transformations.

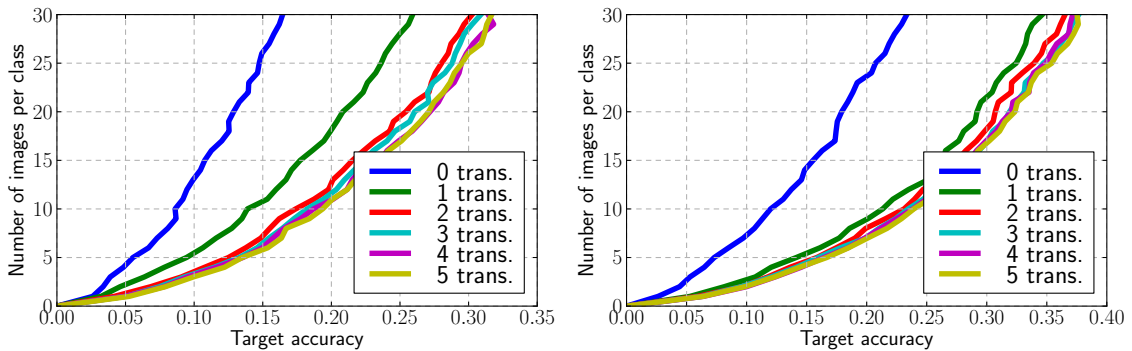


Figure 2.11: Evolution of the accuracy with an increased number of images per class on the CUB dataset with transformations selected by ITP. Left: SIFT-Fisher. Right: Color-Fisher.

Dataset	Birds	Aircrafts	Cars
Raw BCNN + flip [Lin et al., 2015]	80.1	76.8	82.9
Our Raw BCNN	80.3	75.4	83.0
Our Raw BCNN + flip	81.4	77.8	84.6
Our Raw BCNN ITP	82.6	79.1	86.7
Fine-tuned BCNN + flip [Lin et al., 2015]	84.0	84.8	90.6
State-of-the-art (finetuning)	84.1	85.1	92.6

Table 2.6: Performance of our method compared to the state of the art. Our method holds state-of-the-art performance for off-the-shelf features.

2.4.9 Comparison to state of the art

We now compare our results against the state of the art. In particular, we use the bilinear features of Lin et al. [2015] and are able to improve on them. We show the results of adding up to 5 transformations with ITP on the performance of the bilinear features in Table 2.6. One important conclusion of this experiment is that careful selection of the classification parameters is crucial to get good performance. The approach “Our Raw BCNN + flip” uses the same data as Lin et al. [2015] but is consistently above because of it. Our proposed method, named “Our Raw BCNN ITP” improves significantly over using only the flip transformation. Our method holds state-of-the-art performance for non-finetuned features. We only use off-the-shelf features without fine-tuning, as the process of applying ITP on fine-tuned features is difficult and left as future work. We also argue that performing fine-tuning on a new dataset requires considerably more time and resources (several days on a GPU) than simple feature extraction. The last row of Table 2.6 is taken from Lin et al. [2015] and is the current state-of-the-art on the datasets we use. All these methods use fine-tuning.

2.4.10 Locality of transformed descriptors

The positioning of virtual examples with respect to their original one is unclear. In particular, it is difficult to know if they provide a good candidate for a new iid sample that would have been drawn from the class distribution. To investigate these matters, we propose to compute several statistics. In figure 2.12, we plot the average distance between a transformed example and its original, which we compare to the mean inter-example distance, as well as the intra-class average distance. This provides a good evaluation of the robustness of our features to certain transformations. Also, one can see that virtual examples are highly correlated to their originals, as their distance to it is usually much lower than to an other

example of the same class.

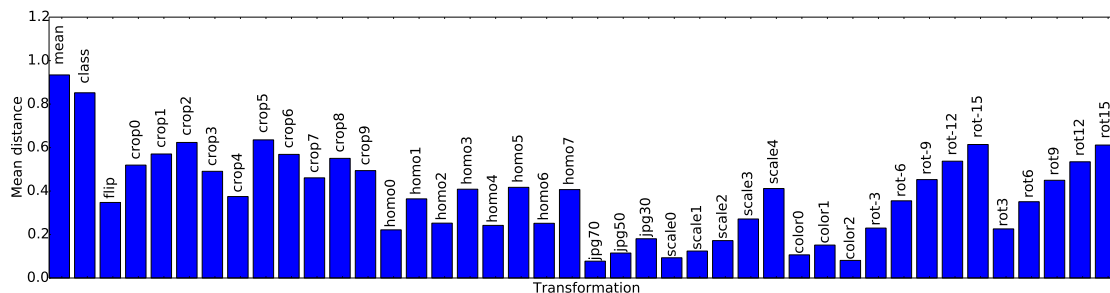


Figure 2.12: Histogram of the average distance of a transformed example to its original. Note that since all features are normalized, the upper bound is effectively 1. The “mean” bar is the average distance between all original examples, and “class” the average distance between examples of the same class. Results are for CNN features on the CUB dataset. As we can see, these features are more robust to color changes and flip than to crops or rotations.

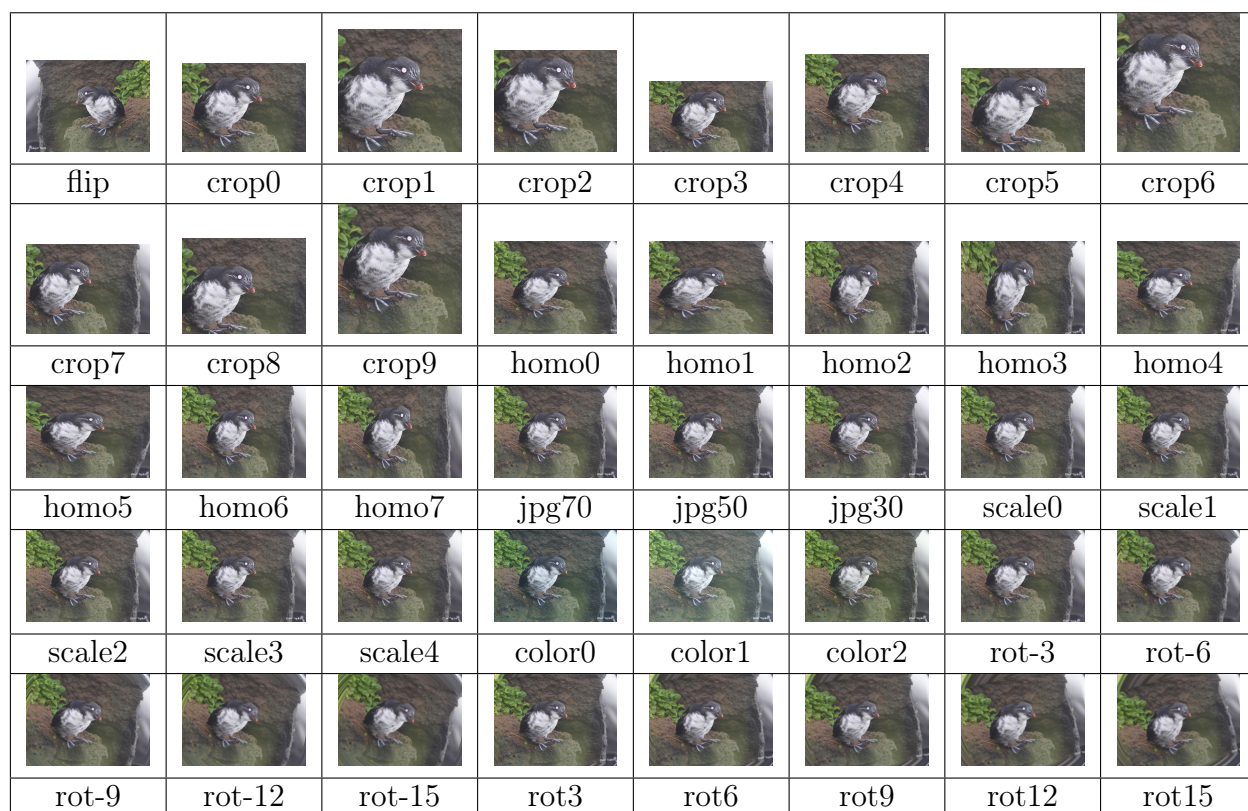


Figure 2.13: All transformations used in this article, applied to a CUB image.

Chapter 3

Patch Description with Convolutional Kernel Networks

3.1	Introduction	45
3.2	Related Work	46
3.3	Convolutional Descriptors	49
3.4	Convolutional Kernel Descriptors	52
3.5	Image and Patch Retrieval Datasets	61
3.6	Experiments	63

Solche geringe Wahrnehmungen sind also von mehr Wirksamkeit, als man denken mag. Sie sind es, welche dies wunderbare Etwas, [...] die in ihrem Zusammensein klar, jedoch ihren einzelnen Teilen nach verworren sind.

Leibniz: Neue Abhandlungen über den menschlichen Verstand

3.1 Introduction

THIS chapter explores convolutional architectures as robust visual descriptors for image patches and evaluates them in the context of patch and image retrieval. We explore several levels of supervision for training such networks, ranging from fully supervised to unsupervised. In this context, requiring supervision may seem unusual since data for retrieval tasks typically does not come with labels. Convolutional Neural Networks (CNNs) have achieved state-of-the-art in many other computer vision tasks, but require abundant labels to learn their parameters. For this reason, previous work with CNN architectures on image retrieval have focused on using global [Babenko et al., 2014] or aggregated local [Razavian et al., 2014, Gong et al., 2014, Ng et al., 2015, Babenko and Lempitsky, 2015, Tolias et al., 2015] CNN descriptors that were learned on an unrelated classification task. To improve the performance of these transferred features, Babenko et al. [2014] showed that fine-tuning global descriptors on a dataset of landmarks results in improvements on retrieval datasets that contain buildings. It is unclear, however, if the lower levels of a convolutional architecture, appropriate for a local description, will be impacted by such a global fine-tuning [Yosinski et al., 2014]. Recent approaches have, thus, attempted to discriminatively learn low-level convolutional descriptors, either by enforcing a certain level of invariance through explicit transformations [Fischer et al., 2014] or by training with a patch-matching dataset [Zagoruyko and Komodakis, 2015, Simo-Serra et al., 2015]. In all cases, the link between the supervised classification objective and image retrieval is artificial, which motivates us to investigate the performance of new unsupervised learning techniques.

To do so, we propose an unsupervised patch descriptor based on Convolutional Kernel Networks (CKNs)[Mairal et al., 2014b]. This required to turn the CKN proof of concept of Mairal et al. [2014b] into a descriptor with state-of-the-art performance on large-scale benchmarks. This work introduces significant improvements of the original model, algorithm, and implementation, as well as adapting the approach to image retrieval. Our conclusion is that supervision might not be necessary to train convolutional networks for image and patch retrieval, since our unsupervised descriptor achieves the best performance on several standard benchmarks.

One originality of our work is also to *jointly* evaluate our models on the problems of *patch* and *image* retrieval. Most works that study patch representations [Brown et al., 2011, Winder et al., 2009, Zagoruyko and Komodakis, 2015, Fischer et al., 2014], do so in the context of patch retrieval only, and do not test whether conclusions also generalize to image retrieval (typically after an aggregation step). In fact, the correlation between the two evaluation methods (patch- and image-level) is not clear beforehand, which motivated us to design a new dataset to answer this

question. We call this dataset RomePatches; it consists of views of several locations in Rome [Li et al., 2010], for which a sparse groundtruth of patch matches is obtained through 3D reconstruction. This results in a dataset for patch and image retrieval, which enables us to quantify the performance of patch descriptors for both tasks.

To evaluate the descriptor performance, we adopt the pipeline of Fig. 1.10, described in detail in section 3.6.1.1. We use the popular Hessian-Affine detector of Mikolajczyk and Schmid [2004], which has been shown to give state-of-the-art results [Tuytelaars and Mikolajczyk, 2008]. The regions around these points are encoded with the convolutional descriptors proposed in this work. We aggregate local features with VLAD-pooling [Jégou et al., 2010] on the patch descriptors to build an approximate matching technique. VLAD pooling has been shown to be better than Bag-of-Word, and to give similar performance to Fisher Vectors [Peronnin and Dance, 2007], another popular technique for image retrieval [Jégou et al., 2012].

The remainder of this chapter is organized as follows. We discuss previous work that is most relevant to our approach in Section 3.2. We describe the framework for convolutional descriptors and convolutional kernel networks in Sections 3.3 and 3.4. We introduce the RomePatches dataset as well as standard benchmarks for patch and image retrieval in Section 3.5. Section 3.6 describes experimental results.

3.2 Related Work

In this section we first review the state of the art for patch description and then present deep learning approaches for image and patch retrieval. For deep patch descriptors, we first present supervised and, then, unsupervised approaches.

3.2.1 Patch descriptors

A patch is a image region extracted from an image. Patches can either be extracted densely or at interest points. The most popular patch descriptor is SIFT [Lowe, 2004], which showed state-of-the-art performance [Mikolajczyk and Schmid, 2005] for patch matching. It can be viewed as a three-layer CNN, the first layer computing gradient histograms using convolutions, the second, fully-connected, weighting the gradients with a Gaussian, and the third pooling across a 4x4 grid. Local descriptors that improve SIFT include SURF [Bay et al., 2006], BRIEF [Calonder et al., 2010] and LIOP [Wang et al., 2011]. Recently, Dong and Soatto [2015] build on SIFT using local pooling on scale and location to get state-of-the-art performance in patch retrieval.

All these descriptors are hand-crafted and their relatively small numbers of parameters have been optimized by grid-search. When the number of parameters to be set is large, such an approach is unfeasible and the optimal parametrization needs to be learned from data.

A number of approaches learn patch descriptors without relying on deep learning. Most of them use a strong supervision. Brown et al. [2011] (see also Winder et al. [2009]) design a matching dataset based on 3D models of landmarks and use it to train a descriptor consisting of several existing parts, including SIFT, GLOH [Mikolajczyk and Schmid, 2005] and Daisy [Tola et al., 2010]. Philbin et al. [2010] learn a Mahalanobis metric for SIFT descriptors to compensate for the quantization error, with excellent results in instance-level retrieval. Simonyan et al. [2014] propose the “Pooling Regions” descriptor and learn its parameters, as well as a linear projection using stochastic optimization. Their learning objective can be cast as a convex optimization problem, which is not the case for classical convolutional networks.

An exception that departs from this strongly supervised setting is [Bo et al., 2010] which presents a match-kernel interpretation of SIFT, and a family of kernel descriptors whose parameters are learned in an unsupervised fashion. The Patch-CKN we introduce generalizes kernel descriptors; the proposed procedure for computing an explicit feature embedding is faster and simpler.

3.2.2 Deep learning for image retrieval

If a CNN is trained on a sufficiently large labeled set such as ImageNet [Deng et al., 2009], its intermediate layers can be used as image descriptors for a wide variety of tasks including image retrieval [Babenko et al., 2014, Razavian et al., 2014]. The output of one of the fully-connected layers is often chosen because it is compact, usually 4,096-dim. However, global CNN descriptors lack geometric invariance [Gong et al., 2014], and produce results below the state of the art for instance-level image retrieval.

In [Razavian et al., 2014, Gong et al., 2014], CNN responses at different scales and positions are extracted. We proceed similarly, yet we replace the (coarse) dense grid with a patch detector. There are important differences between their work and ours. While they use the penultimate layer as patch descriptor, we show in our experiments that we can get improved results with preceding layers, that are cheaper to compute and require smaller input patches. Closely related is the work of Ng et al. [2015] which uses VLAD pooling on top of very deep CNN feature maps, at multiple scales with good performance on Holidays and Oxford. Their approach is similar to the one of Gong et al. [2014], but faster as it factorizes computation using whole-image convolutions. Building on this, Tolias et al. [2015] uses an

improved aggregation method compared to VLAD, that leverages the structure of convolutional feature maps.

Babenko et al. [2014] use a single global CNN descriptor for instance-level image retrieval and fine-tune the descriptor on an external landmark dataset. We experiment with their fine-tuned network and show improvement also with lower levels on the Oxford dataset. CKN descriptors still outperform this approach. Finally, [Jiang et al., 2014] proposes a Siamese architecture to train image retrieval descriptors but do not report results on standard retrieval benchmarks.

3.2.3 Deep patch descriptors

Recently [Long et al., 2014, Fischer et al., 2014, Simo-Serra et al., 2015, Zagoruyko and Komodakis, 2015] outperform SIFT for patch matching or patch classification. These approaches use different levels of supervision to train a CNN. Long et al. [2014] learn their patch CNNs using category labels of ImageNet. Fischer et al. [2014] create surrogate classes where each class corresponds to a patch and distorted versions of this patch. Matching and non-matching pairs are used in [Simo-Serra et al., 2015, Zagoruyko and Komodakis, 2015]. There are two key differences between those works and ours. First, they focus on patch-level metrics, instead of actual image retrieval. Second, and more importantly, while all these approaches require some kind of supervision, we show that our Patch-CKN yields competitive performance in both patch matching and image retrieval without supervision.

3.2.4 Unsupervised learning for deep representations

To avoid costly annotation, many works leverage unsupervised information to learn deep representations. Unsupervised learning can be used to initialize network weights, as in Erhan et al. [2009, 2010]. Methods that directly use unsupervised weights include domain transfer [Donahue et al., 2014] and k-means [Coates and Ng, 2012]. Most recently, some works have looked into using temporal coherence as supervision [Goroshin et al., 2015, 2014]. Closely related to our work, Agrawal et al. [2015] propose to train a network by learning the affine transformation between synchronized image pairs for which camera parameters are available. Similarly, Jayaraman and Grauman [2015] uses a training objective that enforces for sequences of images that derive from the same ego-motion to behave similarly in the feature space. While these two works focus on a weakly supervised setting, we focus on a fully unsupervised one.

3.3 Convolutional Descriptors

In this section, we briefly review notations related to CNNs and the possible learning approaches.

3.3.1 Convolutional Neural Networks

In this work, we use convolutional features to encode patches extracted from an image. We call convolutional descriptor any feature representation f that decomposes in a *multi-layer* fashion as:

$$f(x) = \gamma_K(\sigma_K(W_K \dots \gamma_2(\sigma_2(W_2 \gamma_1(\sigma_1(W_1 x)) \dots))), \quad (3.1)$$

where x is an input patch represented as a vector, the W_k 's are matrices corresponding to linear operations, the σ_k 's are pointwise non-linear functions, e.g., sigmoids or rectified linear units, and the functions γ_k perform a downsampling operation called “feature pooling”. Each composition $\gamma_k(\sigma_k(W_k \bullet))$ is called a “*layer*” and the intermediate representations of x , between each layer, are called “*maps*”. A map can be represented as pixels organized on a spatial grid, with a multidimensional representation for each pixel. Borrowing a classical terminology from neuroscience, it is also common to call “*receptive field*” the set of pixels from the input patch x that may influence a particular pixel value from a higher-layer map. In traditional convolutional neural networks, the W_k matrices have a particular structure corresponding to spatial convolutions performed by small square filters, which will need to be learned. In the case where there is no such structure, the layer is called “fully-connected”.

The hyper-parameters of a convolutional architecture lie in the choice of nonlinearities σ_k , type of pooling γ_k , in the structure of the matrices W_k (notably the size and number of filters) as well as in the number of layers.

The only parameters that are learned in an automated fashion are usually the filters, corresponding to the entries of the matrices W_k . In this chapter, we investigate the following ways of learning: (i) encoding local descriptors with a CNN that has been trained for an unrelated classification task (Sec. 3.3.2.1), (ii) using a CNN that has been trained for a classification problem that can be directly linked to the target task (e.g. buildings, see Sec. 3.3.2.1), (iii) devising a surrogate classification problem to enforce invariance (Sec. 3.3.2.2), (iv) directly learning the weights using patch-level groundtruth (Sec. 3.3.3) or (v) using unsupervised learning, such as convolutional kernel networks, which we present in Section 3.4.

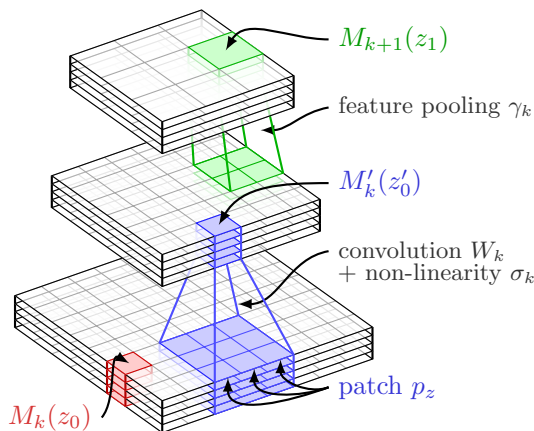


Figure 3.1: Typical organization of two successive layers of a CNN. The spatial map M'_k is obtained from M_k by convolution and pointwise non-linearity, and the top layer M_{k+1} is obtained from M'_k by a downsampling operation called feature pooling. Usual CNNs use ReLU non-linearities and max-pooling [Krizhevsky et al., 2012], while CKNs rely on exponentials and Gaussian pooling [Mairal et al., 2014b].

3.3.2 Learning Supervised Convolutional Descriptors

The traditional way of learning the weights $\mathcal{W} = (W_1, W_2, \dots, W_K)$ in (3.1) consists in using a training set $\mathcal{X} = (x_1, x_2, \dots, x_n)$ of examples, equipped with labels $\mathcal{Y} = (y_1, y_2, \dots, y_n)$, choose a loss function $\ell(\mathcal{X}, \mathcal{Y}, \mathcal{W})$ and minimize it over \mathcal{W} using stochastic gradient optimization and back-propagation [LeCun et al., 1989, Bottou, 2012]. The choice of examples, labelings and loss functional leads to different weights.

3.3.2.1 Learning with category labels

A now classical CNN architecture is AlexNet [Krizhevsky et al., 2012]. AlexNet consists of 8 layers: the first five are convolutional layers and the last ones are fully connected.

In this case, the training examples are images that have been hand-labeled into C classes such as “bird” or “cow” and the loss function is the softmax loss:

$$\ell(\mathcal{X}, \mathcal{Y}, \mathcal{W}) = \sum_{i=1}^n \log \sum_{j=1}^C \exp \left(M_K^{(i)}[j] - M_K^{(i)}[y_i] \right). \quad (3.2)$$

In Eq. (3.2) and throughout the chapter, $M_k^{(i)}$ is the output of the k -th layer ($k \in \{1, \dots, K\}$) of the network applied to example x_i . The $[j]$ notation corresponds to the j -th element of the map.

Even though the network is designed to process images of size 224×224 , each neuron of a map has a “receptive field”, see the “coverage” column in Table 3.2 from Section 3.6. Using an image of the size of the receptive field produces a 1×1 map that we can use as a low dimensional patch descriptor. To ensure a fair comparison between all approaches, we rescale the fixed-size input patches so that they fit the required input of each network.

We explore two different sets of labellings for AlexNet: the first one, which we call AlexNet-ImageNet, is learned on the training set of ILSVRC 2012 ($C = 1000$), as in the original paper [Krizhevsky et al., 2012]. This set of weights is popular in off-the-shelf convolutional features, even though the initial task is unrelated to the target image retrieval application. Following Babenko et al. [2014], we also fine-tune the same network on the Landmarks dataset, to introduce semantic information into the network that is more related to the target task. The resulting network is called AlexNet-Landmarks.

3.3.2.2 Learning from surrogate labels

Most CNNs such as AlexNet augment the dataset with jittered versions of training data to learn the filters W_k in (3.1). Dosovitskiy et al. [2014], Fischer et al. [2014] use virtual patches, obtained as transformations of randomly extracted ones to design a classification problem related to patch retrieval. For a set of patches \mathcal{P} , and a set a transformations \mathcal{T} , the dataset consists of all $\tau(p)$, $(\tau, p) \in \mathcal{T} \times \mathcal{P}$. Transformed versions of the same patch share the same label, thus defining surrogate classes. Similarly to the previous setup, the network uses softmax loss (3.2).

In this chapter, we evaluate this strategy by using the same network, called PhilippNet, as in Fischer et al. [2014]. The network has three convolutional and one fully connected layers, takes as input 64×64 patches, and produces a 512-dimensional output.

3.3.3 Learning with patch-level groundtruth

When patch-level labels are available, obtained by manual annotation or 3D-reconstruction [Winder et al., 2009], it is possible to directly learn a similarity measure as well as a feature representation. The simplest way to do so, is to replace the virtual patches in the architecture of Dosovitskiy et al. [2014], Fischer et al. [2014] described in the previous section with labeled patches of RomeTrain. We call this version “FisherNet-Rome”.

It can also be achieved using a Siamese network [Chopra et al., 2005], i.e. a CNN which takes as input the two patches to compare, and where the objective function

enforces that the output descriptors’ similarity should reproduce the ground-truth similarity between patches.

Optimization can be conducted with either a metric-learning cost [Simo-Serra et al., 2015]:

$$\ell(\mathcal{X}, \mathcal{Y}, W_K) = \sum_{i=1}^n \sum_{j=1}^n C(i, j, \|M_K^{(i)} - M_K^{(j)}\|) \quad (3.3)$$

with

$$C(i, j, d) = \begin{cases} d & \text{if } y_i = y_j \\ \max(0, 1 - d) & \text{otherwise} \end{cases} \quad (3.4)$$

or as a binary classification problem (“match”/“not-match”) with a softmax loss as in eq. (3.2) [Zbontar and LeCun, 2014, Zagoruyko and Komodakis, 2015]. For those experiments, we use the parameters of the siamese networks of Zagoruyko and Komodakis [2015], available online¹. Following their convention, we refer to these architectures as “DeepCompare”.

3.4 Convolutional Kernel Descriptors

In this chapter, the unsupervised learning strategy for learning convolutional networks is based on the convolutional kernel networks (CKNs) of Mairal et al. [2014b]. Similar to CNNs, these networks have a multi-layer structure with convolutional pooling and nonlinear operations at every layer. Instead of learning filters by optimizing a loss function, say for classification, they are trained layerwise to approximate a particular nonlinear kernel, and therefore require no labeled data.

The presentation of CKNs is divided into three stages: (i) introduction of the abstract model based on kernels (Sections 3.4.1, 3.4.2, and 3.4.3); (ii) approximation scheme and concrete implementation (Sections 3.4.4, 3.4.5, and 3.4.7); (iii) optimization (Section 3.4.6).

3.4.1 A Single-Layer Convolutional Kernel for Images

The basic component of CKNs is a match kernel that encodes a similarity between a pair of images (M, M') of size $m \times m \times d$ pixels, which are assumed to be square. The integer d represents the number of channels, say 3 for RGB images. Note that when applied to image retrieval, these images M, M' correspond to regions – patches – extracted from an image. We omit this fact for simplicity since this presentation of CKNs is independent of the image retrieval task.

¹<https://github.com/szagoruyko/cvpr15deepcompare>

We denote by Ω the set of pixel locations, which is of size $|\Omega| = m \times m$, and choose a patch size $e \times e$. Then, we denote by P_z (resp. $P'_{z'}$) the $e \times e \times d$ patch of M (resp. M') at location $z \in \Omega$ (resp. $z' \in \Omega$). Then, the single-layer match kernel is defined as follows:

Definition 1 *Single-Layer Convolutional Kernel.*

$$K(M, M') = \sum_{z, z' \in \Omega} e^{-\frac{1}{2\beta^2}\|z-z'\|^2} \kappa(P_z, P'_{z'}), \quad (3.5)$$

with

$$\kappa(P, P') = \|P\| \|P'\| e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}, \quad (3.6)$$

where α and β are two kernel hyperparameters, $\|\cdot\|$ denotes the usual ℓ_2 norm, and \tilde{P} and \tilde{P}' are ℓ_2 -normalized versions of the patches P and P' .

K is called a *convolutional kernel*; it can be interpreted as a match-kernel that compares all pairs of patches from M and M' with a nonlinear kernel κ , weighted by a Gaussian term that decreases with their relative distance. The kernel compares indeed all locations in M with all locations in M' . It depends notably on the parameter α , which controls the nonlinearity of the Gaussian kernel comparing two normalized patches \tilde{P} and \tilde{P}' , and on β , which controls the size of the neighborhood in which a patch is matched with another one. In practice, the comparison of two patches that have very different locations z and z' will be negligible in the sum (3.5) when β is small enough. Hence, the parameter β allows us to control the local shift-invariance of the kernel.

3.4.2 From Kernels to Infinite-Dimensional Feature Maps

Designing a positive definite kernel on data is equivalent to defining a mapping of the data to a Hilbert space, called reproducing kernel Hilbert space (RKHS), where the kernel is an inner product [Cucker and Zhou, 2007]; exploiting this mapping is sometimes referred to as the “kernel trick” [Schölkopf and Smola, 2002]. In this section, we will show how the kernel (3.5) may be used to generalize the concept of “feature maps” from the traditional neural network literature to kernels and Hilbert spaces.² The kernel K is indeed positive definite (see the appendix of Mairal et al. 2014b) and thus it will suits our needs.

Basically, feature maps from convolutional neural networks are spatial maps where every location carries a finite-dimensional vector representing information

²Note that in the kernel literature, “feature map” denotes the mapping between data points and their representation in a reproducing kernel Hilbert space (RKHS). Here, feature maps refer to spatial maps representing local image characteristics at every location, as usual in the neural network literature LeCun et al. [1998].

from a local neighborhood in the input image. Generalizing this concept in an infinite-dimensional context is relatively straightforward with the following:

Definition 2 *Let \mathcal{H} be a Hilbert space. The set of feature maps is the set of applications $\varphi : \Omega \rightarrow \mathcal{H}$.*

Given an image M , it is now easy to build such a feature map. For instance, consider the nonlinear kernel for patches κ defined in Eq. (3.6). According to the Aronzsjan theorem, there exists a Hilbert space \mathcal{H}_κ and a mapping ϕ_κ such that for two image patches P, P' – which may come from different images or not –, $\kappa(P, P') = \langle \phi_\kappa(P), \phi_\kappa(P') \rangle_{\mathcal{H}_\kappa}$. As a result, we may use this mapping to define a feature map $\varphi_M : \Omega \rightarrow \mathcal{H}_\kappa$ for image M such that $\varphi_M(z) = \phi_\kappa(P_z)$, where P_z is the patch from M centered at location z . The first property of feature maps from classical CNNs would be satisfied: at every location, the map carries information from a local neighborhood from the input image M .

We will see in the next subsection how to build sequences of feature maps in a multilayer fashion, with invariant properties that are missing from the simple example we have just described.

3.4.3 From Single-Layer to Multi-Layer Kernels

We now show how to build a sequence of feature maps $\varphi_M^1, \dots, \varphi_M^k$ for an input image M initially represented as a finite-dimensional map $\varphi_M^0 : \Omega_0 \rightarrow \mathbb{R}^{p_0}$, where Ω_0 is the set of pixel locations in M and p_0 is the number of channels. The choice of initial map φ_M^0 is important since it will be the input of our algorithms; it is thus discussed in Section 3.4.7. Here, we assume that we have already made this choice, and we explain how to build a map $\varphi_M^k : \Omega_k \rightarrow \mathcal{H}_k$ from a previous map $\varphi_M^{k-1} : \Omega_{k-1} \rightarrow \mathcal{H}_{k-1}$. Specifically, our goal is to design φ_M^k such that

- (i) $\varphi_M^k(z)$ for z in Ω_k carries information from a local neighborhood from φ_M^{k-1} centered at location z ;
- (ii) the map φ_M^k is “more invariant” than φ_M^{k-1} .

These two properties can be obtained by defining a positive definite kernel K_k on patches from φ_M^{k-1} . Denoting by \mathcal{H}_k its RKHS, we may call $\varphi_M^k(z)$ the mapping to \mathcal{H}_k of a patch from φ_M^{k-1} centered at z . The construction is illustrated in Figure 3.2.

Concretely, we choose a patch shape \mathcal{P}_k , which is a set of coordinates centered at zero along with a set of pixel locations Ω_k such that for all z in Ω_k and u in \mathcal{P}_k , the location $z + u$ is in Ω_{k-1} . Then, the kernel K_k for comparing two patches

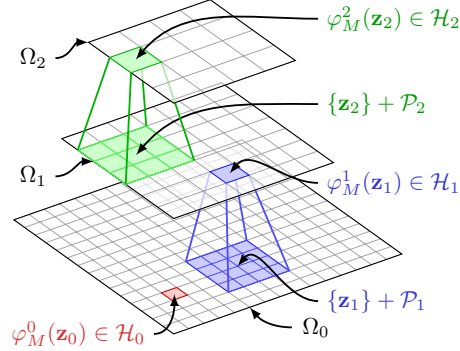


Figure 3.2: Construction of the sequence of feature maps $\varphi_M^1, \dots, \varphi_M^k$ for an input image M . The point $\varphi_M^k(z)$ for z in Ω_k represents the mapping in \mathcal{H}_k of a patch from φ_M^{k-1} centered at z . Figure adapted from Mairal et al. [2014b].

from φ_M^{k-1} and $\varphi_{M'}^{k-1}$ at respective locations z, z' in Ω_k is defined as

$$\sum_{u, u' \in \mathcal{P}_k} e^{-\frac{1}{2\beta_k^2} \|u-u'\|^2} \kappa_k(\varphi_M^{k-1}(u+z), \varphi_{M'}^{k-1}(u'+z')), \quad (3.7)$$

where

$$\kappa_k(\varphi, \varphi') = \|\varphi\|_{\mathcal{H}_{k-1}} \|\varphi'\|_{\mathcal{H}_{k-1}} e^{-\frac{1}{2\alpha_k^2} \|\tilde{\varphi} - \tilde{\varphi}'\|_{\mathcal{H}_{k-1}}^2},$$

for all φ, φ' in \mathcal{H}_{k-1} , where $\tilde{\varphi}$ (resp. $\tilde{\varphi}'$) are normalized—that is, $\tilde{\varphi} = (1/\|\varphi\|_{\mathcal{H}_{k-1}})\varphi$ if $\varphi \neq 0$ and 0 otherwise. This kernel is similar to the convolutional kernel for images already introduced in (3.5), except that it operates on infinite-dimensional feature maps. It involves two parameters α_k, β_k to control the amount of invariance of the kernel. Then, by definition, $\varphi_M^k : \Omega_k \rightarrow \mathcal{H}_k$ is the mapping such that the value (3.7) is equal to the inner product $\langle \varphi_M^k(z), \varphi_{M'}^k(z') \rangle_{\mathcal{H}_k}$.

This framework yields a sequence of infinite-dimensional image representations but requires finite-dimensional approximations to be used in practice. Among different approximate kernel embeddings techniques [Williams and Seeger, 2001, Perronnin et al., 2010, Vedaldi and Zisserman, 2012], we will introduce a data-driven approach that exploits a simple expansion of the Gaussian kernel, and which provides a new way of learning convolutional neural networks without supervision.

3.4.4 Approximation of the Gaussian Kernel

Specifically, the previous approach relies on an approximation scheme for the Gaussian kernel, which is plugged in the convolutional kernels (3.7) at every layer; this scheme requires learning some weights that will be interpreted as the parameters of a CNN in the final pipeline (see Section 3.4.5).

More precisely, for all x and x' in \mathbb{R}^q , and $\alpha > 0$, the Gaussian kernel $e^{-\frac{1}{2\alpha^2}\|x-x'\|_2^2}$ can be shown to be equal to

$$\left(\frac{2}{\pi\alpha^2}\right)^{\frac{q}{2}} \int_{v \in \mathbb{R}^q} e^{-\frac{1}{\alpha^2}\|x-v\|_2^2} e^{-\frac{1}{\alpha^2}\|x'-v\|_2^2} dv. \quad (3.8)$$

Furthermore, when the vectors x and x' are on the sphere—that is, have unit ℓ_2 -norm, we also have

$$e^{-\frac{1}{2\alpha^2}\|x-x'\|_2^2} = \mathbb{E}_{v \sim p(v)}[s(v^\top x)s(v^\top x')], \quad (3.9)$$

where s is a nonlinear function such that $s(u) \propto e^{-\frac{1}{\alpha^2} + \frac{2u}{\alpha^2}}$ and $p(v)$ is the density of the multivariate normal distribution $\mathcal{N}(0, (\alpha^2/4)\mathbf{I})$. Then, different strategies may be used to approximate the expectation by a finite weighted sum:

$$e^{-\frac{1}{2\alpha^2}\|x-x'\|_2^2} \approx \frac{1}{p} \sum_{j=1}^p \eta_j s(v_j^\top x) s(v_j^\top x'), \quad (3.10)$$

which can be further simplified, after appropriate changes of variables,

$$e^{-\frac{1}{2\alpha^2}\|x-x'\|_2^2} \approx \sum_{j=1}^p e^{w_j^\top x + b_j} e^{w_j^\top x' + b_j}, \quad (3.11)$$

for some sets of parameters w_j in \mathbb{R}^p and b_j in \mathbb{R} , $j = 1, \dots, p$, which need to be learned. The approximation leads to the kernel approximations $\langle \psi(x), \psi(x') \rangle$ where $\psi(x) = [e^{w_j^\top x + b_j}]_{j=1}^p$, which may be interpreted as the output of a one-layer neural network with p neurons and exponential nonlinear functions.

The change of variable that we have introduced yields a simpler formulation than the original formulation of Mairal et al. [2014b]. Given a set of training pairs of normalized signals $(x_1, x'_1), \dots, (x_n, x'_n)$ in \mathbb{R}^q , the weights w_j and scalars b_j may now be obtained by minimizing

$$\min_{W, b} \sum_{i=1}^n \left[e^{\frac{\|x_i - x'_i\|_2^2}{2\alpha^2}} - \sum_{j=1}^p e^{w_j^\top x_i + b_j} e^{w_j^\top x'_i + b_j} \right]^2, \quad (3.12)$$

which is a non-convex optimization problem. How we address it will be detailed in Section 3.4.6.

3.4.5 Back to Finite-Dimensional Feature Maps

Convolutional kernel networks use the previous approximation scheme of the Gaussian kernel to build finite-dimensional image representations $M_1 : \Omega_1 \rightarrow \mathbb{R}^{p_1}$, $M_2 : \Omega_2 \rightarrow \mathbb{R}^{p_2}$, \dots , $M_k : \Omega_k \rightarrow \mathbb{R}^{p_k}$ of an input image M with the following properties:

- (i) There exists a patch size $e_k \times e_k$ such that a patch $P_{l,z}$ of M_l at location z —which is formally a vector of size $e_k^2 p_{k-1}$ —provides a finite-dimensional approximation of the kernel map $\varphi_M^l(z)$. In other words, given another patch $P'_{l,z'}$ from a map M'_l , we have $\langle \varphi_M^l(z), \varphi_M^l(z) \rangle_{\mathcal{H}_l} \approx \langle P_{l,z}, P'_{l,z'} \rangle$.³
- (ii) Computing a map M_l from M_{l-1} involves convolution with learned filters and linear feature pooling with Gaussian weights.

Specifically, the learning and encoding algorithms are presented in Algorithms 4 and 5, respectively. The resulting procedure is relatively simple and admits two interpretations.

- The first one is to consider CKNs as an approximation of the infinite-dimensional feature maps $\varphi_M^1, \dots, \varphi_M^k$ presented in the previous sections [see Mairal et al., 2014b, for more details about the approximation principles].
- The second one is to see CKNs as particular types of convolutional neural networks with contrast-normalization. Unlike traditional CNNs, filters and nonlinearities are learned to approximate the Gaussian kernel on patches from layer $k-1$.

With both interpretations, this representation induces a change of paradigm in unsupervised learning with neural networks, where the network is not trained to reconstruct input signals, but where its non-linearities are derived from a kernel point of view.

3.4.6 Large-Scale Optimization

One of the challenge we faced to apply CKNs to image retrieval was the lack of scalability of the original model introduced by Mairal et al. [2014b], which was a proof of concept with no effort towards scalability. A first improvement we made was to simplify the original objective function with changes of variables, resulting in the formulation (3.12), leading to simpler and less expensive gradient computations.

The second improvement is to use stochastic optimization instead of the L-BFGS method used by Mairal et al. [2014b]. This allows us to train every layer by using a set of one million patches and conduct learning on all of their possible pairs, which is a regime where stochastic optimization is unavoidable. Unfortunately, applying stochastic gradient descent directly on (3.12) turned out to be

³Note that to be more rigorous, the maps M_l need to be slightly larger in spatial size than φ_M^l since otherwise a patch $P_{l,z}$ at location z from Ω_l may take pixel values outside of Ω_l . We omit this fact for simplicity.

Algorithm 4 Training layer k of a CKN.

HYPER-PARAMETERS: Kernel parameter α_k , patch size $e_k \times e_k$, number of filters p_k .

INPUT MODEL: A CKN trained up to layer $k-1$.

INPUT DATA: A set of training images.

ALGORITHM:

- Encode the input images using the CKN up to layer $k-1$ by using Algorithm 5;
- Extract randomly n pairs of patches (P_i, P'_i) from the maps obtained at layer $k-1$;
- Normalize the patches to make them unit-norm;
- Learn the model parameters by minimizing (3.12), with $(x_i, x'_i) = (P_i, P'_i)$ for all $i = 1, \dots, n$;

OUTPUT: Weight matrix W_k in $\mathbb{R}^{p_{k-1}e_k^2 \times p_k}$ and b_k in \mathbb{R}^{p_k} .

Algorithm 5 Encoding layer k of a CKN.

HYPER-PARAMETERS: Kernel parameter β_k ;

INPUT MODEL: CKN parameters learned from layer 1 to k by using Algorithm 4;

INPUT DATA: A map $M_{k-1} : \Omega_{k-1} \rightarrow \mathbb{R}^{p_{k-1}}$;

ALGORITHM:

- Extract patches $\{P_{k,z}\}_{z \in \Omega_{k-1}}$ of size $e_k \times e_k$ from the input map M_{k-1} ;
- Compute contrast-normalized patches

$$\tilde{P}_{k,z} = \frac{1}{\|P_{k,z}\|} P_{k,z} \text{ if } P_{k,z} \neq 0 \text{ and } 0 \text{ otherwise.}$$

- Produce an intermediate map $\tilde{M}_k : \Omega_{k-1} \rightarrow \mathbb{R}^{p_k}$ with linear operations followed by non-linearity:

$$\tilde{M}_k(z) = \|P_{k,z}\| e^{W_k^\top \tilde{P}_{k,z} + b_k}, \quad (3.13)$$

where the exponential function is meant “pointwise”.

- Produce the output map M_k by linear pooling with Gaussian weights:

$$M_k(z) = \sum_{u \in \Omega_{k-1}} e^{-\frac{1}{\beta_k^2} \|u-z\|^2} \tilde{M}_k(u).$$

OUTPUT: A map $M_k : \Omega_k \rightarrow \mathbb{R}^{p_k}$.

very ineffective due to the poor conditioning of the optimization problem. One solution is to make another change of variable and optimize in a space where the input data is less correlated.

More precisely, we proceed by (i) adding an “intercept” (the constant value 1) to the vectors x_i in \mathbb{R}^q , yielding vectors \tilde{x}_i in \mathbb{R}^{q+1} ; (ii) computing the resulting (uncentered) covariance matrix $G = \frac{1}{n} \sum_{i=1}^n \tilde{x}_i \tilde{x}_i^\top$; (iii) computing the eigenvalue decomposition of $G = U \Delta U^\top$, where U is orthogonal and Δ is diagonal with non-negative eigenvalues; (iv) computing the preconditioning matrix $R = U(\Delta + \tau I)^{1/2} U^\top$, where τ is an offset that we choose to be the mean value of the eigenvalues. Then, the matrix R may be used as a preconditioner since the covariance of the vectors $R\tilde{x}_i$ is close to identity. In fact, it is equal to the identity matrix when $\tau = 0$ and G is invertible. Then, problem (3.12) with preconditioning becomes

$$\min_Z \sum_{i=1}^n \left[e^{\frac{\|x_i - x'_i\|^2}{2\alpha^2}} - \sum_{j=1}^p e^{z_j^\top R\tilde{x}_i} e^{z_j^\top R\tilde{x}'_i} \right]^2, \quad (3.14)$$

obtained with the change of variable $[W^\top, b^\top] = Z^\top R$. Optimizing with respect to Z to obtain a solution (W, b) turned out to be the key for fast convergence of the stochastic gradient optimization algorithm.

Note that our effort also consisted on implementing heuristics for automatically selecting the learning rate during optimization without requiring any manual tuning, following in part standard guidelines from Bottou [2012]. More precisely, we select the initial learning rate in the following range: $\{1, 2^{-1/2}, 2^{-1}, \dots, 2^{-20}\}$, by performing 1K iterations with mini-batches of size 1000 and choosing the one that gives the lowest objective, evaluated on a validation dataset. After choosing the learning rate, we keep monitoring the objective on a validation set every 1K iteration, and perform backtracking in case of divergence. The learning rate is also divided by $\sqrt{2}$ every 50K iterations. The total number of iterations is set to 300K. Regarding initialization, weights are randomly initialized according to a standard normal distribution. These heuristics are fixed over all experiments and resulted in a stable parameter-free learning procedure, which we will release in an open-source software package.

3.4.7 Different Types of CKNs with Different Inputs

We have not discussed yet the initial choice of the map M_0 for representing an image M . In this chapter, we follow Mairal et al. [2014b] and investigate three possible inputs:

1. **CKN-raw**: We use the raw RGB values. This captures the hue information, which is discriminant information for many application cases.

2. **CKN-white**: It is similar to CKN-raw with the following modification: each time a patch $P_{0,z}$ is extracted from M_0 , it is first centered (we remove its mean color), and whitened by computing a PCA on the set of patches from M_0 . The resulting patches are invariant to the mean color of the original patch and mostly respond to local color variations.
3. **CKN-grad**: The input M_0 simply carries the two-dimensional image gradient computed on grayscale values. The map has two channels, corresponding to the gradient computed along the x-direction and along the y-direction, respectively. These gradients are typically computed by finite differences.

Note that the first layer of CKN-grad typically uses patches of size 1×1 , which are in \mathbb{R}^2 , and which are encoded by the first layer into p_1 channels, typically with p_1 in [8; 16]. This setting corresponds exactly to the kernel descriptors introduced by Bo et al. [2010], who have proposed a simple approximation scheme that does not require any learning. Interestingly, the resulting representation is akin to SIFT descriptors.

Denoting by $(G_{x,z}, G_{y,z})$ the gradient components of image M at location z , the patch P_z is simply the vector $[G_{x,z}, G_{y,z}]$ in \mathbb{R}^2 . Then, the norm $\|P_z\|_2$ can be interpreted as the gradient magnitude $\rho_z = \sqrt{G_{x,z}^2 + G_{y,z}^2}$, and the normalized patch \tilde{P}_z represents a local orientation. In fact, there exists θ_z such that $\tilde{P}_z = [\cos(\theta_z), \sin(\theta_z)]$. Then, we may use the relation (3.8) to approximate the Gaussian kernel $\exp(-\|\tilde{P}_z - \tilde{P}'_{z'}\|^2 / (2\alpha_1^2))$. We may now approximate the integral by sampling p_1 evenly distributed orientations $\theta_j = 2j\pi/p_1$, $: j \in \{1, \dots, p_1\}$, and we obtain, up to a constant scaling factor,

$$e^{-\frac{1}{2\alpha_1^2} \|\tilde{P}_z - \tilde{P}'_{z'}\|^2} \approx \sum_{j=1}^{p_1} e^{-\frac{1}{\alpha_1^2} \|\tilde{P}_z - P_{\theta_j}\|^2} e^{-\frac{1}{\alpha_1^2} \|\tilde{P}'_{z'} - P_{\theta_j}\|^2}, \quad (3.15)$$

where $P_\theta = [\cos(\theta), \sin(\theta)]$. With such an approximation, the j -th entry of the map $\tilde{M}_1(z)$ from (3.13) should be replaced by

$$\rho_z e^{-\frac{1}{2\alpha_1^2} ((\cos(\theta_j) - \cos(\theta_z))^2 + (\sin(\theta_j) - \sin(\theta_z))^2)}.$$

This formulation can be interpreted as a soft-binning of gradient orientations in a “histogram” of size p_1 at every location z . To ensure an adequate distribution in each bin, we choose $\alpha_1 = ((1 - \cos(2\pi/d_1))^2 + \sin(2\pi/d_1)^2)^{1/2}$. After the pooling stage, the representation becomes very close to SIFT descriptors.

A visualization of all input methods can be seen in figure 3.3. See [Mairal et al., 2014a] for more analysis of image preprocessing.

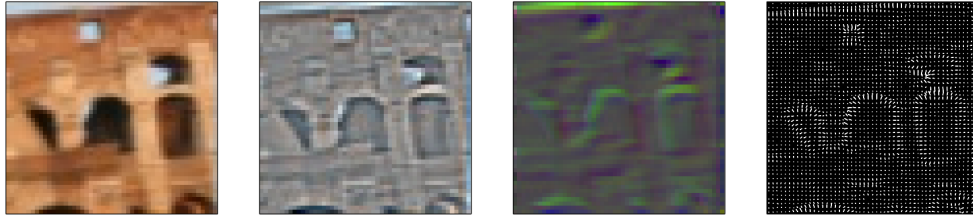


Figure 3.3: Visualization of the possible CKN inputs: CKN-raw (left); input when subtracting the mean color of each patch (middle-left); CKN-white, similar to the previous but with whitening (middle-right) and CKN-grad (right). Patches were extracted with a size of 3×3 ; images are reconstructed by averaging the pixel values of the pre-processed patches (up to normalization to fit the 0-256 range). Best viewed in color on a computer screen.

3.5 Image and Patch Retrieval Datasets

In this section, we give details on the standard datasets we use to evaluate our method, as well as the protocol we used to create our new dataset.

3.5.1 Standard datasets

We give details on the commonly used benchmarks for which we report results.

3.5.1.1 Patch retrieval

The dataset introduced in [Mikolajczyk et al., 2005] is now standard to benchmark patch retrieval methods. This dataset consists of a set of 8 scenes viewed under 6 different conditions, with increasing transformation strength. In contrast to [Winder et al., 2009, Zagoruyko and Komodakis, 2015] where only DoG patches are available, the Mikolajczyk et al. [2005] dataset allows custom detectors. We extract regions with the Hessian-Affine detector and match the corresponding descriptors with Euclidean nearest-neighbor. A pair of ellipses is deemed to match if the projection of the first region using the ground-truth homography on the second ellipse overlaps by at least 50%. The performance is measured in terms of mean average precision (mAP).

3.5.1.2 Image retrieval

We selected three standard image retrieval benchmarks: Holidays, Oxford and the University of Kentucky Benchmark (UKB).

Holidays The Holidays dataset [Jégou et al., 2008] contains 500 different scenes or objects, for which 1,491 views are available. 500 images serve as queries. Following common practice, in contrast to [Babenko et al., 2014] though, we use the unrotated version, which allows certain views to display a 90° rotation with respect to their query. While this has a non-negligible impact on performance for dense representations (the authors of [Babenko et al., 2014] report a 3% global drop in mAP), this is of little consequence for our pipeline which uses rotation-invariant keypoints. External learned parameters, such as k-means clusters for VLAD and PCA-projection matrix are learned on a subset of random Flickr images. The standard metric is mAP.

Oxford The Oxford dataset [Philbin et al., 2007] consists of 5,000 images of Oxford landmarks. 11 locations of the city are selected as queries and 5 views per location is available. The standard benchmarking protocol, which we use, involves cropping the bounding-box of the region of interest in the query view, followed by retrieval. Some works, such as [Babenko et al., 2014] forgo the last step. Such a non-standard protocol yields a boost in performance. For instance Babenko and Lempitsky [2015] report a close to 6% improvement with non-cropped queries. mAP is the standard measure.

UKB Containing 10,200 photos, the University of Kentucky Benchmark (UKB) [Nister and Stewenius, 2006] consists of 4 different views of the same object, under radical viewpoint changes. All images are used as queries in turn, and the standard measure is the mean number of true positives returned in the four first retrieved images ($4 \times \text{recall}@4$).

3.5.2 Rome

One of the goals of this work is to establish a link between performance in patch retrieval with performance in image retrieval. Because of the way datasets are constructed differs (e.g. Internet-crawled images vs successive shots with the same camera, range of viewpoint changes, semantic content of the dataset, type of key-point detector), patch descriptors may display different performances on different datasets. We therefore want a dataset that contains a groundtruth both at patch and image level, to jointly benchmark the two performances. Inspired by the seminal work of [Winder et al., 2009], we introduce the Rome retrieval dataset, based on 3D reconstruction of landmarks. The Rome16K dataset [Li et al., 2010] is a Community-Photo-Collection dataset that consists of 16,179 images downloaded from photo sharing sites, under the search term “Rome”. Images are partitioned into 66 “bundles”, each one containing a set of viewpoints of a given location in Rome

(e.g. “Trevi Fountain”). Within a bundle, consistent parameters were automatically computed and are available⁴. The set of 3D points that were reconstructed is also available, but we choose not to use them in favor of our Hessian-Affine keypoints. To determine matching points among images of a same bundle, we use the following procedure. i) we extract Hessian-Affine points in all images. For each pair of images of a bundle, we match the corresponding SIFTs, using Lowe’s reverse neighbor rule, as well as product quantization [Jégou et al., 2011] for speed-up. We filter matches, keeping those that satisfy the epipolar constraint up to a tolerance of 3 pixels. Pairwise point matches are then greedily aggregated to form larger groups of 2D points viewed from several cameras. Groups are merged only if the reproduction error from the estimated 3D position is below the 3 pixel threshold.

To allow safe parameter tuning, we split the set of bundles into a train and a test set, respectively containing 44 and 22 bundles.

3.5.2.1 Patch retrieval

We design our patch retrieval datasets by randomly sampling in each train and test split a set of 1,000 3D points for which at least 10 views are available. The sampling is uniform in the bundles, which means that we take roughly the same amount of 3D points from each bundle. We then sample 10 views for each point, use one as a query and the remaining as targets. Both our datasets therefore contain 1,000 queries and 9,000 retrieved elements. We report mean average precision (mAP). An example of patch retrieval classes can be seen in Fig. 3.4.

3.5.2.2 Image retrieval

Using the same aforementioned train-test bundle split, we select 1,000 query images and 1,000 target images evenly distributed over all bundles. Two images are deemed to match if they come from the same bundle, as illustrated in Fig. 3.5

3.6 Experiments

In this section, we describe the implementation of our pipelines, and report results on patch and image retrieval benchmarks.

3.6.1 Implementation details

We provide details on the patch and image retrieval pipelines. Our goal is to evaluate the performance of patch descriptors, and all methods are therefore given the

⁴www.cs.cornell.edu/projects/p2f



Figure 3.4: Examples of patches matched under our procedure. We observe significant changes in lighting, but smaller changes in rotation and skew.

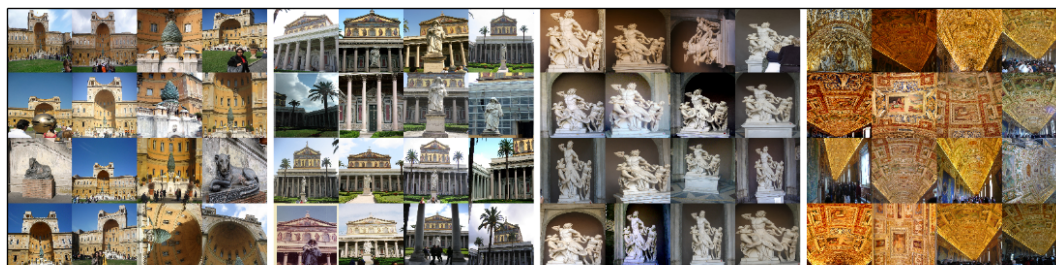


Figure 3.5: Example of classes of the image retrieval dataset of Rome. Each class consists of a particular location. Some bundle display significant viewpoint changes (extreme left and right), while others have little variation in appearance (middle). Best viewed in color.

same input patches (computed at Hessian-Affine keypoints), possibly resized to fit the required input size of the method. We also evaluate all methods with the same aggregation procedure (VLAD with 256 centroids). We believe that improvements in feature detection and aggregation are orthogonal to our contribution and would equally benefit all architectures.



Figure 3.6: Example of matching points in two different images. Salient points are extracted (left), affine rectified (middle), and normalized in rotation (right). Note that the level of invariance which remains to be covered by the patch descriptor is relatively low, as most of the work has been accomplished by the detector.

3.6.1.1 Pipeline

We briefly review our image retrieval pipeline.

Keypoint detection. A popular design choice in image representations, inspired by text categorization methods, is to consider images as sets of local patches, taken at various locations. The choice of these locations is left to the interest point detector, for which multiple alternatives are possible.

In this work, we use the popular “Hessian-Affine” detector of Mikolajczyk and Schmid [2004]. It aims at finding reproducible points, meaning that selected 3D points of a scene should always belong to the set of detected image points when the camera undergoes small changes in settings (e.g. viewpoint, blur, lighting). Because of the “aperture” problem the set of such points is limited to textured patches, to the exclusion of straight lines, for which precise localization is impossible. This leaves “blobs”, as used for instance in Lowe’s Difference of Gaussians (DoG) detector [Lowe, 2004] or corners, as is the case for our Hessian-Affine detector. Specifically, a “corneriness” measure is computed on the whole image, based on the Hessian of the pixel intensities. The set of points whose corneriness is above a threshold is kept. The detector takes the points at their characteristic scale

and estimates an affine-invariant local region. Rotation invariance is achieved by ensuring the dominant gradient orientation always lies in a given direction. This results in a set of keypoints with locally-affine invariant regions. Fig. 3.6 shows the various steps for detecting keypoints. We sample the point with a resolution of 51×51 pixels, value that was found optimal for SIFT on Oxford. Pixels that fall out of the image are set to their nearest neighbor in the image. This strategy greatly increases patch retrieval performance compared to setting them to black, as it does not introduce strong gradients.

Note that the choice of a particular interest point detector is arbitrary and our method is not specific to Hessian-Affine locations. To show this, we also experiment with dense patches in section 3.6.3.5

Patch description. Because of the affine-invariant detectors, and as seen in Fig. 3.6, for a given 3D point seen in two different images, the resulting patches have small differences (e.g. lighting, blur, small rotation, skew). The goal of patch description, and the focus of this work, is to design a patch representation, *i.e.* a mapping Φ of the space of fixed-size patches into some Hilbert space, that is robust to these changes.

Matching/Aggregation. Stereo-vision uses this keypoint representation to establish correspondences between images of the same instance, with 3D reconstruction as an objective. The cost of this operation is quadratic in the number of keypoints, which is prohibitive in image retrieval systems that need to scan through large databases. Instead, we choose to aggregate the local patch descriptors into a fixed-length global image descriptor. For this purpose, we use the popular VLAD representation [Jégou et al., 2012]. Given a clustering in the form of a Voronoi diagram with points $\{c_1, \dots, c_k\}$ of the feature space (typically obtained using k-means on an external set of points), VLAD encodes the set of visual words $\{x_1, \dots, x_n\}$ as the total shift with respect to their assigned centroid:

$$\text{VLAD} = \sum_{i=1}^k \sum_{j=1}^n \varepsilon_{i,j} (x_j - c_i), \quad (3.16)$$

where $\varepsilon_{i,j}$ is the assignment operator, which is 1 if x_j is closer to centroid c_i than to the others, and 0 otherwise.

The final VLAD descriptors is power-normalized with exponent 0.5 (signed square-root), as well as ℓ_2 -normalized.

3.6.1.2 Convolutional Networks training

AlexNet-ImageNet. We use the Caffe package [Jia et al., 2014] and its provided weights for AlexNet, that have been learned according to Krizhevsky et al. [2012]. Specifically, the network was trained on ILSVRC’12 data for 90 epochs, using a learning rate initialized to 10^{-2} and decreased three times prior to termination. Weight decay was fixed to 0.0005, momentum to 0.9 and dropout rate to 50%. It uses three types of image jittering: random 227×227 out of 256×256 cropping, flip and color variation.

AlexNet-Landmarks. Following Babenko et al. [2014], we fine-tune AlexNet using images of the Landmarks dataset⁵. Following their protocol, we strip the network of its last layer and replace it with a 671-dim fully-connected one, initialized with random Gaussian noise (standard deviation 0.01). Other layers are initialized with the old AlexNet-ImageNet weights. We use a learning rate of 10^{-3} for fine-tuning. Weight decay, momentum and dropout rate are kept to their default values (0.0005, 0.9 and 0.5). We use data augmentation at training time, with the same transformations as in the original paper (crop, flip and color). We decrease the learning rate by 10 when it saturates (around 20 epochs each time). We report a validation accuracy of 59% on the Landmarks dataset, which was confirmed through discussion with the authors. On Holidays, we report a mAP of 77.5 for the sixth layer (against 79.3 in [Babenko et al., 2014]), and 53.6 (against 54.5) on Oxford. Even though slightly below the results in the original paper, fine-tuning still significantly improves on ImageNet weights for retrieval.

PhilippNet. For PhilippNet, we used the model provided by the authors. The model is learned on 16K surrogate classes (randomly extracted patches) with 150 representatives (composite transformations, including crops, color and contrast variation, blur, flip, etc.). We were able to replicate their patch retrieval results on their dataset, as well as on Mikolajczyk et al’s dataset when using MSER keypoints.

PhilippNet-Rome. The patch retrieval dataset of RomePatches does not contain enough patches to learn a deep network. We augment it using patches extracted in a similar fashion, grouped in classes that correspond to 3D locations and contain at least 10 examples. We build two such training sets, one with 10K classes, and one with 100K classes. Training is conducted with the default parameters.

⁵<http://sites.skoltech.ru/compvision/projects/neuralcodes/>

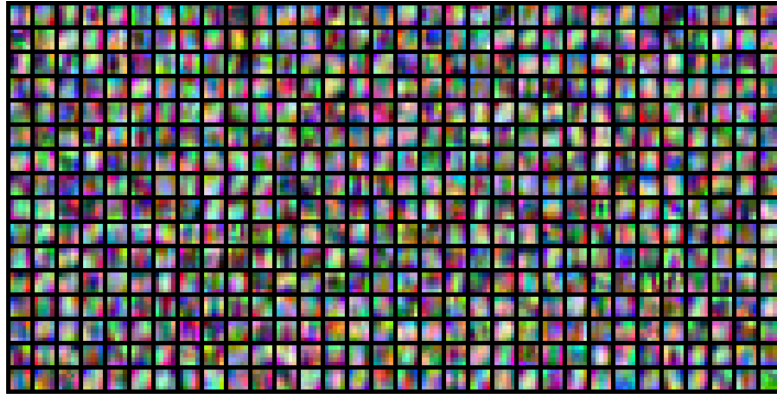


Figure 3.7: All 512 kernels of the first layer of our CKN-raw architecture.

Deepcompare. As previously described, we use the online code provided by Zagoruyko and Komodakis [2015]. It consists of networks trained on the three distinct datasets of Winder et al. [2009]: Liberty, NotreDame and Yosemite. For our image retrieval experiments, we can only use the siamese networks, as the others do not provide a patch representation. These were observed in the original paper to give suboptimal results.

Convolutional Kernel Networks To train the convolutional kernel networks, we randomly subsample a set of 100K patches in the train split of the Rome dataset. For each layer, we further extract 1M sub-patches with the required size, and feed all possible pairs as input to the CKN. The stochastic gradient optimization is run for 300K iterations with a batch size of 1000, following the procedure described in section 3.4.6. Training a convolutional kernel network, for a particular set of hyperparameters, roughly takes 10 min on a GPU. This stands in contrast to the 2-3 days for training using the L-BFGS implementation used in Mairal et al. [2014b]. We show in Fig. 3.7 a visualization of the first convolutional filters of CKN-raw. The sub-patches for CKN-grad and CKN-white are too small (3×3) and not adapted to viewing.

3.6.2 Patch retrieval

3.6.2.1 CKN parametric exploration

The relatively low training time of CKNs (10 min. on a recent GPU), as well as the fact that the training is layer-wise – and therefore lower layer parameters can be reused if only the top layer change – allows us to test a relatively high number of parameters, and select the ones that best suit our task. We tested parameters for

convolution and pooling patch sizes in range of 2 to 5, number of neuron features in powers of 2 from 128 to 1024. For the σ parameter, the optimal value was found to be 10^{-3} for all architectures. For the other parameters, we keep one set of optimal values for each input type, described in Table 3.1.

Input	Layer 1	Layer 2	dim.
CKN-raw	5x5, 5, 512	—	41,472
CKN-white	3x3, 3, 128	2x2, 2, 512	32,768
CKN-grad	1x1, 3, 16	4x4,2,1024	50,176

Table 3.1: For each layer we indicate the sub-patch size, the subsampling factor and the number of filters. For the gradient network, the value 16 corresponds to the number of orientations.

In Figure 3.8, we explore the impact of the various hyper-parameters of CKN-grad, by tuning one while keeping the others to their optimal values.

3.6.2.2 Dimensionality reduction

Since the final dimension of the CKN descriptor is prohibitive for most applications of practical value (see Table 3.1), we investigate dimensionality reduction techniques. Note that this step is unnecessary for CNNs, whose feature dimension do not exceed 512 (PhilippNet). We only perform unsupervised dimensionality reduction through PCA, and investigate several forms of whitening. Denoting X the $n \times d$ matrix of n CKN features (we used $n = 10K$), the singular value decomposition of X writes as

$$X = USV^\top, \quad (3.17)$$

where U and V are orthogonal matrices and S is diagonal with non-increasing values from upper left to bottom right. For a new $n' \times d$ matrix of observations X' , the dimensionality reduction step writes as

$$X'_{\text{proj}} = XL^\top, \quad (3.18)$$

where L is a $d' \times d$ projection matrix.

The three types of whitening we use are: i) no whitening; ii) full whitening; iii) semi-whitening. Full whitening corresponds to

$$L = V(1:d', :)/D(1:d', 1:d'), \quad (3.19)$$

while semi-whitening corresponds to

$$L = V(1:d', :)/\sqrt{D(1:d', 1:d')}. \quad (3.20)$$

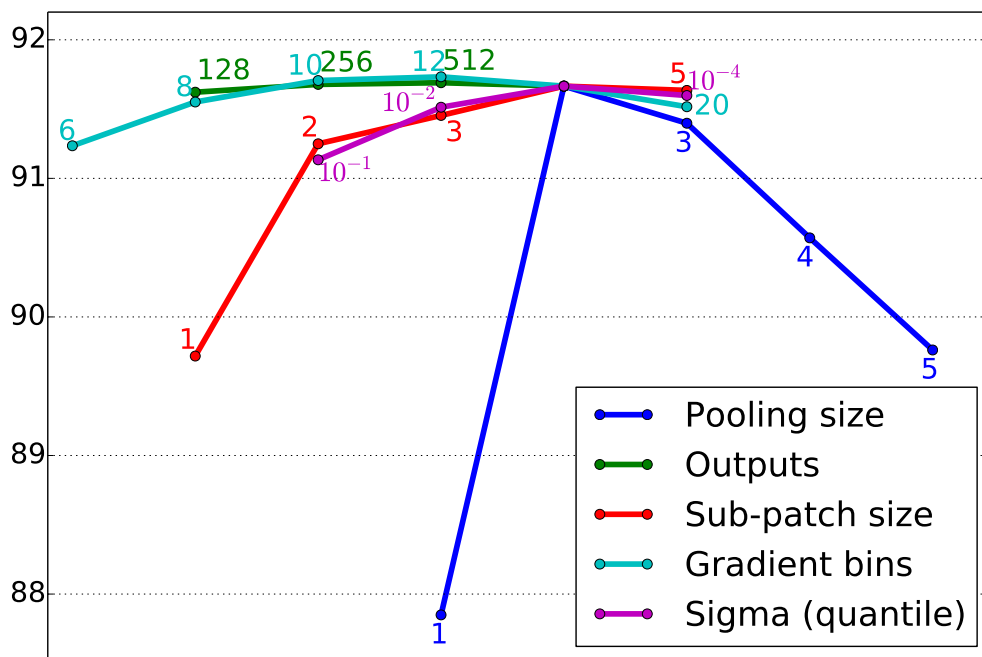


Figure 3.8: mAP results on the train set of RomePatches with CKN-grad, whose hyper-parameters have been changed one by one around the optimal point. Note that 10 and 12 gradient bins give slightly better results but 16 was kept to align with the logarithm scale of the grid.

The matrix division $/$ denotes the matrix multiplication with the inverse of the right-hand matrix. The square-root is performed element-wise.

Results of the different methods on the RomePatches dataset are displayed in Fig. 3.9. We observe that semi-whitening works best for CKN-grad and CKN-white, while CKN-raw is slightly improved by full whitening. We keep these methods in the remainder of the experiments, as well as a final dimension of 1024.

3.6.2.3 Results

We compare the convolutional architectures on our three patch datasets: RomePatches-train, RomePatches-test and Mikolajczyk et al’s dataset. Results are given in Table 3.2. For AlexNet CNNs, we report results for all outputs of the 5 convolutional layers (after ReLU). We note that SIFT is an excellent baseline for these methods, and that CNN architectures that were designed for local invariances perform better than the ones used in AlexNet, as observed by Fischer et al. [2014]. The results of the PhilippNet on Mikolajczyk et al’s dataset are different from the ones reported by Fischer et al. [2014], as we evaluate on Hessian-Affine descriptors while they

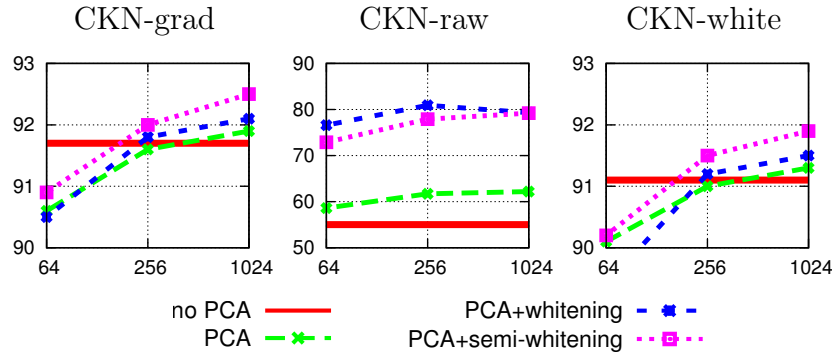


Figure 3.9: Influence of dimensionality reduction on patch retrieval performance. Results reported in mAP (%) on the train split of RomePatches as a function of the PCA dimension. As a comparison, SIFT reports 91.6%.

Architecture	coverage	Dim	RomePatches		Miko.
			train	test	
SIFT	51x51	128	91.6	87.9	57.8
AlexNet-conv1	11x11	96	66.4	65.0	40.9
AlexNet-conv2	51x51	256	73.8	69.9	46.4
AlexNet-conv3	99x99	384	81.6	79.2	53.7
AlexNet-conv4	131x131	384	78.4	75.7	43.4
AlexNet-conv5	163x163	256	53.9	49.6	24.4
PhilippNet	64x64	512	86.1	81.4	59.7
PhilippNet	91x91	2048	88.0	83.7	61.3
CKN-grad	51x51	1024	92.5	88.1	59.5
CKN-raw	51x51	1024	79.3	76.3	50.9
CKN-white	51x51	1024	91.9	87.7	62.5

Table 3.2: Results of convolutional architectures for patch retrieval.

use MSER. To have a comparable setting, we use their network with an input of 64x64 that corresponds to the coverage of one top neuron, as well as their protocol that slide it on 91x91 patches. We notice that this last step only provides a small increase of performance (2% for patch retrieval and 1% for image retrieval). We observe that PhilippNet outperforms both SIFT and AlexNet, which was the conclusion of Fischer et al. [2014]; CKN trained on whitened patches do however yield better results.

3.6.2.4 DeepCompare

As the architectures of DeepCompare [Zagoruyko and Komodakis, 2015] do not rely on an underlying patch descriptor representation but rather on similarities between pairs of patches, some architectures can only be tested for patch-retrieval. We give in Table 3.3 the performances of all their architectures on RomePatches.

Network	RomePatches-Train	RomePatches-Test
2ch2stream Y	89.3	85.7
2ch2stream ND	88.3	84.9
2ch2stream L	90.2	86.6
2ch Y	86.6	83.5
2ch ND	81.2	78.4
2ch L	83.4	80.1
siam Y	82.9	79.2
siam ND	84.8	81.0
siam L	83.2	79.8
siam2stream Y	78.8	75.4
siam2stream ND	84.2	80.6
siam2stream L	82.0	78.6

Table 3.3: Evaluation of the deepcompare architectures on RomePatches in mAP (%). Networks were trained on the three subsets of the Multi-view Stereo Correspondence Dataset: Yosemite (Y), Notre-Dame (ND) and Statue of Liberty (L). The network notations are as in the original paper [Zagoruyko and Komodakis, 2015].

We note that the only networks that produce descriptors that can be used for image retrieval are the architectures denoted “siam” here. They also perform quite poorly compared to the others. Even the best architectures are still below the SIFT baseline (91.6/87.9).

The method of Zagoruyko and Komodakis [2015] optimizes and tests on different patches (DoG points, 64x64 patches, greyscale), which explains the poor performances when transferring them to our RomePatches dataset. The common evaluation protocol on this dataset is to sample the same number of matching and non-matching pairs, rank them according to the measure and report the false positive rate at 95% recall (“FPR@95%”, the lower the better). The patches used for the benchmark are available online, but not the actual split used for testing. With our own split of the Liberty dataset, we get the results in Table 3.4. We note that all our results, although differing slightly from the ones of Zagoruyko and Komodakis [2015], do not change their conclusions.

Descriptor	FPR@95% (%)
SIFT	19.9
Best AlexNet (3)	13.5
siam-l2 (trained on Notre-Dame)	14.7
2ch-2stream (trained on Notre-Dame)	1.24
CKN-grad	27.7
CKN-white	30.4
CKN-raw	41.7
best CKN	14.4

Table 3.4: Experiments on a set of pairs of the Liberty dataset. Measure is false positive rate @95% recall, and therefore the lower the better. The third layer of AlexNet is used, as it provides the best results.

We note that our CKNs whose hyperparameters were optimized on RomePatches work poorly. However, optimizing again the parameters for CKN-grad on a grid leads to a result of 14.4% (“best CKN”). To obtain this result, the number of gradient histogram bins was reduced to 6, and the pooling patch size was increased to 8 on the first layer and 3 on the second. These changes indicate that the DoG keypoints are less invariant to small deformations, and therefore require less rigid descriptors (more pooling, less histograms). The results are on par with all comparable architectures (siam-l2: 14.7%, AlexNet: 13.5%), as the other architectures either use central-surround networks or methods that do not produce patch representations. The best method of Zagoruyko and Komodakis [2015], 2ch-2stream, reports an impressive 1.24% on our split. However, as already mentioned, this architecture does not produce a patch representation and it is therefore difficult to scale to large-scale patch and image retrieval applications.

3.6.2.5 Impact of supervision

We study the impact of the different supervised trainings, between surrogate classes and real ones. Results are given in Table 3.5.

Figure 3.10 gives the first convolutional filters of the PhilippNet learned on surrogate classes, and on Rome. As can be seen, the plain surrogate version reacts to more diverse color, as it has been learned with diverse (ImageNet) input images. Both Rome 10K and Rome 100K datasets have colors that correspond to skies and buildings and focus more on gradients. It is interesting to note that the network trained with 100K classes seems to have captured finer levels of detail than its 10K counterpart.

Training parameters	RomePatches-Train	RomePatches-Test
PhilippNet	86.1	81.4
PhilippNet-Rome 10K	84.1	80.1
PhilippNet-Rome 100K	89.9	85.6
(Best) CKN-grad	92.5	88.1
(Baseline) SIFT	91.6	87.9

Table 3.5: Impact of supervision on patch retrieval. PhilippNet is trained on surrogate classes, while PhilippNet-Rome is trained on a larger set of RomePatches-Train, containing either 10K or 100K classes. We see that retraining improves performance provided enough data is given. Supervised CNNs are still below the SIFT baseline, as well as the unsupervised CKNs.

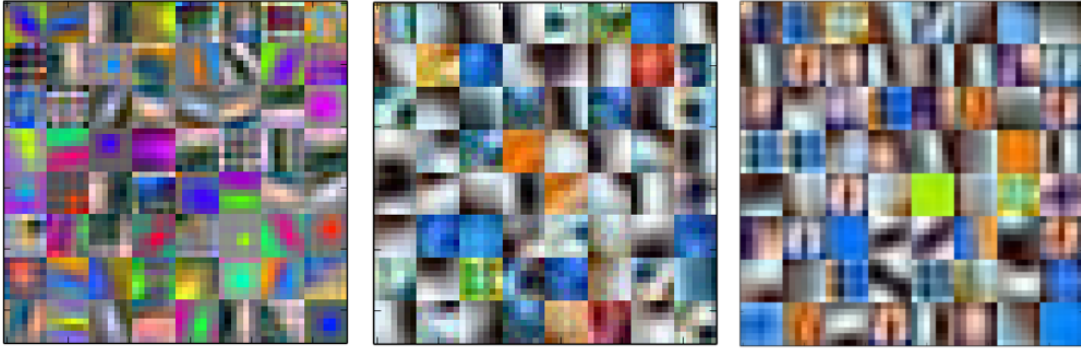


Figure 3.10: First convolutional filters of the PhilippNet learned with surrogate classes (left), 10K Rome classes (middle) and 100K Rome classes. Best viewed in color.

3.6.2.6 Robustness

We investigate in Fig. 3.11 the robustness of CKN descriptors to transformations of the input patches, specifically rotations, zooms and translations. We select 100 different images, and extract the same patch by jittering the keypoint along the aforementioned transformation. We then plot the average L_2 distance between the descriptor of the original patch and the transformed one. Note the steps for the scale transformation; this is due to the fact that the keypoint scale is quantized on a scale of powers of 1.2, for performance reasons.

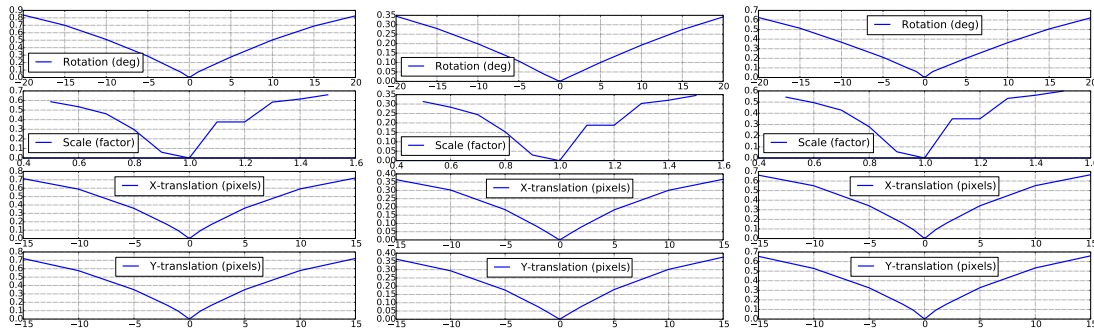


Figure 3.11: Impact of various transformations on CKN descriptors. Represented is the average distance between a patch descriptor and its jittered version, as a function of the transformation magnitude. Best viewed in numerical format.

3.6.3 Image retrieval

3.6.3.1 Settings.

We learn a vocabulary of 256 centroids on a related database: for Holidays and UKB we use 5,000 Flickr images and for Oxford, we train on Paris [Philbin et al., 2008]. The vocabulary for RomePatches-Train is learned on RomePatches-Test and vice-versa. The final VLAD descriptor size is 256 times the local descriptor dimension.

3.6.3.2 Results

We compare all convolutional approaches as well as the SIFT baseline in the image retrieval settings. Results are summarized in Table 3.6.

On datasets for which color is dominant (e.g. Holidays or UKB), the best individual CKN results are attained by CKN-white, improved by combining the three channels. On images of buildings, gradients still perform best and the addition of color channels is harmful, which explains on the one hand the poor performance of AlexNet and on the other hand the relatively good performance of PhilippNet which was explicitly trained to be invariant to colorimetric transformations.

3.6.3.3 Influence of context

Through experiments with AlexNet-landmarks, we study the impact of context in the local features. Specifically, we test whether fine-tuning on a dataset that shares semantic information with the target retrieval dataset improves performance. We compare the same network architecture – AlexNet – in two settings: when parameters are learned on ImageNet (which involves a varied set of classes) and when

	Holidays	UKB	Oxford	Rome	
				train	test
SIFT	64.0	3.44	43.7	52.9	62.7
AlexNet-conv1	59.0	3.33	18.8	28.9	36.8
AlexNet-conv2	62.7	3.19	12.5	36.1	21.0
AlexNet-conv3	79.3	3.74	33.3	47.1	54.7
AlexNet-conv4	77.1	3.73	34.3	47.9	55.4
AlexNet-conv5	75.3	3.69	33.4	45.7	53.1
PhilippNet 64x64	74.1	3.66	38.3	50.2	60.4
PhilippNet 91x91	74.7	3.67	43.6	51.4	61.3
CKN-grad	66.5	3.42	49.8	57.0	66.2
CKN-raw	69.9	3.54	23.0	33.0	43.8
CKN-white	78.7	3.74	41.8	51.9	62.4
CKN-mix	79.3	3.76	43.4	54.5	65.3

Table 3.6: Image retrieval results. Results are in mAP except for UKB where we measure the average number of true positives in the first 4 results. CKN-mix is the result of the concatenation of the VLAD descriptors for the three channels.

	Holidays		Oxford	
	ImageNet	Landmarks	ImageNet	Landmarks
conv1	59.0	57.7	18.8	20.7
conv2	62.7	72.4	12.5	23.8
conv3	79.3	76.1	33.3	38.6
conv4	77.1	73.9	34.3	38.3
conv5	75.3	66.1	33.4	32.5

Table 3.7: Influence of the pretraining dataset on image retrieval. With the same architecture (AlexNet), training is conducted either on ILSVRC’12 data or on the Landmarks dataset. Using semantic information related to buildings and places yields improvements for Oxford, but not for Holidays.

they are learned on the Landmarks dataset which solely consists of buildings and places. Results, shown in Table 3.7, show clear improvement for Oxford, but not for Holidays. We explain this behavior by the fact that the network learns building-specific invariances and that fewer building structures are present in Holidays as opposed to Oxford.

Dataset	Holidays (CKN-mix)	UKB (CKN-mix)	Oxford (CKN-grad)
Full (262K-dim)	79.3	3.76	49.8
PCA (4096-dim)	82.9	3.77	47.2

Table 3.8: Impact of dimensionality reduction by PCA+whitening on the best channel for each dataset. PCA to 4096 dimensions.

3.6.3.4 Dimensionality reduction

As observed in previous work [Jégou and Chum, 2012], projecting the final VLAD descriptor to a lower dimension using PCA+whitening can lead to lower memory costs, and sometimes to slight increases in performance (e.g. on Holidays [Gong et al., 2014]). We project to 4096-dim descriptors, the same output dimension as Babenko et al. [2014] and obtain the results in Table 3.8. We indeed observe a small improvement on Holidays but a small decrease on Oxford.

3.6.3.5 Dense keypoints

Our choice of Hessian-Affine keypoints is arbitrary and can be suboptimal for some image retrieval datasets. Indeed we observe that by sampling points on a regular grid of 8×8 pixels at multiple scales, we can improve results. We learn the CKN parameters and the PCA projection matrix on a dense set of points in Rome, and apply it to image retrieval as before. We use SIFT descriptors as a baseline, extracted at the exact same locations. We observe that for CKN-grad and CKN-white, the previous models lead to suboptimal results. However, increasing the pooling from 3 (resp. 2 for the second layer) to 4 (resp. 3), leads to superior performances. We attribute this to the fact that descriptors on dense points require more invariance than the ones computed at Hessian-Affine locations. Results on the Holidays dataset are given in Table 3.9. As observed on Hessian-Affine points, the gradient channel performs much better on Oxford. We therefore only evaluate this channel. As explained in section 3.5, there are two ways to evaluate the Oxford dataset. The first one crops queries, the second does not. While we only consider the first protocol to be valid, it is interesting to investigate the second, as results in Table 3.10 tend to indicate that it favors dense keypoints (and therefore the global descriptors of Babenko et al. [2014]).

3.6.3.6 Comparison with state of the art

Table 3.11 compares our approach to recently published results. Approaches based on VLAD with SIFT [Arandjelovic and Zisserman, 2013, Jégou et al., 2012] can be

Descriptor	SIFT	grad	raw	white	mix
Hessian-Affine	64.0	66.5	69.9	78.7	79.3
Dense (same parameters)	70.3	68.5	72.3	76.8	.
Dense (changed pooling)	70.3	71.3	72.3	80.8	82.6

Table 3.9: Dense results on Holidays. Right hand side of the table are CKN descriptors. “Same parameters” correspond to CKNs that have the same parameters as Hessian-Affine ones, yet learned on densely extracted patches. “Changed pooling” have pooling size increased by one at each layer (CKN-raw is unchanged as it only has one layer, and already gives good performances).

Method \ descriptor	SIFT	CKN-grad
Hessian-Affine, no crop	45.7	49.0
Hessian-Affine, crop	43.7	49.8
Dense, no crop	51.1	55.4
Dense, crop	47.6	50.9

Table 3.10: Dense keypoints on the Oxford dataset. “Crop” indicates the protocol where queries are cropped to a small bounding-box containing the relevant object, while “no crop” takes the full image as a query. For CKN, parameters have increased pooling sizes for dense keypoints, as on Holidays.

improved significantly by CKN local descriptors (+15% on Holidays). To compare to the state of the art with SIFT on Oxford [Arandjelovic and Zisserman, 2013], we use the same Hessian-Affine patches extracted with gravity assumption [Perd’och et al., 2009]. Note that this alone results in a 7% gain.

We also compare with global CNN [Babenko et al., 2014]. Our approach outperforms it on Oxford, UKB, and Holidays. For CNN features with sum-pooling encoding [Babenko and Lempitsky, 2015], we report better results on Holidays and UKB, and on Oxford with the same evaluation protocol. Note that their method works better than ours when used without cropping the queries (58.9%).

On Holidays, our approach is slightly below the one of Gong et al. [2014], that uses AlexNet descriptors and VLAD pooling on large, densely extracted patches. It is however possible to improve on this result by using the same dimensionality reduction technique (PCA+whitening) which gives 82.9% or dense keypoints (82.6%).

Method \ Dataset	Holidays	UKB	Oxford
VLAD [Jégou et al., 2012]	63.4	3.47	-
VLAD++ [Arandjelovic and Zisserman, 2013]	64.6	-	55.5*
Global-CNN [Babenko et al., 2014]	79.3	3.56	54.5
MOP-CNN [Gong et al., 2014]	80.2	-	-
Sum-pooling OxfordNet [Babenko and Lempitsky, 2015]	80.2	3.65	53.1
Ours	79.3	3.76	49.8
Ours+PCA 4096	82.9	3.77	47.2

Table 3.11: Comparison with state-of-the-art image retrieval results for global descriptors.

Chapter 4

Conclusion

4.1	Summary of Contributions	81
4.2	Future Directions	82

4.1 Summary of Contributions

The focus of this thesis is on learning representations that are invariant to nuisance factors such as geometric variability

A standard approach to learning invariant image classifiers is to generate transformed versions of the original images. However, major challenges are which transformations to select (*ie*, those that yield transformation improvements without highly increasing the training cost), and how to incorporate the transformed examples into the classification process. We proposed in Chapter 2 an algorithm called Image Transformation Weighting that is able to efficiently select a set of weighting parameters to maximize classification accuracy. Its relaxation, called Image Transformation Pursuit, yields a selection of the best performing transformations, independently of the dataset, the initial set of transformations, the classification loss, or the way to aggregate transformed examples. It is also more scalable than ITW. We tested our algorithms on three standard fine-grained classification datasets and achieved significant improvements in terms of classification accuracy with only a handful of transformations, for several features such as Fisher Vectors and off-the-shelf CNNs. Another conclusion of our work is that among all aggregation schemes for transformed examples, data augmentation performs best, provided test images are also transformed and their individual scores averaged for the final evaluation.

In Chapter 3, we showed that Convolutional Kernel Networks (CKNs) offer similar and sometimes even better performances than classical Convolution Neural Networks (CNNs) in the context of patch description, and that the good performances observed in patch retrieval translate into good performances for image retrieval, reaching competitive results on several standard benchmarks. The main advantage of CKNs compared to CNNs is their very fast training time, and the fact that unsupervised training suppresses the need for manually labeled examples. We introduced a new patch descriptor called “Patch-CKN” and reported superior results to both SIFT and other convolutional descriptors, on a new dataset, as well as on standard benchmarks for patch and image retrieval.

Last, both contributions have shown that it is possible to reduce the time spent annotating images. Virtual examples are always less informative than real ones, yet they come at no cost because they can be generated efficiently. On the other hand, we have shown in Chapter 3 that our new local descriptors not only outperform hand-crafted and supervisedly learned features, but also only need unsupervised data to learn their parameters. Practically, this means that our framework requires no annotated data, which in the case of local descriptors can be of low quality. If our descriptors were to be used for more semantically-oriented tasks such as classification, this could also help reducing the amount of human labor required in data acquisition.

4.2 Future Directions

Our work can be extended in different ways. We now propose three directions.

Transformation selection and supervised learning. One possible extension of our work is the application of our transformation selection procedures to end-to-end learning. Indeed, fine-tuning gives state-of-the-art performance for fine-grained image classification, and the application of ITP in this context is an open problem. Because our algorithm requires testing each individual transformation several times, it requires fully learning a CNN each time, which is unpractical. One could imagine several variants: for instance alternating one step of end-to-end-learning with the selected transformations, and one step of descriptor extraction followed by ITP to select the transformations. Our Image Transformation Weighting algorithm (C.f. Chapter 2) could also be used in parallel of training to compute a new weighting of virtual examples. A tentative algorithm would run like this:

1. Select an initial weighting of transformations, for instance a uniform weighting.
2. Feed the examples to learn the neural network. Use weighting sampling to select which transformation is to be used.
3. Every K iterations of the learning algorithm, extract transformed examples with the current network parameters.
4. Using ITW, compute an optimal weighting of transformations
5. Update the weighting and continue learning with it.

The ITW algorithm would be tractable as deep learning intermediate representations are usually low-dimensional (4096).

Patch-CKN for local descriptor search. Our initial goal for the Patch-CKN descriptors of Chapter 3 was to replace SIFT as an all-purpose local descriptor. We have shown the benefits of using Patch-CKNs in place of SIFT descriptors for patch matching as well as the computation of aggregated descriptors such as VLAD. One of the state-of-the-art approaches for image retrieval relies today on local descriptor indexing and searching, as popularized by the Hamming Embedding scheme of Jégou et al. [2008] and more recently ASMK [Tolias et al., 2013]. It is easy to replace the SIFT descriptor of these approaches with our descriptors or any other one, and expect to get better results. As it turns out, this is not directly the case, and preliminary experiments have shown that such a naive attempt does not give the expected results. Several factors can play a role in this phenomenon. First,

indexing methods typically start by heavy quantization (to 64 bits for HE, and 8 bit for the inverted files of ASMK) of the local descriptors. We have shown that our Patch-CKN descriptors display better behaviour around 1024 dimensions. Second, SIFTs are histograms and therefore have very different statistical distributions from our deep descriptors. Last, most works that use SIFT have special normalization against burstiness (see, e.g., the work of Jégou et al. [2009]), which do not translate well to Patch-CKNs. It is suspected that these caveats can be lifted with careful experiments and in particular proper local normalization. Since our descriptors have been optimized for local description and patch-matching, they should also perform better for searching and indexing, which is closer to the original task than aggregation. Geometric verification should also benefit from our approach as it only relies on local descriptor quality. These are possible directions to enhance Patch-CKNs methods.

Dense Patch-CKN. One other direction for our new local descriptor Patch-CKNs of Chapter 3 is the application to densely extracted keypoints, which have been shown to give better performance for retrieval as well as classification. This was partially covered in our work, but in a rather inefficient way, by extracting all dense coordinates and computing independently one descriptor per patch (this was done out of fairness, to be able to get exactly the same input for each descriptor, and because knowing the exact coverage and location of a neuron of a higher layer into the image is difficult) Yet, convolutional architectures have the advantage that lower-level convolutions can be factorized when one needs to extract several region descriptors. This is in particular used in image detection with the work of Girshick et al. [2014]. The resulting feature map could be used as an input to VLAD aggregation, or directly to a classifier for image classification. The early works on CKNs of Mairal et al. [2014b] used these unsupervised feature maps for classification but only restricted themselves to unrealistic datasets such as MNIST (digits) and CIFAR (32x32 images). One interesting possible future work would be to have an unsupervised convolutional descriptor relying on CKNs, able to perform classification on large-scale datasets similar to ImageNet. There are two main problems for this approach. First, end-to-end learning allows to efficiently handle data augmentation, and allows transformations of the input images at almost no cost, while methods with precomputed descriptors suffer from longer extraction times. This could be efficiently handled by our ITP algorithm, which provides a way to directly link our two main contributions. Applying ITP to perform supervised learning of features on ImageNet has already been used with a great bit of success by Perronnin and Larlus [2015]. The second difficulty is that unsupervisedly learned convolutional layers cannot efficiently filter non-discriminative information, and therefore needs to be have higher dimensional feature maps to

stay competitive with their supervised counterparts. This is the main downside of this approach and would prove untractable if too many neurons are required. One possible way to circumvent this would be to perform a supervised projection step, for instance with Linear Discriminant Analysis after each layer. This would effectively resemble the approach of Chan et al. [2014].

Appendix A

Publications

This thesis has led to several publications, which are listed below.

International Conferences

- Transformation Pursuit for Image Classification. Mattis Paulin, Jerome Revaud, Zaid Harchaoui, Florent Perronnin and Cordelia Schmid. International Conference on Computer Vision and Pattern Recognition. 2014.
- Local Convolutional Features with Unsupervised Training for Image Retrieval Mattis Paulin, Matthijs Douze, Zaid Harchaoui, Julien Mairal, Florent Perronnin and Cordelia Schmid, International Conference on Computer Vision. 2015.

International Journals

- Convolutional Patch Representations for Image Retrieval: an Unsupervised Approach. Mattis Paulin, Julien Mairal, Matthijs Douze, Zaid Harchaoui, Florent Perronnin and Cordelia Schmid. Accepted at the International Journal for Computer Vision.

Others

- Selection itérative de transformations pour la classification d'images. Mattis Paulin, Jerome Revaud, Zaid Harchaoui, Florent Perronnin, Cordelia Schmid Reconnaissance de Formes et Intelligence Artificielle 2014.

- The INRIA-LIM-VocR and AXES submissions to Trecvid 2014 Multimedia Event Detection. Matthijs Douze, Dan Oneata, Mattis Paulin, Clément Leray, Nicolas Chesneau, Danila Potapov, Jakob Verbeek, Karteek Alahari, Zaid Harchaoui, Lori Lamel, Jean-Luc Gauvain, Christoph Andreas Schmidt, Cordelia Schmid. Technical report. 2014.

Appendix B

Released Software

This thesis has led to three major software projects released by the author, detailed in the following appendix. The first one is an all-purpose stochastic gradient toolbox called JS GD, the two following correspond to the two main chapters.

B.1 JS GD

JS GD is a stochastic gradient descent toolbox for large-scale multiclass classification problems, and was co-developed with Matthijs Douze (currently Facebook AI Research). It contains

- Code to solve large-scale multiclass classification problems up to millions of examples in thousands of dimensions with seven different losses and many averaging options.
- Detailed instructions on how to install.
- Python and Matlab bindings.
- A readable matlab code designed to be comprehensive rather than fast.
- A realistic test samples with Bag-of-Words features.

To allow for large distribution, we designed a comprehensive website ¹, featuring sources and test data, details on the algorithms, an FAQ and a detailed installation walkthrough.

We now describe the technical content of the toolbox.

¹<http://lear.inrialpes.fr/src/jsgd>

B.1.1 Stochastic gradient descent

The idea behind stochastic gradient descent (SGD) is to use a non-deterministic oracle for optimization, that whose expectation is equal to the true oracle. In particular, and this is the case for the JSJD toolbox, we study losses that write as empirical means of example-wise losses $\ell_i(w) = \ell(x_i, y_i|w)$. In this case, (x_i, y_i) are couples of (example,label) for a multiclass classification task. The optimization problem writes as:

$$\underset{w}{\text{Optimize}} \Omega(w) + \frac{1}{n} \sum_{i=1}^n \ell_i(w), \quad (\text{B.1})$$

where Ω is a regularization term.

This problem is the empirical mean of the general expectation problem:

$$\underset{w}{\text{Optimize}} \mathbb{E}_{x,y} [\Omega(w) + \ell(x, y|w)]. \quad (\text{B.2})$$

Therefore if i is drawn uniformly, $\Omega(w) + \ell_i(w)$ is an approximate oracle for the original problem. Stochastic gradient descent theory states that optimizing such an oracle with simple gradient descent optimizes the full problem with a convergence rate that is independant of the number of examples n [Bottou and Bousquet, 2007]. This particularity can be extremely useful when dealing with large-scale datasets.

The optimization algorithm draws a sample (x_t, y_t) at each iteration and uses the update rule:

$$w_{t+1} \rightarrow w_t - \eta_t \nabla [\Omega + \ell_t](w_t), \quad (\text{B.3})$$

where η_t is a decreasing function that verifies $\sum_{t=1}^{\infty} \eta_t = \infty$.

It can be shown [Bottou and Bousquet, 2007] that such an algorithm converges for convex functions with a linear convergence rate for smooth ones.

B.1.2 Featured losses

The JSJD Toolbox proposes several standard machine learning losses for multi-class classification. For a given weight parameter w and bias b , these write as an average over (x, y) tuples of (examples,labels) of an elementary loss $L(w, b, x, y)$.

The proposed losses are:

Weighted One-versus-rest.

$$L_{\text{OVR}}(w, b, x, y) = \sum_{\ell=1}^C \sum_{i=1}^n \max\{0, 1 - y_{i,\ell}(w_{\ell}x_i + b_{\ell})\}, \quad (\text{B.4})$$

with

$$y_{i,\ell} = \begin{cases} 1 & \text{if } y_i = \ell \\ -1 & \text{if } y_i \neq \ell \end{cases} \quad (\text{B.5})$$

Multiclass Structured Support Vector Machine (MUL)

$$L_{\text{MUL}}(w, b, x, y) = \sum_{i=1}^n \max_y (w_y x_i + b_y - w_{y_i} x_i - b_{y_i} + \Delta(y, y_i)). \quad (\text{B.6})$$

Multiclass Squared Hinge Loss (MUL2)

$$L_{\text{MUL2}}(w, b, x, y) = \sum_{i=1}^n [\max_y (w_y x_i + b_y - w_{y_i} x_i - b_{y_i} + \Delta(y, y_i))]^2. \quad (\text{B.7})$$

Ranking (RNK)

$$L_{\text{RNK}}(w, b, x, y) = \sum_{i=1}^n \sum_{y=1}^C \max(0, w_y x_i + b_y - w_{y_i} x_i - b_{y_i} + \Delta(y, y_i)). \quad (\text{B.8})$$

Weighted Average Ranking (WAR)

$$L_{\text{WAR}}(w, b, x, y) = \sum_{i=1}^n \sum_{y=1}^C \sum_{j=1}^C \frac{1}{j} \frac{\max(0, w_y x_i + b_y - w_{y_i} x_i - b_{y_i} + \Delta(y, y_i))}{\sum_{c=1}^C \mathbb{1}[w_c x_i + b_c + \Delta(y, c) \leq w_y x_i + b_y]}. \quad (\text{B.9})$$

Multinomial Logistic Loss (LOG)

$$L_{\text{LOG}}(w, b, x, y) = \sum_{i=1}^n -w_{y_i} x_i - b_{y_i} + \log \sum_{y=1}^C \exp(w_y x_i + b_y). \quad (\text{B.10})$$

Entropy Regularized Hinge Loss (STF)

$$L_{\text{STF}}(w, b, x, y) = \sum_{i=1}^n -w_{y_i} x_i - b_{y_i} + \frac{1}{\beta} \log \sum_{y=1}^C \exp(\beta(w_y x_i + b_y + \Delta(y, y_i))). \quad (\text{B.11})$$

B.1.2.1 Averaging

To improve the convergence rate and consistency of the descent, two averaging schemes are implemented. It is also possible to not average at all.

Optimal averaging. For $\rho \in (0, 1]$, it keeps only the ρT last weights, where T is the total number of iterations.

$$\bar{w} = \frac{1}{\rho T} \sum_{t=\lceil(1-\rho)T\rceil}^T w_t. \quad (\text{B.12})$$

Note that $\rho = 1$ is exactly the standard averaging SGD.

Weighted averaging. Weighted averaging is conducted over all iterations, but gives a higher influence to the last weights:

$$\bar{w} = \frac{2}{T(T+1)} \sum_{t=1}^T t w_t. \quad (\text{B.13})$$

Auto-determination of the step size. The main issue with SGD is to determine the step size η_0 of the descent. To do so, we use a trick due to Leon Bottou [Bottou, 2012]. If η_0 is input as 0, JSgd starts by running a few iterations with various possible step sizes, and keeps the one that gives the *highest decrease in objective*. This heuristic usually leads to good step sizes.

Bias post-processing. There are two parameters optimized during the algorithm: the weight w and the bias b . There is no guarantee for a step size η_0 that would fit w to also give a good descent rate for b , which usually leads to suboptimality in the latter. To remedy to this problem, we use the following heuristic. We select a step size as usual, and normally run the SGD procedure. We then find b that minimizes the loss, where w is fixed, equal to the previous result. This can be done exactly in the OVR case, or with a new SGD for the others. If verbose mode is set, the objective before and after this post-processing is displayed, usually with significant ameliorations. This option can of course be deactivated.

B.2 ITP

Posterior to the publication of the conference article from which Chapter 2 stems, all the code that was used for the experiments was released, including:

Image transformation snippets. The package includes code to generate all transformations that were described in the conference article, including, crops, flips, homographies and colorimetric changes. The exact parameters that were used for our experiments can be used to reproduce results.

Classification software. The code features a binding to the previously described JSgd software that allows to adapt general purpose multi-class classification to classification with virtual examples. In particular, it proposes all schemes to aggregate scores as described in the related chapter.

Image Transformation Pursuit Algorithm. This is the main part of the package. It contains all variants of our proposed ITP algorithm, with all exposed

losses, as well as virtual examples usage variants. It is optimized to work with small features in an uncompressed way, and large features with Product Quantization (PQ) encoding.

Complete documentation and sources. The code is commented and heavily documented to support extensions and allow for easy understanding.

A website with installation procedures. The website to download the code is accessible at <http://lear.inrialpes.fr/people/paulin/projects/ITP/>. It contains detailed installation guidelines, toy features to test the software, and an FAQ for addressing common problems.

B.3 Patch-CKN

As with the previous conference article, material consisting of sources and data was released for the content of Chapter 3

The website is found at <http://lear.inrialpes.fr/people/paulin/projects/RomePatches/> and it includes:

The complete RomePatches dataset. We make publicly available for download the dataset which we created to benchmark our descriptors for both patch and image retrieval tasks. The actual patches can be directly downloaded. Unfortunately we could not make available the entire images due to copyright issues, and therefore only link to the urls. The train/test split is explicit in every case.

Precomputed descriptors. The descriptors (both CKN and SIFT) used in the article's experiments are available. They are able to reproduce the results that we reported in the corresponding chapter.

CKN generation code. The provided code is able to adapt the toolbox available on J. Mairal's website to our settings. In particular it makes use of the change of variable that we introduced in Section 3.4. The exact steps to download and modify the code are detailed in a comprehensive tutorial.

Installation guide Included is a quick tutorial guide to install and run test examples on custom data.

Bibliography

- Y. S. Abu-Mostafa. Hints. *Neural Computation*, 1995.
- P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.
- Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-Embedding for Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.
- G. An. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computations*, 1996.
- R. Arandjelovic and A. Zisserman. All about VLAD. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- A. Babenko and V. Lempitsky. Aggregating deep convolutional features for image retrieval. In *International Conference on Computer Vision*, 2015.
- A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *European Conference on Computer Vision*, 2014.
- P. Baldi and P. J. Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems*, 2013.
- H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, 2006.
- Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2009.
- D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

- T. Binford and T. Levitt. Quasi-invariants : theory and exploitation. In *Proceedings of Darpa Image Understanding Workshop*, 1993.
- C. Bishop. Training with noise is equivalent to Tikhonov regularization. In *Neural computation*, 1995a.
- C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford UP, 1995b.
- L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *Advances in Neural Information Processing Systems*, 2010.
- L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*. Springer, 2012.
- L. Bottou and O. Bousquet. The tradeoffs of large-scale learning. In *Advances in Neural Information Processing Systems*, 2007.
- M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.
- J. B. Burns, R. S. Weiss, and E. M. Riseman. View Variation of Point-Set and Line-Segment Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.
- M. Calonder, V. Lepetit, C. Strecha, and Fua. BRIEF: Binary robust independent elementary features. In *European Conference on Computer Vision*, 2010.
- T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. Pcanet: A simple deep learning baseline for image classification? *arXiv preprint arXiv:1404.3606*, 2014.
- K. Chatfield and A. Zisserman. Visor: Towards on-the-fly large-scale object category retrieval. In *Asian Conference on Computer Vision*, Lecture Notes in Computer Science. Springer, 2012.
- M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *International Conference on Machine Learning*, 2012.
- S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
- R. G. Cinbis, J. Verbeek, and C. Schmid. Multi-fold mil training for weakly supervised object localization. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014.

- S. Clinchant, G. Csurka, F. Perronnin, and J.-M. Renders. XRCE's participation to Imageval. *ImageEval workshop at CVIR*, 2007.
- A. Coates and A. Y. Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*. Springer, 2012.
- F. Cucker and D.-X. Zhou. *Learning theory : an approximation theory viewpoint*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, New York, 2007.
- D. DeCoste and M. Burl. Distortion-invariant recognition via jittered queries. In *International Conference on Computer Vision and Pattern Recognition*, 2000.
- D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 2002.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.
- J. Dong and S. Soatto. Domain-size pooling in local descriptors: Dsp-sift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in Neural Information Processing Systems*, 2014.
- D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.
- D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 2010.
- P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with Convolutional Neural Networks: a comparison to SIFT. arXiv Preprint, 2014.

- M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 1973.
- L. Florack, B. t. Haar Romeny, J. Koenderink, and M. Viergever. General intensity transformations and second order invariants. In *Theory & Applications of Image Analysis: Selected Papers from the 7th Scandinavian Conference on Image Analysis*. World Scientific, 1992.
- W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *Transactions on Pattern Analysis & Machine Intelligence*, 1991.
- E. Gavves, B. Fernando, C. G. M. Snoek, A. W. M. Smeulders, and T. Tuytelaars. Fine-grained categorization by alignments. In *International Conference on Computer Vision*, 2013.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, 2014.
- R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised feature learning from temporal data. In *Advances in Neural Information Processing Systems*, 2014.
- R. Goroshin, M. Mathieu, and Y. LeCun. learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems*, 2015.
- P.-H. Gosselin, N. Murray, H. Jégou, and F. Perronnin. Revisiting the Fisher vector for fine-grained classification. *Pattern Recognition Letters*, 2014.
- C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, 1988.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- C. Huang, S. Zhu, and K. Yu. Large scale strongly supervised ensemble metric learning, with applications to face verification and retrieval. Technical report, arXiv, 2012.

- M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015.
- D. Jayaraman and K. Grauman. Learning image representations equivariant to ego-motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- H. Jégou and O. Chum. Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening. In *European Conference on Computer Vision*, 2012.
- H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, 2008.
- H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
- H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.
- H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.
- Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM multimedia*, 2014.
- W. Jiang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *Proceedings of the European Conference on Computer Vision*. Springer, 2004.
- D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

- J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE International Conference on Computer Vision Workshops*, 2013.
- A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.
- M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, 2010.
- C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2013.
- Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, 1989.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proc. of the IEEE*, 1998.
- Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *European Conference on Computer Vision*, 2010.
- T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *International Conference on Computer Vision*, 2015.
- T. Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 1998.
- J. Long, N. Zhang, and T. Darrell. Do Convnets learn correspondances? In *Advances in Neural Information Processing Systems*, 2014.
- G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*. MIT Press, 2007.
- D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Ieee, 1999.

- D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 2004.
- L. Maaten, M. Chen, S. Tyree, and K. Q. Weinberger. Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 410–418, 2013.
- J. Mairal, F. Bach, and J. Ponce. *Sparse Modeling for Image and Vision Processing*. Foundations and Trends in Computer Graphics and Vision. now publishers, Dec. 2014a.
- J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, 2014b.
- S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.
- S. Mallat. *A wavelet tour of signal processing (3rd ed.)*. Academic Press, 2008.
- J. Marín, D. Vázquez, D. Gerónimo, and A. López. Learning appearance in virtual scenarios for pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
- K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*. Springer, 2002.
- K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal on Computer Vision*, 2004.
- K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.
- K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal on Computer Vision*, 2005.
- J. Y. H. Ng, F. Yang, and L. S. Davis. Exploiting Local Features from Deep Networks for Image Retrieval. In *Deep Vision Workshop (CVPRW)*, 2015.
- D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.
- P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, 1998.

- S. Nowozin, P. V. Gehler, J. Jancsary, and C. H. Lampert. *Advanced Structured Prediction*. Neural Information Processing series, 2015.
- M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, and C. Schmid. Convolutional patch representations for image retrieval: an unsupervised approach. *International Journal of Computer Vision*, 2016.
- M. Perd'och, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- F. Perronnin and D. Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3743–3752, 2015.
- F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
- J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
- J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *European Conference on Computer Vision*, 2010.
- L. Pishchulin, A. Jain, C. Wojek, M. Andriluka, T. Thormählen, and B. Schiele. Learning people detection models from few training samples. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *preprint arXiv:1403.6382*, 2014.
- J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 2012.

- J. Sánchez, F. Perronnin, T. Mensink, and J. J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal on Computer Vision*, 2013.
- B. Schölkopf and A. J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.
- S. Shalev-Shwartz and N. Srebro. Svm optimization: inverse dependence on training set size. In *International conference on Machine learning*, 2008.
- J. Sietsma and R. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 1991a.
- J. Sietsma and R. J. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 1991b.
- P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems*, 1992.
- P. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, 2003.
- E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *International Conference on Computer Vision*, 2015.
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ArXiv preprint ArXiv:1409.1556*, 2014.
- K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.
- J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, 2003.
- K. Sohn and H. Lee. Learning invariant representations with local transformations. *International Conference on Machine Learning*, 2012.
- R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.
- E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

- G. Tolias, Y. Avrithis, and H. Jégou. To aggregate or not to aggregate: selective match kernels for image search. In *International Conference on Computer Vision*, 2013.
- G. Tolias, R. Sivic, and H. Jégou. Particular Object Retrieval with Integral Max-Pooling of CNN Activations. In *International Conference on Representation Learning*, 2015.
- T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 2007.
- T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 2008.
- V. N. Vapnik. *Statistical Learning Theory*. Wiley Interscience, 1998.
- A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.
- P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, 2008.
- C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The CUB-200-2011 Dataset. Technical report, CalTech, 2011.
- L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, 2013.
- Z. Wang, B. Fan, and F. Wu. Local intensity order pattern for feature description. In *International Conference on Computer Vision*, 2011.
- C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, 2001.
- S. Winder, G. Hua, and M. Brown. Picking the best Daisy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). <http://cs.unc.edu/~ccwu/siftgpu>, 2007.
- L. Yaeger, R. Lyon, and B. Webb. Effective training of a neural network character classifier for word recognition. In *Advances in Neural Image Processing Systems*, 1996.

- J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014.
- S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. *preprint arXiv:1409.4326*, 2014.