Spatially Consistent Nearest Neighbor Representations for Fine-Grained Classification

Valentin Leveau

Collège Doctoral du Languedoc-Roussillon

THÈSE

Pour obtenir le grade de

Docteur

Délivré par l'**Université de Montpellier**

Préparée au sein de l'école doctorale **IS2***
Et de l'unité de recherche **UMR 5506**

Spécialité: **Informatique**

Présentée par **Valentin Leveau**
valentin.leveau@inria.fr

Représentations d'images basées sur un principe de voisins partagés pour la classification fine

Soutenue le 09/11/2016 devant le jury composé de

M. Michel CRUCIANU	Professeur	CNAM	Rapporteur
M. Benoit HUET	Maître de conférences	EURECOM	Rapporteur
M. Christophe GARCIA	Professeur des universités	INSA LYON	Examinateur
M. William PUECH	Professeur des universités	Université de Montpellier	Examinateur
M. Alexis JOLY	Chargé de recherche	INRIA	Co directeur
M. Olivier BUISSON	Docteur	INA	Co directeur
M. Patrick VALDURIEZ	Directeur de recherche	INRIA	Directeur

This thesis focuses on the issue of *fine-grained classification*, which is a particular classification task where classes may be visually distinguishable only from subtle localized details, and where background often acts as a source of noise. This work is mainly motivated by the need to devise finer image representations to address such fine-grained classification tasks by encoding enough localized discriminant information such as spatial arrangement of local features.

To this aim, the main research line we investigate in this work relies on spatially localized similarities between images computed thanks to efficient approximate nearest neighbor search techniques and localized parametric geometry. The main originality of our approach is to embed such spatially consistent localized similarities into a high-dimensional global image representation that preserves the spatial arrangement of the fine-grained visual patterns (contrary to traditional encoding methods such as BoW, Fisher or VLAD Vectors). In a nutshell, this is done by considering all raw patches of the training set as a large visual vocabulary and by explicitly encoding their similarity to the query image. In more details:

— The first contribution proposed in this work is a classification scheme based on a spatially consistent k-nn classifier that relies on pooling similarity scores between local features of the query and those of the similar retrieved images in the vocabulary set.

— Then, the main contribution of this work is a new aggregation-based explicit embedding derived from a newly introduced match kernel based on shared nearest neighbors of localized feature vectors combined with local geometric constraints. The originality of this new similarity-based representation space is that it directly integrates spatially localized geometric information in the aggregation process.

— Finally, as a third contribution, we proposed a strategy to drastically reduce, by up to two orders of magnitude, the high-dimensionality of the previously introduced over-complete image representation while still providing competitive image classification performance.

We validated our approaches by conducting a series of experiments on several classification tasks involving rigid objects such as *FlickrsLogos32* or *Vehicles29* but also on tasks involving finer visual knowledge such as *FGVC-Aircrafts*, *Oxford-Flower102* or *CUB-Birds200*. We also demonstrated significant results on fine-grained audio classification tasks such as the *LifeCLEF* 2015 bird species identification challenge by proposing a temporal extension of our image representation. Finally, we notably showed that our dimensionality reduction technique used on top of our representation resulted in highly interpretable visual vocabulary composed of the most representative image regions for different visual concepts of the training base.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Contents

1.1 Motivations

1.1.1 The data sharing era

Past decades have been devoted to giving people the ability to produce and share contents at a very large scale thanks to the advent of the Internet and mobile networks. This raised up to a fundamental breakthrough in our lifestyle, impacting both the whole social structure and the way we build, access and update human knowledge. This probably comes from a primitive need when humans had to learn to share their tasks and knowledge while defending themselves and understanding their environment. Sharing experience, information and data allows human beings to enhance their cleverness through this common consciousness so that the *global model* gets enhanced. As well noticed by Michel Serres in the conference he gave for the 40 years of INRIA [1], the creation and sharing of data that we experience today is certainly not a revolution but an accelerating evolution of some process that was initiated a very long time. Indeed, we have been learning to share data since the use of language which was the first way of sharing experience. Then we learnt to write symbols and documents, which were the first way for storing knowledge. This allowed us to spend less time memorizing and more time reasoning. This is

1. https://www.youtube.com/watch?v=ZCBB0QEmT5g

what mainly differentiates us from other species. We have externalized our memory to share it more easily and relay it to the next generation. Then the invention of printing was the first shape of Internet that allowed us to scale up the production and sharing of data and formalize in a more rigorous way the structuring of the data as a knowledge base. We have progressively externalized our memory to build on top of it more complex functionalities. Thus, we have taken a first step towards setting up a very large scale knowledge base with the never-ending pooling of individual works and experiences. Everyone is now able to quickly contribute, optimize and update this common knowledge.

1.1.2 Towards understanding and structuring data

With the advent of digital and acquisition technology, the archiving and sharing of photos and videos make the available audio-visual content never stops growing. The use of classical database technologies, requiring a manual description, makes it impossible to maintain a complete description of such a high volume of data. Today's video platforms such as Facebook, Wikipedia or YouTube, where about 2,000 hours of video are uploaded every day, make use of metadata generated by the users themselves but this is still insufficient to well integrate and index all content. This is also the main issue of most archive centers that are in charge of maintaining and enriching large scale audio-visual knowledge. In particular, INA, the industrial partner of this thesis, is a French public institute that currently manages approximately 15 million hours of TV and radio, and more than 1.5 million photographs. It archives $5,000$ new hours of video every year. Its main mission is to ensure the collection, archiving, conservation, restoring, enrichment and selling of all audio-visual content considered as being part of the French audio-visual culture. It integrates different media such as TV channels, radio streams, music videos, commercials and professional archives provided by audio-visual professionals (private or public) such as journalists.

Every day, hundreds of professional archivists working at INA are in charge of manually annotating that content. The manual annotation of such volumes of documents mainly relies on the indexing of information like authors, topics, places and other semantic information from a thesaurus composed of hundreds of thousands of entities associated to representative images of programs. The development of automatic and/or semi-automatic analysis tools to assist the annotation work of internal users (archivists, ...) and external users (general public, audio-visual professionals, ...) is therefore of great interest for companies like INA and concerns a wide range of multimedia applications such as:

- The automatic structuring of video streams such as TV news, political debates, entertainment programs, etc. ;
- The tracking of legal entities on French and international media ;
- The detection and tracking of media events ;

(a) Different instance of logos of the same entity

(b) Airplanes

FIGURE 1.1 – Some examples of Fine-Grained classification tasks.

— The large scale discovery of objects and topics in massive audio-visual databases.

In particular, the indexing, retrieval and classification of audio-visual materials based on semantic elements extracted from the contents (images, sounds, videos, notices) are of great interest in the context of INA where the archivists are highly interested in identifying visual objects that are part of the French (audio-visual) culture (such as monuments, logos, buildings, speakers, politicians and other public figures). Over the past decades, the problem of visual classification has been widely studied in the computer vision literature and many methods have demonstrated very good performances on several visual classification tasks, thus reducing more and more the gap between humans and machines. However, as powerful and impressive such non-specialist classification methods are, they are still not mature enough for lots of practical usages that require finer knowledge. This is notably the case for professional contexts, such as life sciences, marketing or media archiving where domain experts need to focus on very specific visual concepts involving complex and sometimes confusing visual patterns. The work presented in this thesis report focuses on such specialized image classification tasks that are source of an increasing interest in the computer vision and data management community.

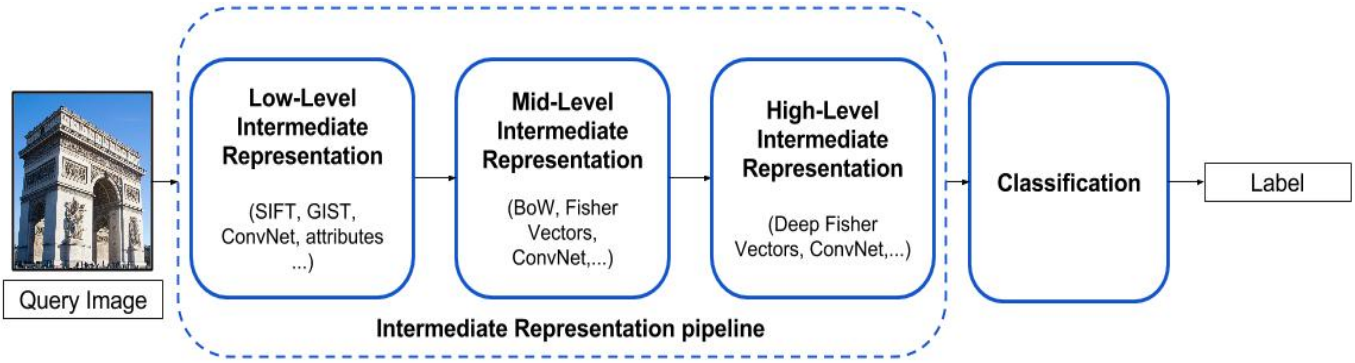


FIGURE 1.2 – Typical Classification pipeline.

1.2 Problem Statement

This work is focused on the research problem usually called Fine-Grained Object Classification [GMJP14] which corresponds to a particular classification task where classes may be visually distinguishable only based on subtle localized details (such as the number of windows between a *Boeing 747-100* and a *Boeing 747-200*) and where background highly acts as a source of noise. A special case is the so-called *instance classification* task: a class is a category defining a concept that can appear in different forms, and an object's instance corresponds to a particular occurrence of that object (*e.g.* a specific INA logo in Figure 1.1). The main challenge of Fine-Grained Classification is that it involves domain specific visual knowledge and the different instances are sometimes very difficult to differentiate as they belong to semantically similar visual concepts that share a lot of common features (such as different dog species, buildings, monuments etc.). As illustrated in Figure 1.1, each specialized classification task comes with its own specific constraints. Plants are for instance much more deformable objects than cars and might reach a high intra-class variability in terms of color and structure. Logos in natural scenes are very rigid objects but might involve extremely cluttered and changing background. Interesting patterns in medical images usually rely on subtle localized details in a rather fixed but cluttered background. Given this intrinsic ambiguity in the data, it is essential to build fine representations of objects and devise classification algorithms that encode these details.

As illustrated in Figure 1.2, a typical image classification system first consists in extracting relevant attributes, or features, describing the content of the images. Then, a classification algorithm aims at predicting the correct class to which the queried object corresponds. These features generally correspond to localized *low-level intermediate representations*, encoding some invariant properties of local image's regions. As we will see later in section 2.3, most classification schemes actually rely on an additional *mid-level intermediate representation* of the image between the low-

level feature extraction process and the classification. This intermediate processing block often consists in either learning in an unsupervised manner some *abstraction* of the low-level feature space before aggregating the local descriptors into a single vectorial representation, or directly learning a progressive non-linear embedding from the pixel space to the target space (as it is the case in Deep Learning methods).

As we will see later in the state-of-the-art chapter, most existing representations have the major disadvantage of losing the spatial and geometric information in the aggregation process. With these methods, two distinct objects can be represented by two very similar sets of descriptors and only the geometric information about the spatial arrangement of these descriptors might help to differentiate them. In other words, local invariants are not always sufficient to discriminate between classes. Thus, we also need to learn discriminant spatial configurations between the detected local patterns. Moreover, although state-of-the-art image representations are based on local descriptors, they are not able to consider small objects. Our major contribution is a new image representation based on a similarity-based embedding method that takes into account fine spatially localized geometric information by directly integrating it in the aggregation process.

1.3 Contributions

In this work, we investigate the use of localized geometric information to devise discriminant fine-grained classification models from large-scale image databases.

The first contribution is a classification scheme based on a spatially consistent k-nn classifier that relies on pooling similarity scores between local features of the query and those of the similar retrieved images in the training set. Practically, an exhaustive exact computation of all these similarity scores would require intractable computation time. Thus, our contribution consists in leveraging the scalability issue involved in the computation of the similarity scores based on an approximate k-nn search technique allowing relaxing the precision/computation time trade-off. We tested our approach on several international benchmarks involving brands in logos images (FlickersLogos32 [RPLvZ11], Belgalogos [JB09]), car models (Vehicles29 [KPFJ14]) as well as a large scale corpus in place at INA corresponding to over 300,000 images representing 5,000 classes of legal entities crawled on the web. Although we did show that our approach outperforms state-of-the-art methods on these classification tasks, notably for the recognition of rigid planar objects such as brand logos, the classification performance is significantly decreased when considering more complex classes of visual objects such as models of cars. This is mainly due to the fact that our matching-based classification scheme does not learn to combine the visual patterns so as to well differentiate the similar classes.

To alleviate this problem, the second contribution is a new image representation relying on the spatial pooling of geometrically consistent visual matches. This representation corresponds to the explicit embedding of a newly introduced match kernel based on shared nearest neighbors of localized feature vectors combined with local geometric constraints. We proposed a first naive solution to build such a representation whose high dimensionality (equal to the number of feature vectors in the training dataset) causes serious overfitting effect in practice and high computational and memory costs. To avoid overfitting and to reduce processing costs, the dimensionality of the resulting over-complete representation is further reduced by pooling the raw consistent matches according to their spatial position in the training images. Learning from these so-called *Shared Nearest Neighbors (SNN)* representations using a logistic regression classifier is shown to provide excellent fine-grained classification performance outperforming the results reported in the literature on several classification tasks.

As a third contribution, we propose to further reduce the high-dimensionality of the previously introduced Shared Nearest Neighbors representation using a recursive feature elimination method. To this end, we introduce a new recursive visual patch selection technique built on top of SNN representation and we show that the number of spatial atoms of the representation can be reduced by up to two orders of magnitude without significantly degrading the encoded information. The resulting representations are shown to provide competitive image classification performance compared to the state-of-the-art methods while being able to learn highly interpretable visual models composed of the most representative image regions for different visual concepts of the training base.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

State-of-the-Art: In chapter 2, we focus on the main algorithms designed to produce effective image representation suited for image classification. We first introduce the general framework of image classification and, in particular, we see why a good representation is a key point when we want to design good classification algorithms. We then review the most popular representation learning algorithms that have been considered in the literature and present related state-of-the-art methods applied for image classification. These methods include coding schemes and aggregation methods, Kernel methods and particularly match kernels that are the closest framework to which our own works are related. We also cover Deep Learning based representations that have been recently popularized (since 2012) by their impressive performance on image classification but also on a wide range of other computer vision tasks. Finally, we discuss these methods by positioning them with respect to

our own works.

Contributions: Chapters 3 to 6 concern the three main previously discussed contributions. We begin in Chapter 3 by the spatially consistent matching-based k-nn classification model based on approximate nearest neighbor search. Chapter 4 introduces the main contribution of this thesis, *i.e.* the *Shared Nearest Neighbor Match-Kernel* (SNN) which is the kernelization of the explicit representation that we use as input of classical supervised learning algorithm. In Chapter 5, we present further investigations that we led on the SNN representation. We notably study the behavior of our approach when using different underlying feature schemes such as handcrafted color descriptors or CNN features learnt in a supervised way. We also present a temporal extension of the SNN representation and we show that it exhibits good identification results on a problem of bird species identification from their songs. In Chapter 6, we present our method to reduce the memory cost of the SNN representation thanks to supervised feature elimination scheme. We notably show how this improves the interpretability of the learned representations and how this process can be cast as a supervised spatial dictionary learning problem where each dictionary entry is highly interpretable with respect to the visual knowledge the model has learnt.

Conclusion Finally, Chapter 7 is focused on the conclusions we can draw from this work by discussing more in detail about the strengths and weaknesses of our methods with respect to state-of-the-art approaches. We finally present potential future research direction with respect to our work.

Chapitre 2

State-of-the-art

Contents

Generalist and specialized image classification tasks are in essence not that different. Given an input image I, the aim is to find to which category it belongs, providing a training set of images of the targeted categories. As pointed in [CLZ13], the problem is however slightly different for fine-grained challenges that usually involve more cluttered backgrounds and a lower inter-class variability. Yet, the real challenge lies in the ambition to develop generic but adaptive classification methods as a foundation of a wide range of specialized applications involving complex and heterogeneous objects. As we have previously said, typical classification algorithms do not usually rely on the raw signal that present too many redundancies, noise and

very complicated structure, but rather on mid-level intermediate representations. This allows us to abstract some of the original signal by capturing interesting invariants and by eliminating the large amount of redundant and useless information for the task to solve. In this chapter, we first introduce the general framework of image classification and we see why a good data representation is a key point when we want to design good classification algorithms. Then, we review the most popular representations that has been considered in the literature and present related state-of-the-art methods applied for visual object recognition. These methods include coding schemes, aggregation methods, kernel methods and, particularly, match kernels which is a category of methods our own works are most related to. We also cover deep learning based representations that have been recently popularized by their impressive performance on image classification but also on a wide range of other computer vision tasks. Finally, we will discuss these methods and investigate their main strengths and weaknesses in the scope of fine-grained classification so as to introduce the main contribution of this work.

2.1 From pixels to image representation

2.1.1 Raw Image Classification and related issues

The basic formulation of image classification is to consider each image as a vector $\mathbf{x_i} \in R^d$ lying on a d-dimensional space where d corresponds to the number of pixels in the image (each pixel being considered as a variable of the problem to be solved). In the context of binary linear classification, each image of the training set is associated to a discrete variable $y_i \in \{-1, 1\}$ corresponding to its label, *i.e.* the class it belongs to. The training phase consists in finding a linear function $f(\mathbf{x})$ that correctly maps each input vectors to its correct label, *i.e.* $f(\mathbf{x_i}) = y_i$. This amounts to finding an optimal separating hyper-plane with a normal vector $\mathbf{w} \in R^d$ and bias b and the associated decision function of the classifier is defined as:

$$f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b) \tag{2.1}$$

Another way to consider the problem of image classification is to assume that images of the same class form some clusters in the raw high-dimensional pixel space (*i.e.* the space of pixel values). Therefore, a naive classification algorithm would simply consist in affecting the label of the nearest neighbor in that space with respect to a particular distance metric. Unfortunately, the raw pixel-wise representation of images is not well suited for such classification. Image classes actually correspond to highly complex and structured distributions such that they are not linearly separable or grouped in proper clusters. Moreover, distance metrics that are used to evaluate similarities between vectors lying in that space are generally sensitive to the dimensionality of the considered representation space, *i.e.* to the so-called *curse of dimensionality* which is itself highly dependent on particular variations in the input distribution.

2.1.2 Invariance and curse of dimensionality

A core aspect of the curse of dimensionality arises from the fact that, as the dimension of the space increases, the volumes in that space increase exponentially faster, and available data become very sparse. This produces the undesirable effect that the Euclidian distances become irrelevant, *i.e.* with less discriminative power. More formally, the range of possible distance values gets very narrow between the min distance and the max distance and tends to zeros when the dimension of the space tends to high values. This property is also referred in the literature as the *distance contrast*, or *relative contrast* problem [VF05, AHK01]. It implies that in high-dimensional spaces, there will never be enough samples to correctly tile the space and, therefore, there is no relevant neighbors. One way to solve the problem would be to find irrelevant features and to eliminate them. This procedure is known as *dimensionality reduction* which can be seen as a linear projection from the input space onto a lower dimensional space. However, as we have said before, real world data is in practice not generated by a simple distribution involving linear correlations between variables. Projecting the data into such lower dimensional linear subspace would thus be a highly destructive process where we would lose a great part of the useful information contained in the data.

Actually, such *contrast distance* phenomenon involved in the *curse of dimensionality* hopefully becomes less true when considering more realistic data spaces where data generally follow mixtures of distributions. Indeed, it has been shown [HKK+10] that the discriminating power of Euclidian distances in high-dimensional spaces actually increases with the number of *relevant* attributes in the representation. Therefore, it turns out that increasing the dimensionality actually does not degrade the discriminating power if the dimensionality is increased with relevant variables with respect to the task to solve. On the contrary, if the dimensionality is increased while decreasing the number of relevant attributes, then the curse of dimensionality fully applies. Therefore, the curse of dimensionality is not only due to the raw number of space dimensions, it is also related to the underlying structure of the space where the data is embedded. As real world data generally lie on structured spaces, dealing with the curse of dimensionality now turns into projecting the data into a more informative space such that simple distance metrics become more relevant in the transformed space. Ideally, we would like each dimension of this space to correspond to a given independent factor that *explains* a particular characteristic of the data (*i.e.* its pose parameters or the degree of activation of a particular visual pattern). This phenomenon is often denoted as the *intrinsic dimensionality* also known as the so-called *manifold hypothesis* of the data.

The manifold hypothesis has been highly studied in the literature, particularly in the scope of unsupervised representation learning (see Bengio et al. [BCV13] for a survey of a wide range of advances and perspectives in representation learning).

The idea behind the manifold hypothesis is that data, represented in a given high-dimensional space, actually lies on a lower dimensional subspace that can be seen as a manifold embedded in the original d-dimensional space. In other words, existing data should have some statistical structure, and only a restricted number of small displacement in R^d are allowed to stay on the manifold. A kind of proof behind this idea is that a very small number of configurations in the original space (say the pixel space for instance) correspond to existing data. Figure 2.1 is an illustration of a simplified 2-dimensional non-linear manifold where the data lie, following a certain distribution in their original 3-dimensional space. Intuitively, if such manifold would exist, moving on this non-linear hyper-surface in a particular direction would cause particular transformations of the image signal (such as rotation of the image patch in Figure 2.1) leading to another existing data. On the contrary, moving on directions driving data away from the manifold would correspond to highly unprobable transformations leading to meaningless and unprobable data. In this perspective, the goal for representation learning would be to learn from a training dataset the natural authorized directions of variation in such high-dimensional spaces to stay on the manifold.

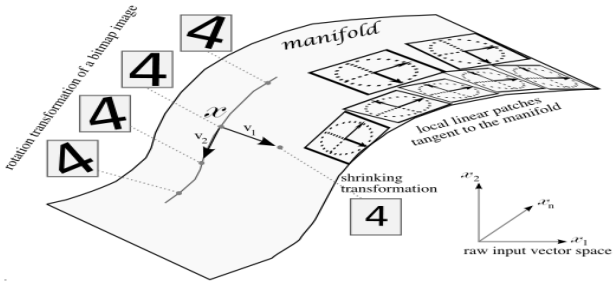


FIGURE 2.1 – Illustration of a 2-D non linear manifold lying in a 3-D space. Taken from Bengio et al. [Ben09].

This would actually correspond to finding, or learn, the parametrization of such manifold and moving along the surface in the original space would be equivalent to move linearly in the space obtained from the change of variable $\Phi()$ associated to this parametrization. The main challenge of image classification then consists in finding such clever mapping $\Phi()$, that captures the structure and useful invariances in the data so as to linearize these complex and structured distribution. Doing so is likely to produce a representation space where the embedded input samples will be easily handled by simple machine learning algorithms giving rise to object classification systems with high accuracy. In practice, such manifolds are prone to correspond to much more complex non-linear parametrization than the simplified one illustrated in Figure 2.1 and finding such explanatory factors is still a very challenging issue. This is particularly true for fine-grained classification tasks where real world image data

generally has highly complex distributions involving many variations and transformations such as: spatially localized invariants, redundancies, spatial configuration of localized patterns, corruption processes such as viewpoint changes, lightning changes or occlusions.

As we have said in the previous chapter when commenting Figure 1.2, most image classification methods considered over the past years make use of such intermediate data representations as entry of traditional machine learning algorithms. We often denote these intermediate representations as *features*, or *feature vectors*. There is two main ways of designing such features *i)* they can be manually designed if we have enough prior information about the relevant statistic of the data, we then denote these features as *engineered features* or *handcrafted features* or *ii)* they can be learned from the raw signal (or on top of generic lower-level handcrafted features as it is often the case) if we don't have such prior or if we do not want to spend time and effort to design them manually. Although learning the features seems more attracting at a first glance, the major difficulty that arises is to design good learning algorithms to discover useful invariants in the data. In the following sections, we are going to review the most common image representations applied in the scope of image classification. We will first review handcrafted intermediate representations in section 2.2. Following sections will be focused on supervised and unsupervised representation learning algorithms that can be used either on top of such handcrafted features or directly on top of raw image patches.

2.2 Handcrafted image representations

The first way of defeating the curse of dimensionality that has been considered in the literature is to explicitly *hard-code* image invariants. This is done by designing engineered feature extraction schemes producing vectors of *relevant* low-level features that well summarize the visual content of the image. In this section, we are going to present a brief overview of such handcrafted feature schemes that have been widely used in traditional computer vision methods designed over last decades. Images can be represented by two main kind of features: global features and local features.

2.2.1 Global Features

Global features simply consist in directly representing an image $\mathcal{I}_x$ into a single feature vector $\Phi(\mathcal{I}_x) \in R^D$ where $\Phi()$ is an operator, or an algorithm, that embeds the image in a D-dimensional space. This operator generally corresponds to an *image processing* algorithm that analyzes the visual content of the image and builds a vector representation whose each component indicates how much a particular visual attribute occurs in the image. The value of each attribute activation is often

computed in a deterministic way and each attribute is associated to a particular hand-engineered detector. The obtained global representation can be then considered as an *abstraction* of the original image because it only keeps a reduced part of the original visual information from which we hope we can distinguish the different classes represented in the images. Ideally, we would like vectors representing a particular class in this new representation space to be well separated from those of other classes, so that simple machine learning methods (such as linear classifiers or nearest neighbors) perform well on top of this representation space. The main challenge for designing such handcrafted global representations relies on the expertise we should have on the task to solve so as to build informative and discriminative features. This presupposes that we have prior information about what kind of visual patterns are useful for this task.

Over the past decades, a lot of research works has been focused on finding such useful global representations for images. Most of these methods often rely on summarizing an image with histograms of different visual attributes such as textures or color information. One of the pioneering work for global color histogram representations is the one of Swain et al. [SB91] where each pixel is described by a uniformly quantized RGB (3-dimensional) color vector. The histogram representation simply consists in counting how much a discretized color occurs in the image. The obtained representation is normalized so as to be robust to lightning changes. Such representations were reported to have very good performance on scene recognition tasks and a lot of related works have been proposed in the computer vision literature such as [HKM+99, GS00, RT01] or [WDJ11] where the authors chose to discretize the color space with an adaptive clustering algorithm [1]. Other proposed global image representations rather consider encoding texture information such as methods relying on wavelets [WWFW98]. Other alternatives are histograms of oriented gradients such as the popular *GIST* representation [OT01] where the image is convolved with a bunch of Gabor filters at different orientations and scales producing as much feature maps as the number of Gabor filters. These feature maps are divided into 16 regions (a 4x4 regular grid) and the feature values of each region are aggregated with an average pooling. The 16 average values of all the feature maps are then concatenated producing a global vector representation that summarizes the gradient information for different parts of the image which is well suited for scene categorization.

Global feature representations have the main advantage to correspond to compact encodings summarizing the whole visual content of an image. This makes them well suited for linear classification algorithms provided that the encoded visual attributes are relevant with respect to the targeted task. However, the main drawback of such representations is that they do not capture localized information of objects in the image which would be a desired property in the context of fine-grained classi-

1. Like the K-means algorithm that we will see in section 2.3

fication. To overcome this drawback, the computer vision community highly focused its attention toward localized handcrafted feature extraction schemes.

2.2.2 Local features

Local feature vectors provide distinct information about specific location in the image. This makes them well suited to deal with common issues encountered in computer vision systems such as occlusions or translations. Many local feature schemes have been designed to be invariant under other transformations, such as rotation or scale, such that they are relatively stable when objects show up with different viewpoints. Local features are usually extracted in the neighboring region of *interest points*, which are specific positions in the image where we expect to find informative patterns. In the remainder of this section, we are not going to exhaustively review interest point detectors and local feature schemes of the literature since it corresponds to a domain that received a lot of attention in the last decades. We will only review the categories of methods related to our own work.

Interest point detectors : Interest points can be extracted using two main ways. The first and more naive one [GMJP14, CLVZ11, ITGJ15] consists in densely sampling image patches with a regular spatial grid (*e.g.* 16x16 image patches every 3 pixels in both horizontal and vertical direction) at several determined scales (*e.g.* considering scaled/resized version of the input image by a constant factor of $\sqrt{2}$). This leads to a fixed number N of extracted local features for each image at possibly multiple resolutions so that to deal with objects appearing in different positions and depth. Another way that has been widely investigated consists in evaluating image locations corresponding to highly informative visual patches. To perform this task, each image patch is passed through a function that returns a saliency score and only the visual patches having a saliency score higher than a determined threshold are kept for the feature extraction phase. These functions or algorithms allow the removal of visual patches likely to correspond to noise or uninformative content (with respect to some priors we have about the task). Popular choices of interest point detectors are the Harris detector [HS88], FAST [RD06], SURF [BTVG06], Harris-affine [MS02], Harris-Laplace [MS04a], Hessian-affine [PCM09], DoG [Low04], MSER [MCUP04]. More details are given in Iscen et al. [ITGJ15] that provides a comparative study between many popular interest point detectors as in Mikolajczyk et al. [MS05].

Local feature schemes: Once the interest points have been detected, vectors of features are computed on the corresponding image patches so as to describe their visual content. One of the mostly used category of local features is the one based on histograms of gradients such as the popular *Scale Invariant Feature Transform* (SIFT) [Low04] that has been extensively used in the computer vision community. SIFT descriptors was originally formulated to,somehow, mimic the kind of features

the brain actually uses to perform recognition tasks. These biology-inspired features are based on localized gradient information in the image (*e.g.* such as oriented edge or corner detectors), and we aim such descriptors to be invariant to scale and orientation changes. More concretely, the basic principle consists in extracting a histogram of gradients computed on image patches of size 16x16 pixels divided into 4x4 sub-regions of 4x4 pixels. For each sub-region, a histogram of gradients is computed by simply counting the number of gradient vectors whose quantized orientation (on 8 values) corresponds to the histogram entry. The resulting vectors is simply the concatenation of the 4x4 histograms with 8 entries leading to a 128-dimensional feature vector to describe the patch. To provide rotation invariance, local scale and dominant gradient orientation of the full patch are previously computed and the patch is oriented in the scale space before extracting the feature vector. SIFT have been successfully used in a lot of different computer vision tasks such as object retrieval [PCI+07, JPD+12, JB09], robotics/tracking and image classification schemes [EVGW+b, GHP07, DDS+09]. A lot of extensions have been proposed over the past decades such as PCA-SIFT [KS04], that simply consists in applying a Principal Component Analysis [Pea01] on the large dataset of pixels' vectors of patches of the gradient image used for SIFT. Affine SIFT [MY09] takes into account camera pose parameters so as to be more invariant to viewpoint changes. Another kind of local descriptors related to SIFT is the so-called *HOG (Histogram of Oriented Gradients)* descriptor [DT05] which consists in densely evaluating histograms of gradients of spatial regions of the image. Another popular scheme that was proposed by Mikolajczyk et al. [MS05] is the so-called *Gradient Location and Orientation Histogram (GLOH)*. It consists in defining more spatial regions to build the histograms by changing the grid used in SIFT to aggregate the local gradients. Actually, they use a log-polar location grid composed of 17 location bins and the gradients are quantized into 16 bins. This lead to 272-dimensional vectors compared to 128 for SIFT. The obtained high-dimensional descriptor is then compressed with PCA so as to recover the size of a SIFT descriptor while being more informative.

Local features have proved to be very successful in appearance based object matching and recognition, as they are robust to viewpoint changes, distortions, lightning or occlusions. In the context of image classification, we will see that many methods aggregate such local features into a global representation so as to use linear classification algorithm on top of it. In the following section, we review these *aggregation-based* methods in details as they have received a lot of attention over recent years in particular in the context of fine-grained image classification. Although handcrafted features have been intensively considered in computer vision, they become less and less attractive because of the upcoming of deep learning methods that perform an end-to-end learning directly from the pixel information. This leads to task-adapted feature schemes rather than handcrafted ones where we need to put a lot of effort to design informative feature space given the prior we have on the task. This highly desired property will be studied in details in section 2.4.

2.3 Aggregation-based methods

The category of methods we are going to review in this section consists in producing global image representations by aggregating previously extracted local features. More concretely, such methods generally rely on a three-stage pipeline: *i)* the feature extraction phase where a set $\mathcal{X} = \{\mathbf{x_i}\}_{i \leq N}$ of N local descriptors (such as those seen in the previous section) are first extracted from the images, *ii)* the encoding, or embedding phase where an unsupervised learning algorithm is used to capture the structure of the input space where the feature vectors lie, and finally *iii)* the aggregation phase where the resulting encoded descriptors are pooled into a global image representation that summarizes the *interesting* visual content of the image. The main strength of such methods relies on the use of an unsupervised learning algorithm which allows us to produce an *abstraction* of the input space rather than simply relying on the raw appearance-based similarities of localized regions of the images. In the remainder of this section, we are going to review the state-of-the-art aggregation methods. The most popular representation of this category is the so-called *Bag of Visual Word (BOVW).*

2.3.1 The Bag-of-Visual-Word representation

The Bag-of-Visual-Words (BoVW) method introduced by Sivic and Zisserman [SZ03], is a global image representation that has been widely used in several computer vision tasks such as object recognition, image retrieval and others. The Bag-of-Word paradigm has been originally designed in the context of text search engines. Text documents can actually be represented by word count vectors, which is explicitly possible because the vocabulary is defined by the language. It is obviously not as easy in the visual case since there is no explicitly defined vocabulary (descriptors correspond to vectors in R^d) and there are, hence, an infinity of possible words that could define the vocabulary. To re-discretize the problem, Sivic and Zisserman [SZ03] therefore proposed to create a visual dictionary which size is controlled. The creation of these words is often achieved through an unsupervised clustering algorithm which consists in quantizing the input space where local features are lying. This is the embedding phase of the Bag-of-Visual-Words aggregation method. In the remainder of the section, we are first going to describe the basic unsupervised algorithm that is used to embed the input descriptors in an adapted quantized space and we will then present the different extensions of the BoVW model that have been proposed in the literature.

Vector Quantization (VQ Coding)

Vector Quantization (VQ Coding) is a particular case of coding-based methods whose goal is to learn a set of basis vectors $\mathbf{d_z} \in R^d$ whose linear combination optimally reconstructs the input vectors $\mathbf{x_i} \in R^d$ (that can be either handcrafted

features such as SIFT or GIST or directly pixel patches). In other words, we aim at learning a set of latent variables that explain well the input distribution. We denote $\mathbf{D}$ the codebook, or dictionary, *i.e.* the dxk matrix whose column vectors $\mathbf{d_z}$ form an overcomplete basis (*i.e.* $k >> d$) that we aim to learn by minimizing the following objective term:

$$J(\mathbf{D}, \mathbf{H}) = \sum_i ||\mathbf{x_i} - \mathbf{D}\mathbf{h_i}||_2^2 \tag{2.2}$$

where $\mathbf{h_i} \in R^k$ denote the vector containing the k coefficients from which we can linearly combine the basis vectors $\mathbf{d_z}$ to reconstruct the original input vector. We often denote $\mathbf{h_i}$'s as the *codes* of the $\mathbf{x_i}$'s. In such coding methods, the learning procedure consists in jointly learning the optimal set of dictionary entries $\mathbf{d_z}$ and for each training sample, the optimal vector of coefficient $\mathbf{h_i}$ that minimizes the expected reconstruction error term given by Eq. 2.2. Once the codebook has been learned, what we aim to do is to perform feature extraction from any new input vector $\mathbf{x_i}$, *i.e.* to compute its representation $\mathbf{h_i}$. In the literature, this feature extraction process if often denoted as *inference*.

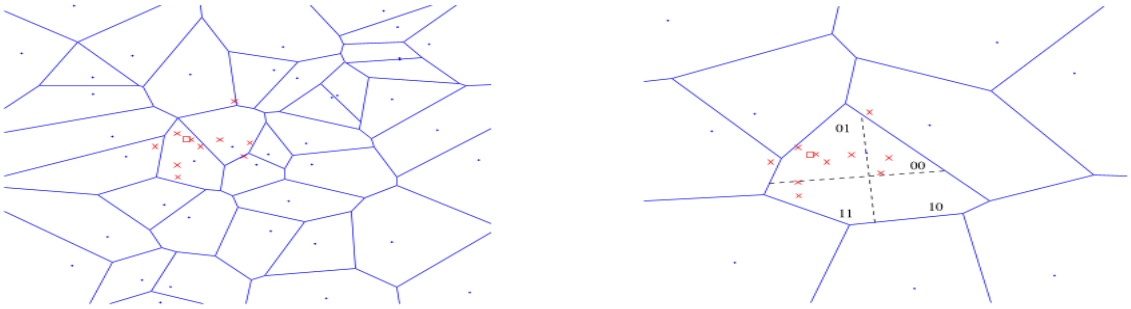
The particularity of VQ coding is that it imposes the codes $\mathbf{h_i}$ to have only one component h_{ij} to be different from zero. This kind of codes is often denoted as *one-hot* codes in the literature. More concretely, we aim at finding a function $Q : R^d \rightarrow R^d$ that maps any input vector $\mathbf{x_i} \in R^d$ to a vector $Q(\mathbf{x_i}) \in R^d$ belonging to a finite subset of k codebook vectors $\{\mathbf{c_j}\}_{j \leq k}$ which are called centroids. The centroids correspond to quantified versions of the original input vectors and the objective of vector quantization consists in finding the optimal sets of k centroids in terms of quantization error:

$$\frac{1}{N} \sum_i^N ||\mathbf{x_i} - Q(\mathbf{x_i})||_2^2 \tag{2.3}$$

The vectors $Q(\mathbf{x_i})$ correspond to the reconstruction values of the input vectors $\mathbf{x_i}$ and can actually be rewritten as $\mathbf{D}\mathbf{h_i}$ where $\mathbf{D}$ is the codebook matrix whose column vectors are the centroids $\mathbf{c_z}$ and the code $\mathbf{h_i}$ is a vector filled with zero except for the component h_{ij} with j is such that $\mathbf{c_j} = Q(\mathbf{x_i})$. This actually amounts to minimizing equation 2.2 subject to $||\mathbf{h_i}||_0 = ||\mathbf{h_i}||_1 = 1 \, , \mathbf{h_{ij}} = 1 \; with \; 1 \leq j \leq k$. Given that we have already learned the codebook, the embedding in the quantized space now corresponds to finding the codebook entry that minimizes its distance to the original input:

$$Q(\mathbf{x_i}) = \mathbf{D}\mathbf{h_i} = arg \min_j ||\mathbf{x_i} - \mathbf{c_j}||_2 \tag{2.4}$$

The algorithm that is mostly used to perform such tasks is the so-called *K-means* algorithm which is an unsupervised learning algorithm that learns a near optimal codebook for vector quantization. It can also be viewed as an unsupervised clustering algorithm that aims to partition the input space into k clusters, the codebook entries.



(a) $k = 40$ (b) $k = 10$

FIGURE 2.2 – Illustration of the VQ coding and Hamming Embedding principle. A visual vocabulary is first learned with traditional VQ coding algorithm. The encoding process of a feature vector consists in quantifying the residual vector between a feature point and its quantized value. A given input vector is then described by *i)* the produced short binary vectors and *ii)* its hard-assigned visual word. Figure taken from Jegou et al. [JDS08].

The region of the input space whose points belong to a particular cluster $\mathbf{C_j}$ is called a *Voronoi* cell denoted by $\mathcal{V}_j$. Each cluster $\mathbf{C_j}$ is represented by its centroid $\mathbf{c_j}$ which corresponds to the expectation of the points of the input space belonging to its associated *Voronoi* cell:

$$\mathbf{c_j} = \frac{1}{N_j} \sum_{\mathbf{x_i} \in C_j} \mathbf{x_i} \tag{2.5}$$

where N_j is the number of input points belonging to C_j. This is actually the optimal thing to do if we want to minimize Equation 2.3 considering it can be rewritten as:

$$E(\mathcal{X}, C) = \frac{1}{2} \sum_i \sum_j \gamma_{ij} ||\mathbf{x_i} - \mathbf{c_j}||_2^2 \tag{2.6}$$

where γ_{ij} correspond to a hard assignment term such that

$$\gamma_{ij} = \begin{cases} 1 & \mathbf{c_j} = argmin_{\mathbf{c_{j'}}} ||\mathbf{x_i} - \mathbf{c_{j'}}||_2^2 \\ 0 & else \end{cases}$$

and minimizing the above term, given assignments are already known, amounts to equating to zero the partial derivatives of $E(\mathcal{X}, C)$ with respect to each cluster enter $\mathbf{c_j}$ which gives:

$$\frac{\partial E(\mathcal{X}, \mathbf{C})}{\partial \mathbf{c_j}} = \sum_i \gamma_{ij} (\mathbf{x_i} - \mathbf{c_j}) = \mathbf{0} \tag{2.7}$$

and so:

$$\mathbf{c_j} = \frac{\sum_i \gamma_{ij} \mathbf{x_i}}{\sum_i \gamma_{ij}} \tag{2.8}$$

where $\sum_i \gamma_{ij} = N_j$. The basic K-means algorithm can thus be directly derived from equation 2.8 and 2.4. The codebook is learned by iteratively assigning each input points to the closest centroid, that has been initialized randomly for the first iteration. Setting each centroid to the empirical mean of its assigned input points.

The Bag-of-Visual-Words representation

Once the visual vocabulary has been learned, images are then represented by a histogram vector whose components correspond to the number of occurrences of each visual word. More formally, we want to represent an image $\mathcal{I}_X$ represented by a set of local feature vectors $\mathcal{X} = \{\mathbf{x_i}\}_{i \leq |\mathcal{X}|}$. Each descriptor is quantized to its nearest cluster center and a global image representation is obtained by sum pooling the representations of all the feature vectors contained in the image:

$$\Phi_{BoVW}(\mathcal{X}) = \sum_i \mathbf{h_i} \tag{2.9}$$

where $\mathbf{h_i}$ is the one-hot vector corresponding the representation of local feature $\mathbf{x_i}$ such that:

$$Q(\mathbf{x_i}) = \mathbf{D}\mathbf{h_i} \tag{2.10}$$

As we can see, $\Phi_{BoVW}(\mathcal{X})$ actually corresponds to a word count vector as each component $\Phi^j_{BoVW}(\mathcal{X})$ is equal to $\sum_i \mathbf{h_i^j}$, *i.e.* the number of input points $\mathbf{x_i}$ that has been quantized to the j-th word of the dictionary.

Beyond Bag of Words

As it relies on vector quantization, the BoVW representation is affected by quantization errors, in particular in high-dimensional spaces where the *curse of dimensionality* occurs. Indeed, in such high-dimensional spaces, very similar visual features might be split across distinct clusters whereas more dissimilar ones might be affected to the same visual word. This results in both mismatches and potentially irrelevant matches. A lot of efforts have been done to deal with these drawbacks giving rise to several improvements of the original BoVW representation.

Normalization strategies A first improvement consists in devising efficient normalization schemes to deal with the fact that less frequent, and potentially more informative patterns, are highly penalized with respect to more frequent ones that are prone to be less informative because they appear anywhere in the dataset (*e.g.* irrelevant background, textures, etc.). The mostly used normalization scheme for BOVW is the popular *Text Frequency-Inverse Document Frequency* (TF-IDF) [SJ72, SZ03, JDS08] that was originally formulated in the context of text search engines. Intuitively, it consists in weighting down the BOVW components corresponding to frequent words and weighting up the components corresponding to rare words.

Soft Assignment strategies Another line of research consists in expanding the assignment of a given local feature to its nearest visual words. One of the pioneering work in this research direction is [VGGVS08, VGVSG10] where the inference step of VQ coding consists in *soft-assigning* each descriptors $\mathbf{x_i}$ to every code vector $\mathbf{c_j}$ as:

$$\gamma_{ij} = \frac{e^{\left(-\beta||\mathbf{x_i}-\mathbf{c_j}||_2^2\right)}}{\sum_k e^{\left(-\beta||\mathbf{x_i}-\mathbf{c_k}||_2^2\right)}} \tag{2.11}$$

producing a code vector of the form $\mathbf{h_i} = [\gamma_{i1}, \gamma_{i2}, ..., \gamma_{iK}]^T \in R^K$ where K is the number of codebook vectors. Other techniques such as [LWL11] rather propose to restrict the soft coding only to the nearest code vectors rather than the whole vocabulary. This method is denoted as *Soft Assignment* and has shown great improvement in both retrieval and classification applications.

Alternative Embeddings Although Vector Quantization has been extensively used in the past decades, it highly suffers from quantization errors and it is highly sensitive to the size k of the vocabulary [YJHN07, HKQ10, MPCM10]. Intuitively, if k is too low, some input samples that are inherently different fall into the same clusters. On the contrary, is k is too large, the selectivity of the representation is too high and similar points will be affected to different clusters. Thus, a lot of methods have been proposed to refine the quantization in a two-stage process (*i.e.* a coarse-to-fine quantizer). One of the most popular methods of the literature in this research direction is *Hamming Embedding* (HE) proposed by Jegou et al. [JDS08] whose principle is illustrated on Figure 2.2. The authors proposed to project the residual vectors of a coarse K-Mean quantizer into a Hamming binary space to increase the selectivity of the original features. More formally, each vector is assigned to a *coarse* K-Mean cluster and the residual vector with respect to this cluster is computed. The dimension of the vectors is first reduced by projecting them on the top-m eigen vectors learned with a PCA. Then a random rotation projection is applied to them to better balance the energy of the projected vector in the original feature space. Finally, each component of the resulting residual vector is binary quantized with respect to a per component median value computed on the whole dataset. Intuitively, balancing the energy allows us to spread the localization information of the vectors across the different components. This prevents from giving more importance to a particular component and reduce the quantization loss introduced by the binarization procedure. Actually, the first PCA stage is more about removing redundant information of the localization and the random rotation is more about providing more discriminant Hamming distances. The resulting binary code vectors are then normalized in a TF-IDF-like procedure. Later, Jegou et al. [JDS11] proposed an alternative improvement of the BoVW model by using a product quantizer rather than the original K-means quantizer. The input feature space is split into m non-overlapping subspaces and a quantizer is learned on each subspace rather than

learning only one quantizer on the full space. More formally, each input sample $\mathbf{x_i}$ is decomposed into m sub-vectors $\mathbf{u_j}(\mathbf{x_i})_{j \leq m}$ and a K-means quantizer $q_j()$ is learned on each set of sub-vectors $\mathbf{u_j}(\mathbf{x_i})_{i \leq N}$. The final representation of a given input sample $\mathbf{x_i}$ then corresponds to the concatenation of its quantized sub-vectors. As for HE, a coarse quantizer is first learned on the input vectors and their short binary (PQ) codes are then computed on residual vectors.

Geometry consistency in BoVW Another important research direction has been investigated to deal with the lack of geometry information in such representation. Indeed, the initial strength of local features is that they allow to deal with a lot of invariance problems such as viewpoint or occlusion. When aggregating such local features into BoVW representation we loose the spatial arrangement information of the local features which is prone to degrade the discrimination between two visual patches. This is why some approaches considered integrating some geometric information into the BoVW representations. For instance, partial geometry can be embedded in the image representation by using the Spatial Pyramid Matching scheme of [LSP06] that considers a multi-resolutional grid in the image. Another alternative that has been successfully applied on several classification tasks is the so-called *Spatial Coordinate Coding (SCC)* [KM11] method where each local descriptor is augmented with additional components corresponding to its location in the image. Geometry consistency schemes will be covered more in details in sections 2.5.2 and 3.3.

2.3.2 Sparse Coding based aggregation methods

Sparse Coding principle

Sparse Coding (SC) [OF97, LBRN06, MBPS09, GL10, AGMM15] is another kind of coding scheme that can be a good alternative to Vector Quantization because, as illustrated on Figure 2.3, we want the code vectors to have more than one non-zero component. This allows us to reconstruct the input samples by the linear combination of a few basis vectors of the dictionary which is highly prone to reduce the encoding error involved in VQ coding. A lot of works [YYGH09, ZGL+13, TLSJ12, ZWH+13, BBLP10, WP14, YYH10] demonstrated significant improvements over BoVW models by using Sparse Coding. Learning such sparse, and possibly overcomplete, representations allows us to *capture* meaningful information that can generate the input samples of the training set. To this aim, a sparsity regularization term $\lambda \sum_i R(\mathbf{h_i})$ is added to the objective function given by equation 2.2 so as to penalize the model to infer codes with too much non-zero components. Although L_0 norm criterion, *i.e.* $R(\mathbf{h_i}) = ||\mathbf{h_i}||_0 = \sum_j I(h_i^j \neq 0)$, seems to be the most relevant criterion for sparsity, this is the L_1 norm term, *i.e.* $R(\mathbf{h_i}) = ||\mathbf{h_i}||_1 = \sum_j |h_i^j|$, that is often used because the L_0 term is not a smooth function which makes the optimization-based inference not possible. The final objective function for sparse

coding now becomes:

$$J(\mathbf{D}, \mathbf{H}) = \sum_i ||\mathbf{x_i} - \mathbf{D}\mathbf{h_i}||_2^2 + \lambda \sum_j |\mathbf{h_i^j}| \tag{2.12}$$

Ideally, what we aim to do is to jointly optimize the reconstruction objective and the inference of sparse codes that minimize the average reconstruction error term (which can be seen as two objective functions that play against each other). Actually, the most representative methods of the literature rather consider alternating between *i)* the inference of the sparse codes knowing the dictionary $\mathbf{D}$ and *ii)* the optimization of the dictionary knowing the codes. As we have seen before in section 2.3.1 for VQ coding with the K-Means algorithm, this optimization strategy is very similar to the *EM* algorithm (that we will cover in section 2.3) where the E step corresponds to the inference part and the M step corresponds to the dictionary learning part.

Dictionary learning In the past decades, a lot of methods have been considered for the dictionary learning part of sparse coding algorithm. The naive way to do it would be to assume that the code vectors do not depend on the dictionary and to simply optimize the reconstruction part of the objective function with gradient-based optimization leading to the following update rule:

$$\mathbf{D} \leftarrow \mathbf{D} + \frac{2\rho}{N} \sum_i (\mathbf{x_i} - \mathbf{D}\mathbf{h_i})\mathbf{h_i}^T \tag{2.13}$$

where ρ is the learning rate and N is the size of the training set. The codebook vectors are then normalized to the unit ball after updating their values so that to avoid degenerate solutions where $\mathbf{D}$ would take high values while the code vectors become small to satisfy the sparsity constraint. This procedure is often denoted as projected gradient-descent [CZ97]. In [LBRN06], the authors rather chose to use a dual lagrangian formulation and showed that this method exhibits significant improvement in terms of convergence speed compared to such iterative projected gradient descent. Another very popular sparse coding method introduced by Mairal et al. [MBPS09] rather makes use of block-coordinate descent algorithm [Tse01, WL08, QSG13] which consists in optimizing equation 2.13 with respect to one column vector $\mathbf{d_j}$ keeping the other ones fixed and cycling this optimization over the column vectors. The particularity of such algorithm is that they generally do not need to specify a learning rate. In their paper, Mairal et al. [MBPS09] also propose an online and mini-batch [LBBH98, Bot10, LBOM12, Bot12] version of this block-coordinate descent dictionary learning (*i.e.* where we only need one, or a few, input samples to update the basis vectors). This makes the method much more scalable both in terms of memory and computation efficiency for large training datasets.

Inferring the sparse codes Contrary to the inference procedure in VQ coding, the sparse coding inference procedure corresponds to a more difficult optimization

problem where we want to find the linear combination of a reduced subset of codebook vectors that well reconstruct the input vector of the training set. The naive way to do it would be, as for the dictionary learning part, to assume that the codes do not depend on the dictionary $\mathbf{D}$ and simply solve the problem with gradient descent to optimize equation 2.13. This is theoretically an exact global minimum as it is a sum of two convex function. However, it is unlikely that the optimization exactly reaches the zero value for a given code component because the sparsity objective term is not differentiable at zero. The h_i^j are then prone to oscillate around zero rather than reaching it whereas we would like to have these shrunk component value to be exactly zero to produce sparse code vectors. One of the most popular methods consists in first updating h_i^j with one step of gradient descent of the reconstruction objective term, *i.e.*:

$$h_i^j \leftarrow h_i^j - 2\rho \mathbf{D_j}^T(\mathbf{Dh_i} - \mathbf{x_i}) \tag{2.14}$$

and then setting h_i^j to zero if the gradient of the sparsity term drives h_i^j to change its sign or apply the traditional gradient update otherwise such as:

$$h_i^j = \begin{cases} 0 & sign(h_i^j) \neq sin(h_i^j - \rho\lambda sign(h_i^j)) \\ h_i^j - \rho\lambda sign(h_i^j) & else \end{cases}$$

This inference procedure corresponds to the Iterative Shrinking and Thresholding Algorithm (ISTA) [DDDM04, BT09]. The interesting property of this algorithm is that it creates saturating regions nearby zero so as to force components that do not explain well the input data to be exactly shrunk to zero. This makes the different components competing with each other in order to explain well the input data. A lot of other inference algorithms have been proposed in the literature of sparse coding such as Least Angle Regression Selection (LARS) [EHJ+04] which is used in the popular online sparse coding method of Mairal et al. [MBPS09], *grafting-based* feature selection methods like [PT03] or the *Feature-sign* inference scheme proposed by Lee et al. [LBRN06] that have been shown to outperform both of these methods. The main drawback of such inference methods is that they rely on an optimization procedure that can be slow and thus not practical for real time feature extraction. A lot of methods attempted to address this issue such as Fast ISTA (FISTA) [BT09] which is an extension of ISTA where the main difference is the introduction of a *momentum* term in the objective function. Loosely speaking, the new code vector is given by the shrinkage function applied to the previous code vector. Although it has been shown that FISTA has significantly lower convergence time than ISTA, it still requires some optimization iteration to produce sparse codes. An alternative research direction that has been investigated consists in jointly learning the sparse coding objective function and with parametric and trainable module that is able to fastly produce good approximations of the code vectors. This is the main objective of Predictive Sparse Decomposition (PSD) that was proposed in Kavukcuoglu et al. [KRL10]. The authors introduced an auxiliary term in the sparse coding cost function that allows the system to infer the codes fastly thanks to a jointly learned

parametric map $f_\alpha(\mathbf{x_i})$. The newly obtained regularized objective function now takes the following form:

$$\sum_i ||\mathbf{x_i} - \mathbf{D}\mathbf{h_i}||_2^2 + \lambda \sum_j |\mathbf{h_i^j}| + ||\mathbf{h_i} - f(\mathbf{x_i})||_2^2 \tag{2.15}$$

where $\mathbf{D}$ is a linear model that can be seen as a linear decoder, the free codes $\mathbf{h_i}$ are learned by the Sparse Coding part of the equation, and $f(\mathbf{x_i}) = \tilde{\mathbf{h}}_\mathbf{i}$ corresponds to the parametric mapping $f_\alpha(\mathbf{x_i})$ allowing fast inference such that:

$$f(\mathbf{x_i}) = TanH(\mathbf{D^T}\mathbf{x_i} + \mathbf{b}) \tag{2.16}$$

Later, Gregor et al. [GL10] noticed that one drawback of such predictive inference is that it is difficult for the system to produce sparse code values close to zero as the derivative of the $TanH$ function is high near 0. Secondly, the learning procedure is not likely to exhibit competition between units which corresponding basis functions could reconstruct the input equally well. To alleviate such problems, the authors proposed the so-called Learning ISTA (LISTA) method which combine both advantages of *i)* shrinking and competition between units of ISTA-based inference algorithms and *ii)* deterministic parametric inference to approximate good code vectors as in PSD.

In the following subsection, we will see that Sparse Coding methods have been shown to be a better alternative than VQ coding to encode local features and that they have been successfully used in the scope of image classification.

Sparse Coding based aggregated representations

As we have seen in the previous subsection, Sparse Coding can be seen as a powerful tool to reduce the error quantization involved in vector quantization methods. Then, they have been naturally considered in the computer and machine learning community to design more informative image representation by aggregating local features encoded with Sparse Coding [YYGH09, ZGL+13, TLSJ12, ZWH+13, BBLP10, WP14, YYH10] (rather than VQ coding). One of the simplest methods in this research line is the so-called *Sparse Coding SVM (SC-SVM)* [WP14] that consists in: *i)* extracting SIFT descriptors on the training images, *ii)* encoding them with the online sparse coding algorithm of Mairal et al. [MBPS09], and *iii)* aggregating them in a single image representation with an average-pooling procedure. Other methods rather make use of the Spatial Pyramid Matching scheme [LSP06] with a max-pooling aggregation scheme. More concretely, each descriptor is encoded with the SC scheme depicted in [LBRN06] using the Feature-sign algorithm for the inference and using L_2-regularized dual Lagragian codebook learning algorithm. Each region R_i of the multi-resolutional spatial pyramid is then represented by a global representation $\mathbf{z_i} \in R^k$ where k is the number of elements in the learned codebook. Each component z_i^j is the max value among all the j-th components of

code vectors lying in the spatial region R_i. As for the original scheme of [LSP06], the different representations for each resolution of the pyramid are concatenated and a linear SVM is learned with stochastic-gradient optimization (see section 2.4) instead of a kernelized SVM algorithm with Histogram Intersection kernel. Later, several methods proposed to enhance such representations by adapting traditional unsupervised sparse coding algorithms in a supervised fashion. One of the most popular approach in this research line is the one of Mairal et al. [MPS+09] which requires that each local feature is assigned to a label. However, this could be an undesired property in a framework of aggregation-based representation where a lot of extracted localized features do not especially correspond to semantically relevant information. Later, Boureau et al. [BBLP10] rather consider to jointly learn the basis vectors and code vectors with the discriminative objective function that takes as input the image-level aggregated representation obtained by an average pooling over the code vectors of the image. Moreover, to provide some spatially localized consistency, the authors consider to encode spatial neighborhood of densely extracted local descriptors, *i.e.* all the descriptors in a localized spatial region are concatenated and the resulting descriptors are passed to the sparse coding algorithm. The authors consider the full pipeline as a deep architecture (see section 2.4) and used the back-propagation algorithm [RHW88, LBBH98] to jointly learn the sparse encoding parameters and the parameters of the discriminative models used on top of the global representation. In a similar way, Yang et al. [YYH10] proposed a supervised extension of their methods [YYGH09] by considering gradient-based optimization across their multi-resolutional pyramidal representation combined with max-pooling.

A noticeable disadvantage of sparse coding approaches (*i.e.* those attempting to minimize equation 2.12) is that the codes are not robust to small changes in the input space, *i.e.* two nearby points in the input space are not likely to have close code vectors. Indeed, as the L_1 penalty function is not smooth, the inference procedure is not likely to decompose two similar points into the same generating basis vectors. Locality-constrained Linear Coding (LLC) is a coding scheme introduced in [WYY+10] motivated by the work of Yu et al. [YZG09] where the authors proposed to alleviate the above mentioned drawbacks of both VQ coding and Sparse Coding methods. Their approach consists in integrating a locality constrain regularization term that force the codes to be sparse, as in SC, but whose components only involve codebook vectors $\mathbf{d_j}$ that are close to the input point $\mathbf{x_i}$ to be encoded. A dictionary is learned such that each input point can be expressed in a (low rank) local coordinate system whose basis vectors are closed to the input point. More formally, the objective term of LLC is given by:

$$\sum_i ||\mathbf{x_i} - \mathbf{D}\mathbf{h_i}||_2^2 + \lambda|\mathbf{h_i} \odot \mathbf{a_i}| \tag{2.17}$$

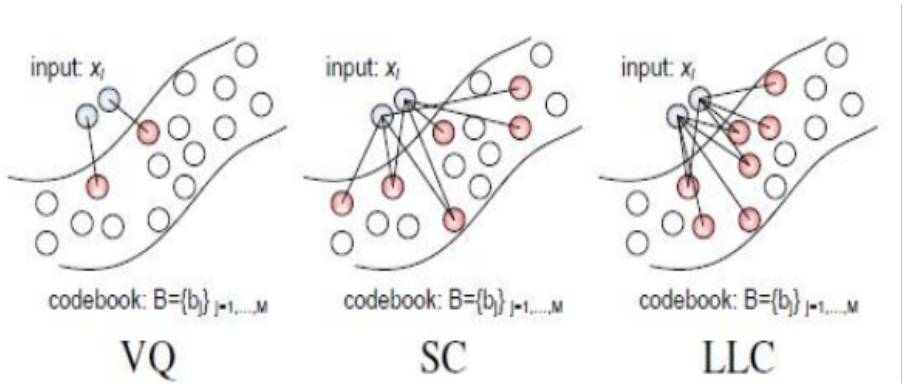


FIGURE 2.3 – Geometrical Comparison of LLC and other coding schemes. Figure taken from [WYY+10].

subject to $\mathbf{1}^T\mathbf{h_i} = 1$ and where:

$$\mathbf{a_i} = \exp(\frac{dist(\mathbf{x_i}, \mathbf{D})}{\sigma}) \tag{2.18}$$

where $\odot$ corresponds to the element-wise multiplication operator. Then, $\mathbf{a_i}$ corresponds to a vector containing all the Euclidian distances between the input $\mathbf{x_i}$ and the column vectors $\mathbf{d_j}$ of $\mathbf{D}$. In other words, $\mathbf{a_i} = \exp(\frac{||\mathbf{x_i}-\mathbf{d_1})||_2}{\sigma}), \exp(\frac{||\mathbf{x_i}-\mathbf{d_2})||_2}{\sigma}), \ldots \exp(\frac{||\mathbf{x_i}-\mathbf{d_k})||_2}{\sigma})]^T$.

Contrary to traditional Sparse Coding schemes, the inference procedure in LLC does not suffer from high computation time as it can be computed analytically rather than through an optimization procedure as shown in [WYY+10]. The authors also proposed to speed up the inference time by approximating the LLC objective function 2.17 by considering the fact that only a few elements in the codebook allow reconstructing the input vector. It turns out that this local coordinates system is mainly composed of codebook vectors that are close to the input vector. This suggests that the optimization of the codes could be performed on a reduced subset of basis vectors corresponding to nearest neighbors of $\mathbf{x_i}$. The new optimization procedure for equation 2.17 now amounts to finding the k nearest neighbors of $\mathbf{x_i}$ and perform the optimization without the locality-based regularization term. The codebook vectors $\mathbf{d_j}$ are then updated in a traditional fashion using a simple stochastic gradient procedure given the inferred codes. Then, the codes are constrained to be located inside the unit ball. Figure 2.3 provides a geometrical illustration of the main difference between the three main kinds of coding schemes presented in this section. We can see that, in LLC, nearby input samples are likely to have similar locality constrained codes. It is not the case in sparse coding mostly because of both the non-smoothness of the L_1 penalty and the over-completeness of the code vectors. We see that the LLC codes are also more robust than those produced by VQ coding schemes. Indeed, in VQ coding, each input vector is only represented by one basis

vector which is likely to produce different codes for similar input vectors due to large quantization error. Moreover, the LLC approximation scheme using nearest neighbor codebook vectors can be seen as an attempt toward creating localized low-rank coordinate systems in the feature space. This allows a kind of local dimensionality reduction adapted to different regions of the input space. This *low-rankness* property has been further explored [JGP14, ZGL+13] to be explicitly optimized in the objective function so as to improve this localized dimensionality reduction as well as the stability of the code vectors. Such locality-constrained coding techniques have shown impressive performance gain over classical Bag-of-Words and Sparse Coding models in several object classification tasks. LLC was reported as the most promising method before that Fisher Vectors and Deep Convolutional Neural Networks [LBBH98, KSH12] became state-of-the-art methods.

2.3.3 Fisher Vectors and VLAD

As opposed to the BoVW that only encodes occurrences of visual words in a document, the *Fisher Vector* (FV) representation proposed by Peronnin et al [PSM10, PD07, SPMV13] rather encodes relationship between a local feature and a given generative model parametrized by a set of parameters $\mathbf{\Theta}$. This embedding is based on the Information Geometry (IG) theory [Ama87] and represents each local feature with a vector of gradient of the log-likelihood with respect to all the parameters in $\mathbf{\Theta}$. This give rises to a D-dimensional representation vector with $|\mathbf{\Theta}| = D$. As for the BoVW model, we want to represent an image $\mathcal{I}_X$ by a set of local feature vectors $\mathcal{X} = \{\mathbf{x_i}\}$ with $|\mathcal{X}| = N$. This is done by encoding each local feature with the Fisher embedding $\varphi_{FV}(\mathbf{x_i}) : R^d \rightarrow R^D$. Similarly to BOVW, all the resulting feature-level Fisher Vectors are then aggregated so as to produce a global Fisher Vector for the image:

$$\Phi_{FV}(\mathcal{X}) = \frac{1}{N}\sum_{i}^{N}\varphi_{FV}(\mathbf{x_i}) = \frac{1}{N}\sum_{i}^{N}\mathbf{L}\nabla_\theta \log p(\mathbf{x_i}|\theta) \tag{2.19}$$

where $\nabla_\theta \log p(\mathbf{x}|\theta) = \frac{\partial \log p(\mathbf{x}|\theta)}{\partial \theta} \in R^D$ and $\mathbf{L}$ is a normalization matrix such that $(\mathbf{L}^T\mathbf{L}) = \mathbf{F}^{-1}$ where $\mathbf{F}$ is the so-called *Fisher Information Matrix* given by $\mathbf{F} = E_{\mathbf{x_i},\mathbf{x_j}}\left\{\nabla_\theta \log p(\mathbf{x_i}|\theta)\nabla_\theta \log p(\mathbf{x_j}|\theta)^T\right\}$.

Underlying generative model

In VQ coding, the input distribution is modeled by explanatory modes, the centroids $\mathbf{c_j}$ of local clusters, and the inference procedure consists, for given input point $\mathbf{x_i}$, in returning its nearest neighbor among these centroids. The main drawback of VQ coding come from this hard assignment step which does not allow to fit the input distribution as precisely as we would like to do so as to avoid quantization errors. For this reason, a common choice for the underlying generative model is the Gaussian

Mixture Model (GMM) [Bis06]. In GMM, we assume the input distribution to be generated by a set of k multi-variate gaussian distributions $\{\mathcal{G}_j\}$ with parameters $\theta_j = \{(\mu_{\theta_j}, \Sigma_{\theta_j})\}$ where $\boldsymbol{\mu_{\theta_j}}$ and Σ_{θ_j} respectively corresponds to the mode of $\mathcal{G}_j$, *i.e.* its mean, and its covariance matrix. The probability of any input point $\mathbf{x} \in R^d$ given any gaussian $\mathcal{G}_j$ is given by:

$$p(\mathbf{x}|\theta_j) = \mathcal{N}(\boldsymbol{\mu_{\theta_j}}, \Sigma_{\theta_j}) \tag{2.20}$$

The marginal probability of any input points is then given by the sum rule:

$$p(\mathbf{x}) = \sum_j p(\mathbf{x}, \theta_j) = \sum_j p(\theta_j) p(\mathbf{x}|\theta_j) \tag{2.21}$$

where $p(\theta_j)$ corresponds to the prior probability of the gaussian $\mathcal{G}_j$ that is often noted π_j in the literature. From now, we will denote $\Theta = \{(\theta_j, \pi_j)\}$ the set of parameters of the GMM that has to be learned. Baye's rule allows us to compute the posterior probability of any gaussian $\mathcal{G}_j$ given any input point $\mathbf{x}$:

$$\gamma_{ij} = p(\theta_j|\mathbf{x}) = \frac{\pi_j . p(\mathbf{x}|\theta_j)}{p(\mathbf{x})} = \frac{\pi_j . p(\mathbf{x}|\theta_j)}{\sum_{j\prime} \pi_{j\prime} . p(\mathbf{x}|\theta_{j\prime})} \tag{2.22}$$

This corresponds to the probability that the particular gaussian $\mathcal{G}_j$ generated the input point $\mathbf{x}$. This can be considered as a soft assignment value of input point $\mathbf{x}$ to gaussian $\mathcal{G}_j$ where $p(\theta|\mathbf{x})$ is a probability distribution that sum to one and the parameter π_j can be estimated as the expected number of points assigned to gaussian $\mathcal{G}_j$

$$\pi_j = \sum_i p(\theta_j|\mathbf{x_i}) = \sum_i \gamma_{ij} \tag{2.23}$$

Finding Maximum Likelihood Estimation (MLE) of parameters of distributions that depends on latent variables, *i.e.* that we do not directly observe as our target gaussian variables, can be perform thanks to a class of iterative algorithms so-called *Expectation Maximization* (EM) [DLR77] that are composed of two main steps *i)* the Expectation step where we estimate the posterior probabilities given the current estimate the model parameters, and *ii)* the Maximization step where we optimize the Maximum Likelihood of the model parameters given the posterior probabilities of the latent variables estimated in the previous step. For GMM, the EM algorithm consists in iterating over

— **E-step** Estimate posterior probabilities γ_{ij} of latent variables with eq. 2.22.

— **M-step** Estimate the gaussian parameters given the γ_{ij}'s with

$$\mu_{\theta_j} = \frac{\sum_i p(\theta_j|\mathbf{x}) \mathbf{x_i}}{\sum_i p(\theta_j|\mathbf{x})} = \frac{\sum_i \gamma_{ij} \mathbf{x_i}}{\sum_i \gamma_{ij}} \tag{2.24}$$

$$\Sigma_{\theta_j} = \frac{\sum_i \gamma_{ij} (\mathbf{x_i} - \mu_{\theta_j})(\mathbf{x_i} - \mu_{\theta_j})^T}{\sum_i \gamma_{ij}} \tag{2.25}$$

One of the main drawbacks of EM algorithms is that they are often slow especially in high-dimensional spaces with fancy distributions. In the Fisher Vector framework, we often consider initializing the model with a better guess on the parameters by first running a K-Means algorithm and then fine-tuning the GMM parameters with the EM algorithm.

The Fisher Embedding

Contrary to VQ coding or Sparse coding-based inference schemes, the input descriptor $\mathbf{x_i}$ are embedded with an optimization-free procedure. It simply consists in computing the gradient of the log-likelihood of the GMM model with respect to the different parameters evaluated at the specific input space location $\mathbf{x_i}$. This gives rise to the sub-vectors computed for each gaussian component:

$$\varphi_{FV}^{\pi_j}(\mathbf{x_i}) = \frac{\partial \log p(\mathbf{x_i}|\theta)}{\partial \pi_j} = \gamma_{ij} - \pi_j \tag{2.26}$$

$$\varphi_{FV}^{\mu_{\theta j}}(\mathbf{x_i}) = \frac{\partial \log p(\mathbf{x_i}|\theta)}{\partial \mu_{\theta j}} = \gamma_{ij} \frac{\mathbf{x_i} - \mu_{\theta j}}{\sigma_j^2} \tag{2.27}$$

$$\varphi_{FV}^{\Sigma_{\theta j}}(\mathbf{x_i}) = \frac{\partial \log p(\mathbf{x_i}|\theta)}{\partial \Sigma_{\theta j}} = \gamma_{ij} [\frac{(\mathbf{x_i} - \mu_{\theta j})^2}{\sigma_j^3} - \frac{1}{\sigma_j}] \tag{2.28}$$

with γ_{ij} given by equation 2.22. Actually, while looking at equation 2.19, the above gradient should be normalized by some matrix $\mathbf{L}$ corresponding to the *square-rooted* inverse Fisher Information Matrix (FIM). The authors in [PSM10] have shown that this matrix can be approximated as diagonal if we assume the soft assignments γ_{ij} close enough to 1 leading to the following newly image-level aggregated representation:

$$\Phi_{FV}^{\pi_j}(\mathcal{X}) = \frac{1}{\sqrt{\pi_j}} \sum_i \gamma_{ij} - \pi_j \tag{2.29}$$

$$\Phi_{FV}^{\mu_{\theta j}}(\mathcal{X}) = \frac{1}{\sqrt{\pi_j}} \sum_i \gamma_{ij} \frac{\mathbf{x_i} - \mu_{\theta j}}{\sigma_j^2} \tag{2.30}$$

$$\Phi_{FV}^{\Sigma_{\theta j}}(\mathcal{X}) = \frac{1}{\sqrt{\pi_j}} \sum_i \gamma_{ij} [\frac{(\mathbf{x_i} - \mu_{\theta j})^2}{\sigma_j^2} - 1] \tag{2.31}$$

The whole Fisher Vector for $\mathbf{x_i}$ is given by the concatenation:

$$\Phi_{FV}(\mathcal{X}) = \biguplus_j [\Phi_{FV}^{\pi_j}(\mathcal{X}), \Phi_{FV}^{\mu_{\theta j}}(\mathcal{X}), \Phi_{FV}^{\Sigma_{\theta j}}(\mathbf{x})]^T \tag{2.32}$$

where $\biguplus_j$ is the concatenation operator over j.

Properties of Fisher Vectors Intuitively, a given component $\varphi_{FV}^{\theta_j}(\mathbf{x_i})$ encodes how the parameter θ_j contribute to generate the input sample $\mathbf{x_i}$ and how it should be modified to increase as much as possible the log-likelihood for that point. This property can be observed if we have a look at equations 2.29, 2.30 and 2.31 where γ_{ij} encodes how much gaussian $\mathcal{G}_j$ is likely to generate input sample $\mathbf{x_i}$. The term $\frac{\mathbf{x_i}-\mu_{\theta j}}{\sigma_j^2}$ is a gradient directional information telling us how the parameter $\mu_{\theta j}$ should be updated to better fit the log-likelihood for the specific input sample $\mathbf{x_i}$. An interesting property is that if a given input sample is perfectly modeled by the underlying generative model, *i.e.* the log-likelihood is maximal for that point, then the gradients will be equal to zero as well as the resulting Fishers Vector $\varphi_{FV}(\mathbf{x_i})$. In other, words, if local points are perfectly modeled by the generative model they do not bring any information with respect to the statistics of the local features in the database. Consequently, they do not contribute to the global image representation when pooling.

Normalization issues for Fisher Vectors As for the BoVW representation, Fisher Vectors have been used in a wide range of computer vision tasks such as object retrieval [PLSP10], image classification such as [DRS11] or in the ImageNet competition where the authors [PSM10] won the fourth edition in 2011 making Fisher Vectors the state-of-the-art before deep learning methods arrived [KSH12, SZ14b, SLJ+15]. The Fisher embedding was also recognized as a better representation and classification scheme in some kind of specific classification tasks as Fine-Grained Classification (FGC) [GMJP14] where they beat CNN-based approaches [KSH12]. In [GMJP14], the authors consider a deterministic prepossessing of the local descriptors which is known in the literature as *RootSIFT* [AZ12]. This was originally motivated by the fact that it corresponds to the explicit embedding associated with the *Hellinger Kernel* often used to compare histograms (as SIFT descriptors that are histograms of gradients). The authors also applied a PCA post processing of the descriptors. This is also a crucial step in the context of Fisher Vectors because the inverse Fisher Information Matrix (FIM) normalization factor involved in equations 2.29, 2.30 and 2.31 assumes a diagonal covariance matrix of the input vectors. It is also interesting to note that the normalized factor $\frac{1}{\sqrt{\pi_k}}$ coming from this approximated inverse FIM also plays the role of a TF-IDF scheme as seen in the BOVW framework. Indeed, low values of π_k is directly related to the a priori probability of occurrence of the k-th gaussian. Last but not least, one of the major improvement that has been done for the Fisher Vector representation is a post processing step called *power normalization* [PSM10]. It corresponds to a component wise non linearity such that the new component value is given by $x_i^{j\prime} = sign(x_i^j)|x_i^j|^\alpha$ where alpha is a real number that is less than 1. The final vector is then L_2 normalized in a traditional way to project it on the unit ball. This post normalization step has been shown [JBJG12, PSM10] to significantly improve classification performance of Fisher Vectors and VLAD. This is mainly due to the fact that it causes a kind of

signal spreading that acts as variance stabilization between the different clusters of the generative model.

Vector of Locally Aggregated Descriptors (VLAD)

The VLAD embedding proposed by Jegou et al. [JPD+12] can be seen as a simplified version of the Fisher embedding by only considering non-normalized differences between input points and hard assigned k-means centroids:

$$\Phi_{VLAD}(\mathcal{X}) = \biguplus_{j} \sum_{\mathbf{x_i} \in \mathcal{X}} \gamma_{ij}(\mathbf{x_i} - \mathbf{c_j}) \tag{2.33}$$

where $\gamma_{ij} = I(j = argmin_{j\prime}||\mathbf{x_i} - \mathbf{c_{j\prime}}||_2)$ and the set of k centroids $\{c_j\}$ has been learned with the K-means algorithm. It can be seen that the VLAD representation is very related to Fisher Vector if we re-rewrite with the following form:

$$\Phi_{VLAD}(\mathcal{X}) = \biguplus_{j} \nabla_{\mathbf{c_j}} E(\mathcal{X}, \mathbf{C}) \tag{2.34}$$

where $\nabla_{\mathbf{c_j}} E(\mathcal{X}, C) = \frac{\partial E(\mathcal{X}, \mathbf{C})}{\partial \mathbf{c_j}}$, given by Equation 2.7, is the gradient of the energy term related to the K-means algorithm.

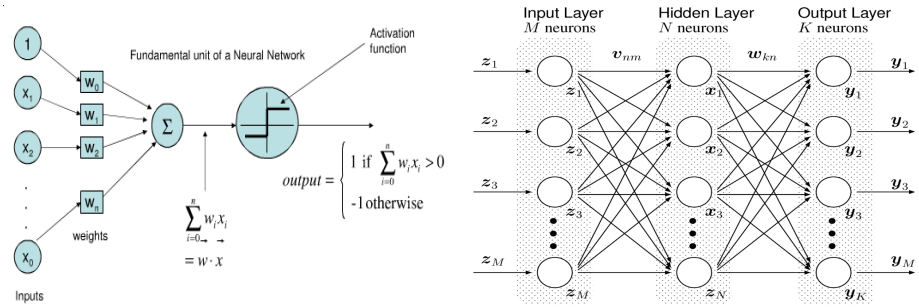
Recent Advances in Fisher Vectors and VLAD representations Since 2012 and the predominance of deep learning methods, the computer vision community have been trying to understand what makes them win more than 15% top-5 accuracy error on ImageNet. Thus, Fishers Vectors and VLAD extensions proposed in the last three years have been focused on integrating this notion of depth in their architectures. The first method that was proposed to this this attempt was the Deep Fisher Vector [SVZ13] representation introduced by Simonyan et al. The authors proposed a Fisher Vector based on stackable layers that can be learned in a greedy-layer wise fashion. The first layer consists in densely extracting low-level handcrafted features in the same way than the original FV embedding. The next stage of the layer consists in performing semi-local Fisher embedding rather than fully aggregating every descriptors of the image into a single representation. This leads to a set of semi-local Fisher Vectors regularly sampled that encodes semi-local contents of the image (*i.e.* a particular region). The resulting sub-vectors are compressed using a supervised dimensionality reduction technique, and, as in the spatial pooling layers of a Convolutional Neural Network (CNN) architectures (see next section), localized subsets of Fisher Vectors are spatially pooled. This is done by concatenating semi-local FV of pool and the resulting local intermediate Fisher Vectors are then compressed using classical PCA and L_2 normalized. This results in a reduced subset of local descriptors that can be plugged in the next layer and the process is repeated greedily until the number of desired layer is reached. This procedure is heavily inspired from the first form of unsupervised deep learning architectures proposed by Hinton

et al. [HOT06]. Later, Peng et al. [PZQP14] proposed a very similar framework called *Stacked Fisher Vectors* adapted for action recognition. Although these methods find motivations in state-of-the-art deep architectures, they lose the main strength of such methods because it not possible to fine-tune the whole architecture with backpropagation [RHW85, LBBH98]. Thus, it is not possible to jointly learn all the layers of the model. To overcome this issue, Sydorov et al. [SSL14] recently proposed an end to end discriminative model based on Fisher Vectors by jointly learning the SVM parameters and the Fishers embedding parameters in a supervised manner. More recently, Peronnin et al. proposed in [PL15] another alternative to combine the benefit of CNN and those of the Fisher Vector embedding. They rather proposed a hybrid architecture consisting in learning in an unsupervised manner classical Fisher Vectors layers and multiple fully connected layers with *Rectified Linear Units (ReLU)* learned in a supervised manner with backpropagation. A very similar improvement has been proposed for the VLAD representation such as in [PWQP14] where the authors proposed to learn in a gradient-based fashion the dictionary used for the VLAD embedding rather learning it in an unsupervised manner. A similar approach has been proposed by Arandjelovic et al. [AGT+15] in the context of place recognition. The authors proposed to improve the VLAD representation by designing a trainable VLAD layer on top of a CNN allowing to directly integrate a supervised dictionary learning procedure in the CNN architecture.

Discussion As we have seen so far, most coding schemes and aggregation-based methods are generally applied on top of a handcrafted feature extraction process. The coding phase consists in *"capturing"* the interesting information in the training dataset of descriptors. The pooling phase allows merging in a deterministic way the extracted information from the set of local descriptors into a global vector representation. These processing steps, *i.e.* feature extraction, coding, pooling and classification are learned independently from each other thus involving complex machine learning systems where a lot of prior information is generally required by the human.

2.4 Convolutional Neural Networks

Although it has received a lot of attention only recently (since 2013), deep learning is an old research direction that has been widely studied since the works of Hinton [RHW85, RHW88, HOT06], LeCun [LCJB+89, LBBH98] and Bengio [Ben93, LB95, BLP+07] in the end of the 80's and 90's. The high amount of attention toward these methods was initially caused by the impressive performance gap within the ImageNet [DDS+09] classification challenge compared to other state-of-the-art methods. In the remainder of this section, we are going to briefly review state-of-the-art deep learning methods for image classification. In particular, we first focus on Convolutional Neural Networks (ConvNet or CNN) [LCJB+89, LBBH98, KSH12]



Perceptron Multi-Layer Perceptron

FIGURE 2.4 – Examples of Feed-Forward Neural Network Architectures.

which is a supervised deep learning architecture that allows us to learn a progressive embedding directly from the pixel space by optimizing a multi-layer feed-forward model. We first explain the *Multi-Layer Perceptron* model which is the basic module a ConvNet is composed of. We then give the details of the learning procedure of ConvNet and we review recent advances in ConvNet architectures which make them the dominant approach in a wide range of computer vision tasks.

2.4.1 Multi-Layer Perceptron

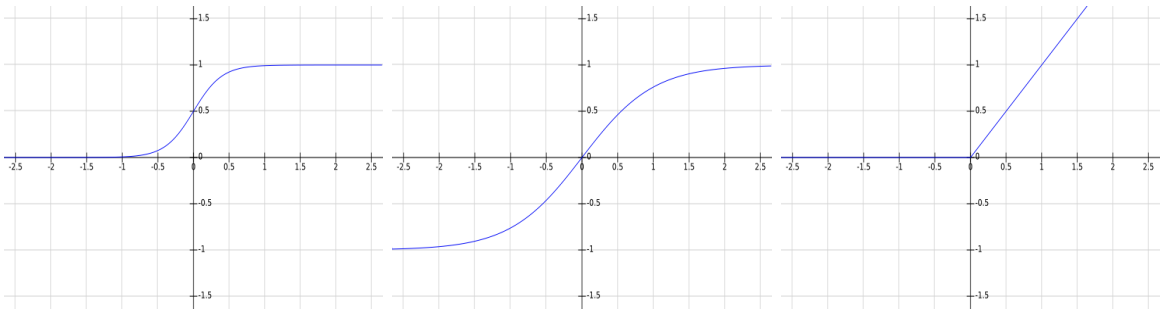
Multi-Layer Perceptron (MLP) [Cyb89, HN04] is often described in the literature as a set of computation units called neurons (each unit is a perceptron unit [Ros58]) interconnected through weighted connections whose weights are parameters of the model to be learned. In MLP, neurons are organized in layers where no connections are allowed between the neurons of the same layer. Each neuron's output value can be computed from the neurons of the previous layer until the end of the network is reached. As illustrated on Figure 2.4, the network is divided into three main parts *i)* the input layer where each value is clamped to a particular component of an input vector $\mathbf{x_i} \in R^D$, *ii)* the output layer returning an output vector $\mathbf{y_i} \in R^k$ where k is the number of units in the output layer, and *iii)* one (or more) hidden layer(s) where output values can be considered as intermediate representations of the input vector $\mathbf{x_i}$. The prediction function of a MLP of L layers can be expressed as:

$$\tilde{y}_i = F_{\mathbf{\Theta}}(\mathbf{x_i}) = F_{\theta_L}(F_{\theta_{L-1}}(...F_{\theta_2}(F_{\theta_1}(\mathbf{x_i}))...)) \tag{2.35}$$

where the specific output of layer l $\mathbf{x_i^l}$ is given by:

$$\mathbf{x_i^{l+1}} = F_{\theta_l}(\mathbf{x_i^l}) = \sigma(\mathbf{W^l x_i^l} + \mathbf{b^l}) \tag{2.36}$$

and the network output value $\tilde{y}_i$ is obtained with a recursion of Equation 2.36 until the final layer L is reached. This corresponds to the so-called *forward pass* step and $\mathbf{\Theta} = \{\theta_1, \theta_2, ..., \theta_L\}$ is the set of parameters of the model where $\theta_l = \{\mathbf{W^l}, \mathbf{b^l}\}$ is the set of parameters of layer l. Each computation of a particular neuron's output



$Sigmoid(z) = \frac{\mathbf{e}^z}{1+\mathbf{e}^z}$ $TanH(z) = \frac{\mathbf{e}^z - \mathbf{e}^{-z}}{\mathbf{e}^z + \mathbf{e}^{-z}}$ $ReLU(z) = max(0, z)$

FIGURE 2.5 – Common Neural Networks activation functions.

is followed by applying a nonlinear activation function $\sigma(.)$ that is generally chosen among the *sigmoid*, the *hyperbolic tangent* or the *Rectified Linear Unit (ReLu)* function which are illustrated in Figure 2.5.

Gradient descent optimization These kinds of methods received a lot of attention in the literature [LBBH98, Bot10]. It consists in finding the direction in the parameter space corresponding to the greatest loss of the cost function. This amounts to locally updating the parameters $\mathbf{w}$ of the model so as to go *downhill* the energy landscape corresponding to the shape of the energy function with respect to the different values of the parameter vector $\mathbf{w}$. The update rule for a training set composed of a set of N sample vectors $\mathcal{X} = \{\mathbf{x_i}\}_{i \leq N}$ associated to a set of labels $\mathcal{Y} = \{\mathbf{y_i}\}_{i \leq N}$ thus consists as follow:

$$\mathbf{w_{t+1}} = \mathbf{w_t} - \mu \nabla_w E(\mathbf{w}, \mathcal{X}, \mathcal{Y})|_{\mathbf{w_t}} \tag{2.37}$$

where $E(w, \mathcal{X}, \mathcal{Y})$ is a differentiable energy/cost function (*e.g.* the squared Euclidian loss) and the gradient vector $\nabla_w E(\mathbf{w}, \mathcal{X}, \mathcal{Y})$ is given by the partial derivatives the energy function:

$$\nabla_w E(\mathbf{w}, \mathcal{X}, \mathcal{Y})|_{\mathbf{w_t}} = \sum_{\mathbf{x_i} \in \mathcal{X}} \sum_j \frac{\partial E(\mathbf{w}, \mathbf{x_i}, y_i)}{\partial w_j}|_{w_t} . \vec{e_j} \tag{2.38}$$

and μ is the learning rate that controls the magnitude of the local displacements in the parameters space. This hyper-parameter needs to be set carefully. Intuitively, if the learning rate is set with a value that is too low, the algorithm will take a long time to converge because it will move slowly in the parameters space. If it is too high, the algorithm is likely to miss the local minimum of the cost function or oscillate around it without reaching it. In the literature, this learning procedure is often referred as *Offline Learning* or *Batch Learning* because each iteration, denoted as *epoch*, consists in evaluating the gradient for the whole training set before updating the weights. In contrast, we denote *Online Learning* the learning procedure that consists in updating

the weights *on the fly* for each training sample $\mathbf{x_i}$. The estimated gradient does not correspond anymore to the exact gradient of the total energy function and so the update direction does not correspond anymore the optimal direction to go to minimize the cost function evaluated on the whole traning set. Instead, it rather corresponds to a noisy gradient direction allowing to move faster in the parameter space. This is why gradient-based online learning is also referred in the literature as Stochastic Gradient Descent (SGD).

The BackPropagation (BP) algorithm An efficient way to compute in a tractable way the partial derivatives of a multi-layer model is the so-called backpropagation algorithm [RHW88, LCJB+89, LBBH98]. It states that, as $F_{\mathbf{\Theta}}$ is a composition of functions F_{θ_l}, partial derivatives with respect to any parameter θ_l can be obtained tanks to the chain rule:

$$\Delta\theta_l = \frac{\partial E}{\partial \theta_l} = \frac{\partial E}{\partial F_{\theta_L}} . \frac{\partial F_{\theta_L}}{\partial F_{\theta_{L-1}}} ... \frac{\partial F_{\theta_{l+2}}}{\partial F_{\theta_{l+1}}} . \frac{\partial F_{\theta_{l+1}}}{\partial \theta_l} \tag{2.39}$$

where $\frac{\partial F_{\theta_l}}{\partial F_{\theta_{l-1}}} = \frac{\partial F_{\theta_l}}{\partial \mathbf{x_i^l}}$ is the derivative of output of layer l with respect to the output of layer $l-1$. In order to update all the parameters of the model in a tractable way the *backward pass* can be performed in a feed forward manner by iterating from layer L to layer 1 thanks to the two following equations:

$$\frac{\partial E}{\partial \mathbf{x_i^l}} = \frac{\partial E}{\partial \mathbf{x_i^{l+1}}} \frac{\partial F_{\theta_l}(\mathbf{x_i^l}, \mathbf{W^l})}{\partial \mathbf{x_i^l}} \tag{2.40}$$

$$\frac{\partial E}{\partial \mathbf{W^l}} = \frac{\partial E}{\partial \mathbf{x_i^{l+1}}} \frac{\partial F_{\theta_l}(\mathbf{x_i^l}, \mathbf{W^l})}{\partial \mathbf{W^l}} \tag{2.41}$$

Although Multi-Layer Perceptron has been shown to be an *universal approximator* [Hor91], (*i.e.* it can approximate any non-linear functions), it requires in practice a huge number of parameters to do it and, this, it is not always possible to train it efficiently. MLP is a non-linear classification algorithm and has been intensively used in a wide range of pattern recognition applications starting from global handcrafted features or pixel representations. However, as we have said in section 2.2, those feature engineering can be highly time consuming and a lot of prior on the task are required. This makes them not very powerful for generic data representation. Moreover, such models take as input global high-dimensional representation which would cause serious overfitting effects, particularly for image signals where we need to take into account spatially localized correlations. Also, it does not take into account the *compositional* property of real-world objects, *i.e.* the fact that an object is often a composition of other objects which are themselves composed of lower-level objects and so on and so forth.

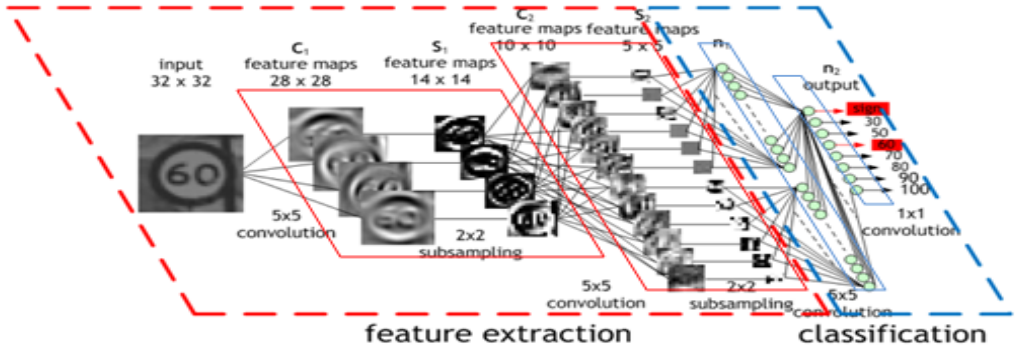


FIGURE 2.6 – Illustration of the LeNet5 Convolutional Neural Network architecture [LBBH98].

2.4.2 Convolutional Neural Networks

One of the main ideas behind CNN (and deep learning methods in general) is to teach a system to perform a progressive embedding from the raw signal to a global representation by learning a hierarchical bunch a feature extractors replicated all over the visual field. Such models are organized in successive layers (as in MLP) that are generally specialized to detect simple shapes (such as edges or corners) and the rest of the recognition process corresponds to other layers in charge of combining these low-level detections to model higher-level shapes. The process is repeated creating more and more abstract representations from which we can perform high-level processing tasks. This idea of replicating filters was first introduced by Fukushima et al. [Fuk80] in the so-called *Neocognitron* model and is the key idea behind Convolutional Neural Networks (CNN) that can be seen as an extension of MLP adapted for any multidimensional array signals characterized by local correlations. ConvNet were well popularized by works of Yann LeCun et al. [LCJB+89, LBBH98] and the most historical and famous CNN architecture corresponds to the so-called *LeNet 5* model (illustrated in Figure 2.6). This model corresponds to a multi-layer feed-forward architecture composed of three main kinds of layers:

— **Convolutional layers** They take images as input and they produce as output K *feature maps* each corresponding to the result of the convolution of the image with a particular convolution kernel (or filter). This convolution corresponds to a feature extraction process producing several feature maps respectively corresponding to K different convolution kernels looking for different patterns. Each spatial location of a given feature map corresponds to the pattern activation of the corresponding filter. This amounts to looking at every location of the image if a given pattern is likely to occur and this gives CNN the ability to be robust to translation. As for traditional MLP, a non-linear activation function is applied to each feature maps value. Historically, CNN models used to consider sigmoidal non-linearities. But, as shown

in Figure 2.5, the shape of such functions is very flat near saturated regimes which make gradient values very small at these locations. We can consider this phenomenon from a geometric perspective where the energy landscape of the model, with respect to the parameter space will be composed of highly flat regions and the model will thus be likely to get stuck in such flat and non-optimal regions of the parameter space. This is why state-of-the-art CNN architectures rather make use of ReLU nonlinear activation function.

— **Sub-sampling layers** They locally aggregate the previously computed feature activations at every spatial neighborhood of each feature map where different pooling strategy can be used (*e.g.* sum pooling, max pooling, etc.). As we can see in Figure 2.6, when we progress in the network, the size of the feature maps keeps decreasing until reaching a spatial resolution of size 1x1 thus leading to a K dimensional vector (where K is the number of convolutional filters of the last convolutional layer).

— **Fully Connected layer** This high-level representation is then passed to a non-convolutional feedforward neural networks (a MLP). It is denoted as Fully Connected layer because, unlike convolutional layer, every neuron of the layer l is connected to every unit of the $(l+1)$-th layer without filter replication. Actually, we can see these last layers as 1x1 convolution layers corresponding to simple dot products and the spatial pooling layer simply corresponds to the identity mapping because there is just one pixel location to pool.

This makes ConvNet very generic and, as far as the modules are differentiable (or at least piece-wise differentiable when considering ReLU non-linearities), the whole system can be learned efficiently thanks to the backpropagation algorithm. However, convolutional layers are a bit specific and we need a special rule to learn them in a tractable way.

Learning convolution kernel through sharing weights The *weight sharing* principle [LBBH98] is a key point of CNN and gives rise to a tractable and efficient learning algorithm because it implicitly implements convolution operations corresponding to filter replication over the visual fields. This gives CNN the ability to extract and detect any kind of features at any location in the input images and to be invariant to translation. Moreover, weight sharing makes the learning procedure of CNN much more efficient by heavily reducing the number of effective parameters to be learned. Weight sharing can be efficiently implemented by constraining every specific instance $\mathbf{W_k^{(x,y)}}$ of a particular kernel $\mathbf{W_k}$ to be updated with the same error term. A reasonable way to do this is to simply consider the sum over the update of the filter instances leading to:

$$\Delta \mathbf{W_k} = \sum_{(x,y)} \Delta \mathbf{W_k^{(x,y)}} \tag{2.42}$$

where $\Delta \mathbf{W}_{\mathbf{k}}^{(\mathbf{x},\mathbf{y})}$ is the error term computed with standard backpropagation to update the specific filter $\mathbf{W}_{\mathbf{k}}^{(\mathbf{x},\mathbf{y})}$ applied to the particular location (x, y). Those filters are updated with the global error term $\Delta \mathbf{W}_{\mathbf{k}}$ which keeps them equal during the learning process provided they have been initialized with the same value.

2.4.3 Recent Advances in CNN

In only 3 years, deep learning has turned into a particularly attractive research direction for a wide range of computer vision tasks. This comes from two main reasons that appeared during this period *i)* the availability of massive labelled datasets such as ImageNet [DDS+09] and *ii)* the availability of GPU-based open source implementations of CNN. This allowed benefiting from the good generalization abilities of CNN when learning from a lot of training data and to drastically speed up the computations involved in the learning phase. Although these two elements certainly brought significant increase in CNN efficiency and discrimination abilities, a lot of methods have been proposed in the past three years to improve the generalization of these architectures.

ConvNet regularization techniques A large number of clever model regularization and data normalization strategies have been integrated in ConvNet allowing them to better generalize or to learn faster and more efficiently. This is the case of the so-called *dropout* method that corresponds to a particular kind of model regularization that does not assume any prior distribution about the parameters of the models. Dropout was first introduced by Krizhevsky et al. [KSH12] who reported for the first time CNN classification performance on the ImageNet competition (developed in details in [SHK+14]). This very simple regularization procedure consists in randomly setting to zero the incoming and outgoing connections of each neuron with some probability p for each learning iteration. This implies that the neuron that have been turned off at a particular iteration will not be updated. This can be seen as a way to prevent neurons from co-adapting to each other which somehow makes the information captured by the network well distributed over the neurons. In other words, we prevent particular neurons, or sets of neurons, to over specialize on certain kind of information and we prevent certain neurons to capture too much correlated information with respect to each other. A second mathematical justification of dropout is that it can be seen as an efficient bagging method making use of approximated Bayesian model averaging over a huge number of sampled models that share their parameters. As shown in Figure 2.7, each training sample corresponds a particular model architecture that has only been trained on this input sample. If the whole architecture is composed of N parameters, learning with dropout is thus equivalent to sample N^2 different architectures that share parameters and only a fraction of them ever get trained with a single example. Once the training is done, the prediction phase simply consists in performing a forward pass without applying dropout. As shown in [SHK+14], it amounts to performing an approximate geometric

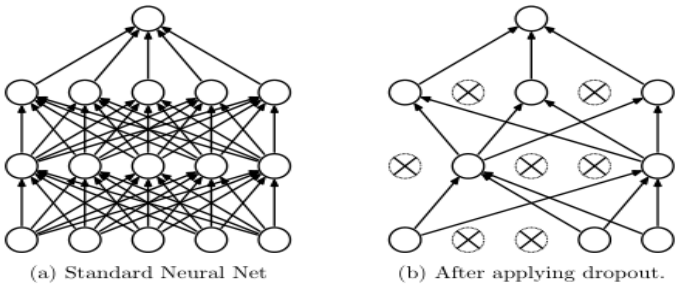


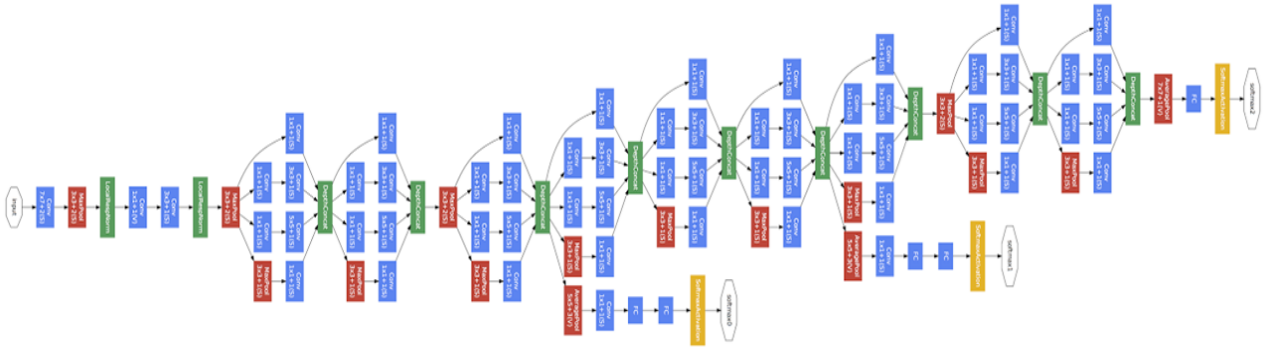
FIGURE 2.7 – Illustration of the dropout regularization technique. Activations of randomly sampled neurons are set to zero during each training iteration. This gives rise to different architectures that are jointly learnt by sharing their parameters. Taken from Srivasta et al. [SHK+14].

average over all this sampled models thus approximating a Bayesian combination model scheme in a very efficient way leading to drastically reduced generalization errors.

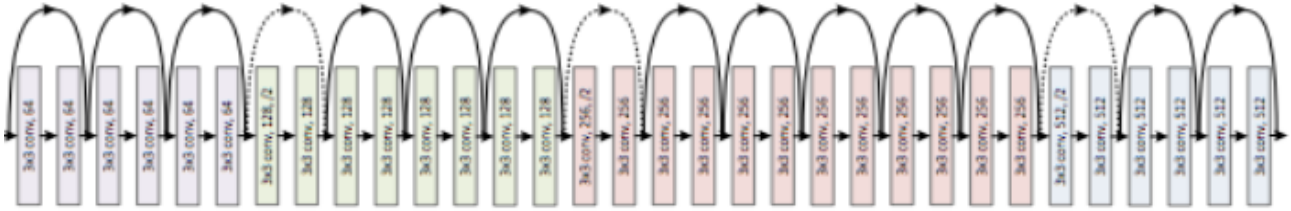
Advanced Normalization techniques Recently, Ioffe et al. [IS15] noticed that normalization was a very important issue of the learning procedure of a CNN. As a consequence, they introduced the *Batch Normalization* (BN) method which is a normalization technique that aims at reducing the so-called *Internal Covariate Shift* issue. Loosely speaking, it consists in the fact that the distribution of layers' inputs of the models change during the learning process which leads to difficulties for the model to continuously adapt to these distribution changes. The main idea of BN is to constraint the network to produce a particular prior distribution corresponding to whitened data. This is done by applying a batch level standardization and by providing to the network the ability to, somehow, undo this deterministic procedure if the optimization objective requires it. BN has the beneficial effect to avoid that the output values of a particular layer concentrate near saturated regimes of non linearities. This is why sigmoid non-linearities were not considered anymore in profit of ReLU non linearities. This also allows increasing learning rates which we did not before mainly because of such difficult normalization issues where the parameters scale was not controlled. This makes BN networks astoundingly much faster to learn (14 time faster). Recent advances in this research line has been developed in Riemannian Networks [Oll13, Oll15, MCO16] that generalize the notion of *natural gradient* by considering its casi-diagonal approximation .

State-of-The-Art ConvNet Architectures Since 2012 with the so-called *AlexNet* architecture [KSH12] that was proposed with the first GPU-based deep networks open-source implementations, a lot of work has been done on devising clever Convo-

(a) VGG Net [SZ14b]



(b) GoogLeNet [SLJ+15]



(c) Res Net [HZRS15]

FIGURE 2.8 – Some examples of state-of-the-art CNN architectures.

lutional Neural Network architectures reducing more and more the prediction error rate on several classification tasks. Simonyan et al. [SZ14b] proposed the *VGGNet* architecture (see Figure 2.8a) that extends the basic architecture of Krizhevsky up to 19 layers showing great improvement in image classification and thus motivating research toward the impact of depth of such models. At the same time, Szegedy et al. proposed the *GoogLeNet* [SLJ+15] architecture by introducing some complex layers called *inception layers*. These layers rely on applying several convolutional filters of different sizes followed by a concatenation and a pooling before applying the resulting feature maps to the next inception layer. Figure 2.8b shows a graphical illustration of such layers. Intuitively, we can say that this kind of layers allows the model to capture patterns of different natures and sizes in the same layer and learns how to select or combine them efficiently to extract more informative patterns from visual data. More recently, He et al. [HZRS15] proposed to investigate the effect of depth in such architectures and introduced a simple method to allow learning with very deep CNN architectures (up to 1000 layers). As shown in Figure 2.8c, the key idea behind their approach consists in integrating identity bridges between couples

of layers allowing the model to *jump* some layers if needed. More formally, if we consider a subset of two layers, as illustrated in Figure 2.8, and we would like this block learns some desirable underlying nonlinear mapping $\mathcal{H}(\mathbf{x})$. Then this mapping can be re-casted as $\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$ that can be seen as the identity function to which we add a nonlinear residual term $\mathcal{F}(\mathbf{x})$ with trainable parameters. Learning the module now turns into learning the parameters of this residual. This why this method has been denoted as Deep Residual Network or *ResNet*. The learning algorithm then consists in learning the non-linear residual mapping $\mathcal{F}(\mathbf{x})$ where it is easier for the learning algorithm to set the weights of this residual term to zero. Doing that, the model encounters less difficulties learning identity mappings. This has some beneficial effects such as reducing the gradient vanishing issue or automatically adapting the effective size of a deep network to the task. This network architecture now reports the best results for the ImageNet classification task and performance are now outstanding when combining both ResNet with inceptional modules [SIV16].

Convolutional Neural Network has been successfully applied to a wide range of computer vision tasks such as with character recognition [Bou06, WWCN12, SG07, EGMS14], face recognition [LGTB97, RC, DG08, Gar09, Duf08, TYRW14], generic object recognition [DDS+09, KH09, EVGW+b] or action recognition [BMW+11, BMW+12, JXYY13, SZ14a, WLG+15]. They also have been used in the context of object detection and localization such as Region-CNN (R-CNN) [GDDM14] where the authors use a region selection algorithm to activate a particular ROI in the different features map of the ConvNet. They integrated a multi-task loss in the ConvNet architecture where the first loss is a bounding box regressor and the second one is a classification (logistic) loss that is applied on the candidate image's regions. Other approaches introduced multi-scale extensions of CNN such as [FCNL12, FCNL13]. These architectures are robust to zooming by jointly analyzing different scales to take into account different sizes of spatial context around the pixels. They successfully applied their methods to semantic segmentation and pixel labelling tasks. CNNs have also been used for verification and similarity embedding with the so-called *Siameses CNN* architecture [CHL05, Koc15, ZDI+15]. Siameses CNN correspond to a particular architecture where we aim at learning a particular embedding where data assigned to the same labels are close to each other but far away from data assigned to a different label. This kind of learning algorithm is denoted as *Metric Learning* because it aims at finding a relevant similarity function between input data. One the first proposed algorithm to learn such embedding was the so-called *DrLim* (Dimensionality reduction by Learning an invariant mapping). It consists in learning two CNN models that share their weights so as to minimize the distance between vectors of the same class and maximize the distance of those belonging to different classes by adding a so-called *contrastive term* to the objective function. Recently, Zeiller et al [ZKTF10, ZF14] proposed a solution to visualize what kind of features a ConvNet is able to learn by designing a kind of *reverse* or generative

CNN architecture so-called *Deconvolutional Neural Network (DCNN)* which consists in forwarding high-level variable activations backward to the input layer. The key point of the method consists in *unpooling layers* that decide where to backpropagate information that has been previously pooled in the forward pass. Later, Zhao et al. [ZMGL15] proposed to combine CNN and DCNN into a global autoencoder [BK88, HBL+07, Ben09, VLL+10, RVM+11] framework called *What-Where Autoencoder* which consists in minimizing both a reconstruction function and a supervised objective function where the encoder (CNN) transmit to the decoder (DCNN) where the information occurred in the forward pass. This kind of architecture has the similar objective of the *Ladder Network* proposed by Rasmus et al. [RBH+15] and has the interesting property that when we do not dispose of labelled examples, the supervised objective function term can be set to zero and the learning algorithm then only turns into an unsupervised model.

2.5 Match Kernels Methods

In the previous sections, we have mainly focused our attention on methods based on learning some parametrized model of the data such as unsupervised generative models or multi-layer nonlinear models. These learning algorithms have the common property to, somehow, *"abstract"* the training data into a model that is a representative summary of them. We thus loose the track of the training samples that we do not need any more for the inference process. In this section, we study a family of methods where the training samples explicitly take part in the decision process. This category of methods corresponds to the so-called *Kernel Methods* also referred sometimes as *Memory-based Methods* [Bis06] because the associated decision function relies on similarities between the input vector and the training samples. This turns the decision process as interpolating from examples we have already seen and the learning process corresponds to learning statistical relationship between the training samples and the task to solve. In this section, we notably focus on Match Kernels methods which are at the cross road between the Kernel methods formalism and Image Matching. In the remainder of this section, we cover these computer vision and machine learning fields so as to properly introduce the Match Kernel formalism and associated state-of-the-art methods.

2.5.1 The Kernel Trick

Linear Support Vector Machines (SVM)

In [Vap99], Vapnik et al. showed that, for a linear binary classifier, minimizing the structural risk (*i.e.* the error test) relies on maximizing the margin between the training points and the separating hyper-plane learned by the model. This margin is strongly determined by the so-called *VC-Dimension* (Vapnik-Cherovenski). Maximizing this margin amounts to minimizing the structural risk and, thus, the

classification capacity of the model. Classifiers that are designed to this attempt are called max-margin classifiers and we are now going to detail Support Vector Machines (SVM) that belongs to this category of model.

In linear SVM, we consider a binary classification problem with a training set composed of samples $\mathbf{x_i} \in R^d$ with associated labels $y_i \in \{-1, 1\}$ and we look for an optimal separating hyper-plane with normal vector $\mathbf{w} \in R^d$ and bias b. The associated decision function of the classifier is defined as:

$$f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b) \tag{2.43}$$

We attempt to learn the parameters of a separating hyper-plane between positive examples and negative examples while maximizing the margin m between them and the hyper-plane. More formally, we want to learn a model such that the orthogonal projection distance of any positive sample is higher or equals to a certain margin that we will try to maximize. More formally, for any input sample $\mathbf{x_i}$, we want:

$$\frac{y_i(\mathbf{w}^T\mathbf{x_i} + b)}{||\mathbf{w}||_2} \geq \frac{m}{2} \tag{2.44}$$

This above equation correspond to the two hyper-planes shown in Figure 2.9 where we do not want any sample inside the middle region. Actually, the whole margin term that we want to maximize can be expressed as:

$$m = m^+ - m^- = \frac{\mathbf{w}^T\mathbf{x}^+ - \mathbf{w}^T\mathbf{x}^-}{||\mathbf{w}||_2} = \frac{2}{||\mathbf{w}||_2} \tag{2.45}$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 \tag{2.46}$$

This corresponds to the un-normalized margin, also called numerical margin, $\mathbf{x}^+$ and $\mathbf{x}^-$ are respectively the positive and negative sample closest to the separating hyper-plane. We can now derive from equation 2.44 and 2.46 the real margin that we want to maximize:

$$\mathbf{w}* = argmin_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||_2^2 \tag{2.47}$$

subject to Equation 2.46 which is a quadratic optimization problem with linear constraints. Thus, it is a convex problem leading to a unique global optimum. Solving this problem can be done by using a lagragian multipliers framework that can be expressed as an unconstrained optimization problem such that:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||_2^2 - \sum_i \alpha_i(y_i(\mathbf{w}^T\mathbf{x_i} + b) - 1) \tag{2.48}$$

where $L(\mathbf{w}, \boldsymbol{\alpha})$ is the lagragian that can be minimized be equating to zero its partial derivatives and $\{\alpha_i\}$ are the lagragian multipliers. This leads to:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x_i} = 0 \Leftrightarrow \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i} \tag{2.49}$$

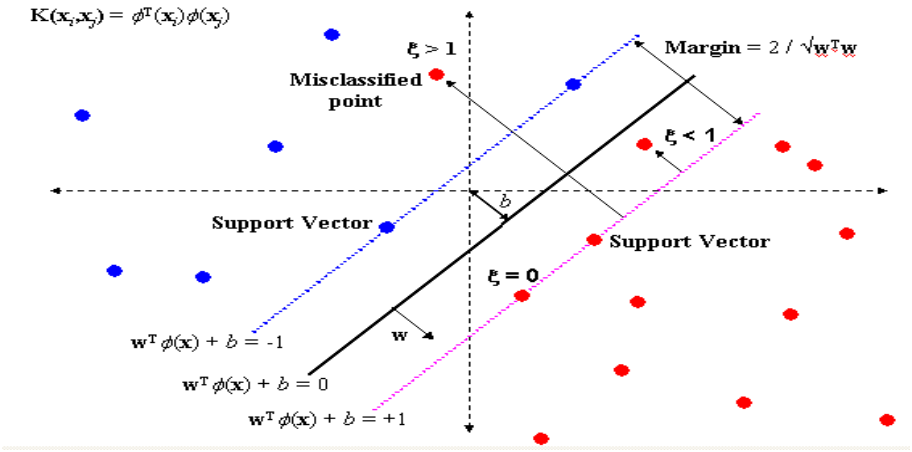


FIGURE 2.9 – Illustration of a linear max margin classifier where the aim is to find a separating hyper-plane where there is as few as possible input point inside a *margin region* delimited by two hyper-planes that are parallel to the middle one and where the distance between them, the *margin*, is maximized. These two hyper-planes are defined by particular input samples lying on them corresponding to th support vectors.

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i = 0 \tag{2.50}$$

which means that the optimal $\mathbf{w}$ corresponds to a linear combination of the input samples. This optimization is called *the primal* form of the SVM formulation. At this stage, it is important to note that the coefficients α_i will be equal to zero where the constraint $y_i(\mathbf{w}^T\mathbf{x_i} + b) > 1$ holds. This means that only the vectors lying on the hyper-planes of equation 2.46 will have coefficients α_i different from zero. The optimal $\mathbf{w}$ will thus correspond to a linear combination of those particular vectors so-called *support vectors*.

Going Non Linear with SVM by using Kernels

By integrating equation 2.49 in the minimization problem 2.48, we obtain a new lagragian to solve:

$$\min_{\vec{\alpha}} \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_i \mathbf{x_i}^T \mathbf{x_j} - \sum_i \alpha_i \tag{2.51}$$

subject to:

$$\sum_i \alpha_i y_i = 0 \tag{2.52}$$

with $\alpha_i \geq 0$. This corresponds to the *dual form* of the SVM problem that now only depends on the α_i coefficient. In a similar way, the decision function can be expressed

as:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b \tag{2.53}$$

The above equations show that the dual formulation of the SVM has the nice property to only depend on the dot products of the training samples. This allows us to design an extension of the optimal solution by considering non-linear decision functions. This can be simply done by introducing a non-linear feature mapping $\Phi : R^d \rightarrow R^k$ where d is the dimension of the original input space and k is the dimension of the new space where to embed the input data. Generally, k is larger than d such that non-linear separable problems gets linearly separable in this intermediate space where we now consider the inputs samples $\Phi(\mathbf{x_i})$. The optimal solutions of this new problem formulation can now be expressed by replacing the dot product $\mathbf{x_i}^T \mathbf{x_j}$ by $\Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})$ leading to the following new term to maximize:

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_i \Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j}) \tag{2.54}$$

The decision function becomes:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i \Phi(\mathbf{x_i})^T \Phi(\mathcal{X}) + b = \sum_i \alpha_i y_i k(\mathbf{x_i}, \mathbf{x}) + b \tag{2.55}$$

such that linear decision boundaries in the space defined by Φ are equivalent to non-linear ones in the original input space. Solving non-linear SVM now amounts to solving the linear SVM taking input samples embedded in the new space. The dot product $\Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})$ can be interpreted as a similarity operator between two vectors that can be expressed in a more generic way as a *kernel function* $k(\mathbf{x_i}, \mathbf{x_j})$ where any similarity function can be chosen to solve the SVM model under the constraint that it can be expressed as an inner product in a certain space. Mercer's theorem assures that this condition holds whenever the application $k(\mathbf{x_i}, \mathbf{x_j})$ is a positive definite form, *i.e.* its eigen values are all positive and different from zero. This has the beneficial property that if the Mercer condition is valid for a given kernel, there is no need to explicitly compute the dot product $\Phi^T \Phi$ and we even do not need to know anything about the change of variable Φ (often referred as the *explicit embedding*), to learn the associated non-linear SVM. This is known in the literature as the *Kernel Trick*. This principle makes Kernel methods powerful tools to deal with nonlinear separable problems. In particular, if we have some prior about how to design a kernel adapted to the task we want to solve (provided it satisfies the Mercer condition). The first mostly used kernels of the literature were the Polynomial kernel $(\mathbf{x_i}^T \mathbf{x_j} + c)^d$ and Gaussian Kernel or *Radial Basis Function* (RBF) kernel $\exp(-\frac{||\mathbf{x_i} - \mathbf{x_j}||^2}{2\sigma^2})$. The polynomial kernel has been designed as a kind of feature augmentation that looks for higher order similarities involving combinations of the different original features. For kernelized SVM, the decision function relies on pooling similarity scores to decide which class a test sample belongs to and the learning procedure then consists in

learning an optimal reduced subset of training samples (*i.e.* the support vectors) with corresponding weights. This is why SVM are also referred as *sparse kernel machines* [Bis06] as they considerably reduce the number of similarity operations to perform to predict the label of a new test sample. They are also referred as *memory-based* or *instance-based* learners as their learning procedure consists in remembering the training samples and how they participate to the decision function. Figure 2.10 shows an illustration of non-linear decision boundary corresponding to RBF-SVM. We see that such model can benefit from powerful discriminating abilities as they can deal with highly nonlinear data distribution. One of the main reason why non-linear SVMs work well is motivated by the Cover's theorem [Cov65] which states that the probability that P samples of dimension N are linearly separable goes to zero very quickly as P grows larger than N. The particularity of the RBF kernel is that it is a Mercer kernel whose explicit embedding maps the input vector onto feature space with an infinite number of dimensions. Indeed, it can be shown [Sha09] that for $\sigma = 1$:

$$\exp\left(-\frac{||\mathbf{x_i} - \mathbf{x_j}||^2}{2}\right) = \sum_{k=0}^{\infty} \frac{(\mathbf{x_i}^T \mathbf{x_j})^k}{k!} \exp\left(-\frac{1}{2}||\mathbf{x_i}||^2\right) \exp\left(-\frac{1}{2}||\mathbf{x_j}||^2\right) \tag{2.56}$$

The other reason why kernel methods have been widely considered relies on the fact that kernels can be also defined for other kind of mathematical objects that does not especially correspond to vectors. Particularly well known examples are strings kernels [LSST+02, LEC+04], graphs kernels [VSKB10, GLS06, Bor07] or histograms kernels such as the popular $\mathcal{X}^2$ kernel [VZ12] defined by $k_{\mathcal{X}^2}(\mathbf{x}, \mathbf{y}) = \sum_i \frac{x_i.y_i}{x_i+y_i}$ or other histogram intersection (HI) [SB91] $k_{HI}(\mathbf{x}, \mathbf{y}) = \sum_i \min(x_i, y_i)$ or the popular Generalized Histogram Intersection (GHI) kernel [BTB05a]. Other approaches consider probabilistic kernel based on generative model by defining similarity functions between distributions with respect to the underlying generative model (*e.g.* gaussian mixture, etc.). Common examples are the Kullback-Lieber Kernel (KLD Kernel) [MHV03] or the Shanon-Johnson Kernel [VZ12]. A large literature has been devoted to such kernels and we advise the interested reader to look at the paper of Chan et al. [CVM04] to have a more exhaustive overview of probabilistic kernels. Kernelized SVM has also been successfully used in sequential classification [KO16, GHP00, BK00] where some kernel functions even allow us to deal with non-fixed size of sequential vectorial data such as in [BBDBS10].

Although they have been heavily used in a lot computer vision tasks, one of the main drawbacks of Kernel SVM relies on the choice of the kernel function which is a hyper-parameter that needs to be cross-validated. But, it is very difficult to have a good prior on the metric (or the mapping function $\Phi(.)$) to be used for a given task. Secondly, as many supervised learning algorithm, they are often performed on top of a global image representations. Thus, it does not take into consideration spatial arrangement of local information occurring in the image. In the remainder of this section, we are going to investigate another research line which consists in

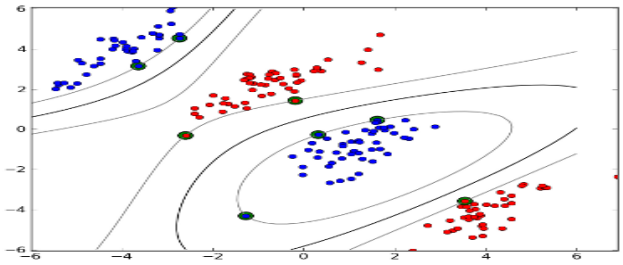


FIGURE 2.10 – Example of the non-linear decision function obtained with a SVM with a Gaussian Kernel.

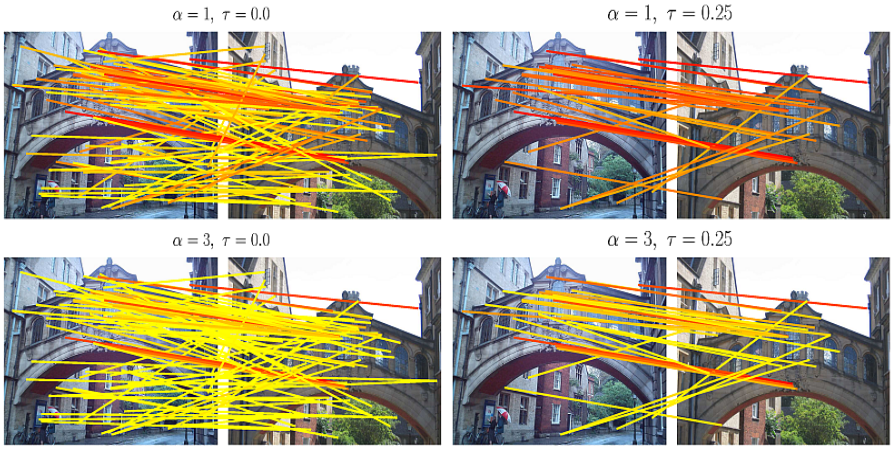


FIGURE 2.11 – Example of two images matched by their respective local features. Taken from Tolias et al. [TAJ16]

kernelizing another kind of mathematical objects : sets of features extracted from images. This gave rise to another kind of kernels called *Match Kernels*.

2.5.2 Match Kernels

Match kernels arose from the need to devise finer similarity functions between images by considering comparison between sets of localized informative and invariant patterns extracted from the images. These techniques are also often denoted as *Matching Kernels* as motivated by the kernelization of *Image Matching* methods widely considered in the computer vision community in the 90's and the 2000's. These methods aim at evaluating the similarity between two images while being robust to a lot of signal distortion such as viewpoint changes, occlusion or cluttered backgrounds. The dominant methodology of such methods considers two main steps: *i)* extracting localized appearance-based invariant representations (such as those seen in section 2.2.2) which allow robustness against lightning, translation, rotation, scaling and occlusion, *ii)* evaluating pairwise similarities between the two sets of fea-

tures (*e.g.* L_2 distance, inner product, etc.) and *iii)* aggregating these similarities through a voting strategy giving rise to a global similarity score between two images.

Figure 2.11 illustrates two images being matched from their respective local features. As we can see on the left picture of the figure, performing such raw exhaustive aggregation of pairwise matching scores is subject to suffer from a highly undesirable property. As the local similarities corresponding to interesting matches are severely outnumbered by irrelevant and noisy matches, each descriptor will, thus, be heavily prone to interact with a lot of other vectors corresponding to irrelevant matches. Thus, even if their individual similarity does not contribute much to the aggregation process, the strong imbalance between these irrelevant matches and relevant ones will give rise to the emergence of a non-negligible noise term in the global similarity score. This noise term will heavily compromise the discriminative ability of the resulting similarity score. This *interference* phenomenon is also strongly related to the so-called *burstiness* [JDS09] phenomenon which states that a given visual element will appear more time in an image than what its statistical expected across the whole dataset. One might expect this phenomenon to strongly occur in images involving highly repetitive patterns such as synthetic or natural texture patterns, or images involving localized object in highly cluttered background. The resulting effect will be that the matching scores corresponding to these repetitive artifacts will pollute the global aggregation score. Over the past decades, a lot of work has been done by the computer vision community to limit this interference effect such as, as illustrated in the right part of figure 2.11, the matching score being aggregated corresponds to relevant and discriminant information. One possible way toward removing such irrelevant matches between features consists in checking if their respective local geometric characteristics are similar (*e.g.* differences of characteristic angles or ratio between characteristic scales for SIFT descriptors) additionally to their similarity in the feature space. These kind of pairwise localized geometry post-verification do not allow to rigorously provide viewpoint invariance. For those reasons, they are often denoted as *Weak Geometry Consistency* (*WGC*) [JDS08, TFJ14] as opposed to *Strong Geometry Consistency (SGC)* that rather considers post-checking how much spatial arrangement or geometric configuration of semi-local groups of features are preserved from an image to the other. Such procedures usually estimate the best 2D affine transformation that maps the spatial positions of the first set to the spatial positions of the second set where each point of the first set is matched to one and only one point of the second set. The next step consists in removing or weighting down the outliers matching scores from the global similarity score. this is done by removing the matching scores corresponding to transformed points of the first set that are too far from their corresponding point in the second set.

Image Matching has been shown to be of great interest for a wide range of computer vision tasks such as for stereo vision [MPHG91] where some examples of applications include camera motion estimation [KLL01, KB05], object pose estima-

tion [NZSF96, Tha06] or 3D reconstruction [PKF07, PNF+08, PKVVG00, Par15]. Other widely considered applications of image matching are image or region retrieval [SMH05, JB09, JDS08] involving large scale scenario where we need to compare a query image with a large quantity of images stored in a database and retrieved the most similar ones. One can also note applications such as large scale content-based near-duplicate copy detections [JB09, PCS09]. Image matching was also extensively used in the context of image classification where the dominant methodology [HH99, VdEPV93, Wan01, BSI08, ZBMM06] is to vote for a label from similarity scores between local descriptors of the query image and the descriptors of the training images. However, such methods for image classification have the main drawback that they work well only on training sets involving a significantly high inter-class variability (*i.e.* when instances of different classes are not visually too similar and do not share a lot of common patterns) which is an undesirable property in the context of advanced classification tasks such as generic object classification [EVGW+a, EVGW+b, GHP07, DDS+09] or fine-grained image classification [WBW+11, WBW+11, KSDFF13, NZ08]. For this reason, the computer vision community abandoned over the past decades such raw image matching-based classification algorithms for generic object classification in favor to more elaborated non-linear models. Instead of relying on such deterministic voting strategies, these methods are rather based on more abstracted visual representation as coding and aggregation-based methods we have discussed earlier in section 2.3. However, as we have said in the beginning of this section and as we will see later in the remainder of this section, the success of the kernelized SVM frameworks combined with the powerful invariances of local feature schemes and image matching methods suggests that kernelizing such image matching methods is a promising research direction to address fine-grained visual classification tasks.

Raw Match Kernels

The basic principle of Match Kernels [Hau99, WCG03, Lyu05] consists in defining a similarity kernel $K(\mathcal{X}, \mathcal{Y}) : E \times E \rightarrow R$ between two images X and Y respectively described by sets of local descriptors $\mathcal{X} = \{\mathbf{x_i}\}_{i \leq |\mathcal{X}|}$ and $\mathcal{Y} = \{\mathbf{y_i}\}_{i \leq |\mathcal{Y}|}$ where $\mathbf{x_i}$'s and $\mathbf{y_i}$'s are high-dimensional feature vectors in R^d. A basic formulation of a match kernel that compares two images by their respective set of features is given by:

$$K(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\gamma(\mathcal{Y}) \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}} k(\mathbf{x}, \mathbf{y}) \tag{2.57}$$

where $k(.,.) : R^d \times R^d \rightarrow R$ is often called the *base* kernel (or local kernel) which is a similarity kernel comparing two individual localized feature vectors and $\gamma(\mathcal{X})$ is a normalization factor such that $K(\mathcal{X}, \mathcal{X}) = K(\mathcal{Y}, \mathcal{Y}) = 1$. A very simple and popular instance of Match Kernel is the normalized sum match kernel formulation [Lyu05] that considers the normalization operator $\gamma(.)$ as the inverse of the cardinal

operator |.| leading to:

$$K(\mathcal{X},\mathcal{Y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|}\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{y}\in\mathcal{Y}} k(\mathbf{x},\mathbf{y}) \tag{2.58}$$

In order to be a valid kernel, the base kernel should verify the Mercer condition such that $K(\mathcal{X},\mathcal{Y})$ satisfies it as well because it is a sum of kernels that fulfill this requirement. In other words, the local kernel has to be expressible as an inner product between $\mathbf{x}$ and $\mathbf{y}$ in some space with a particular explicit embedding $\varphi(.) : R^d \rightarrow R^D$ (with D being possibly equal to ∞) such that:

$$K(\mathcal{X},\mathcal{Y}) = \gamma(\mathcal{X})\gamma(\mathcal{Y})\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{y}\in\mathcal{Y}}\varphi(\mathbf{x})^T\varphi(\mathbf{y}) = \left(\gamma(\mathcal{X})\sum_{\mathbf{x}\in\mathcal{X}}\varphi(\mathbf{x})\right)^T\left(\gamma(\mathcal{Y})\sum_{\mathbf{y}\in\mathcal{Y}}\varphi(\mathbf{y})\right) = \Phi(\mathcal{X})^T\Phi(\mathcal{Y}) \tag{2.59}$$

where $\Phi(\mathcal{X}) = \left(\gamma(\mathcal{X})\sum_{\mathbf{x}\in\mathcal{X}}\varphi(\mathbf{x})\right) : E \rightarrow R^D$. A first drawback of the naive formulation of 2.58 is that it requires very expensive computation time because we need to aggregate every possible pairwise feature similarity which involves a computation time that grows quadratically with the number of features per image. The second main disadvantage is that this kind of exhaustive pairwise similarity aggregation is highly prone to suffer from the burstiness phenomenon that we mentioned at the beginning of this section (each descriptor may interfere with a huge number of features corresponding to noisy and irrelevant matches).

A lot of work has been devoted to design match kernels that reduce this interference problem by considering, for each descriptor of an image, to only aggregate its best matches in the other image. One of the pioneering work in this research line is the work of Wallraven et al. [WCG03] which defines the following match kernel:

$$K(\mathcal{X},\mathcal{Y}) = \frac{1}{2}\frac{1}{|\mathbf{x}|}\sum_{\mathbf{x}\in\mathcal{X}}\max_{\mathbf{y}} k(\mathbf{x},\mathbf{y}) + \frac{1}{2}\frac{1}{|\mathcal{Y}|}\sum_{\mathbf{y}\in\mathcal{Y}}\max_{\mathbf{x}} k(\mathbf{x},\mathbf{y}) \tag{2.60}$$

which is a symmetric kernel that allows us to mimic the basic one of 2.58 while removing the undesirable property that $k(\mathbf{x},\mathbf{y}) \neq 0$ for unrelated feature points. Although this match kernel formulation has been shown [Lyu05, BTB05b] not to be a positive definite kernel (hence not consistent with the Mercer condition), this matching kernel has demonstrated [CJ10] competitive performance on several classification tasks such as [FFFP06, LS03]. As noted by Lyu et al. [Lyu05], an interesting property of this kernel is that the more generic formulation of equation 2.58 can be expressed in similar way to one of equation 2.60 such that:

$$K(\mathcal{X},\mathcal{Y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|}\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{y}\in\mathcal{Y}} k(\mathbf{x},\mathbf{y})^p = \frac{1}{2}\frac{1}{|\mathcal{X}|}\sum_{\mathbf{x}\in\mathcal{X}}\frac{1}{|\mathcal{Y}|}\sum_{\mathbf{y}\in\mathcal{Y}} k(\mathbf{x},\mathbf{y})^p + \frac{1}{2}\frac{1}{|\mathcal{Y}|}\sum_{\mathbf{y}\in\mathcal{Y}}\frac{1}{|\mathbf{x}|}\sum_{\mathbf{x}\in\mathcal{X}} k(\mathbf{x},\mathbf{y})^p$$

(2.61)

which can be seen as a soft generalization of 2.60 where the parameter p allows us to integrate in the match kernel a non-linear polynomial behavior. Then, we can increase the contrast between low-valued noisy matches and relevant ones. This is another way to counterbalance the burstiness effect while providing the definite positiveness to satisfy the Mercer condition. As we can see, setting $p = 1$ and replacing the average pooling operator $\frac{1}{|E|}\sum_{\mathbf{e}\in E}(.)$ in the respective rightmost parts of two summed terms in 2.5.2 by the max pooling operator $\max_{\mathbf{e}\in E}(.)$ allows us to get back to the formulation of 2.60. Considering p as being a very large value is a more clever way to approximate the match kernel of 2.60 because as p grows to infinity the contrast between the maximum local similarity value and the other ones will be such that only the maximum value will participate to the aggregation process. Also, as demonstrated in [BTB05b], the kernel given by equation 2.60 actually corresponds to the optimal kernel $K*$ of the kernel family described by equation 2.5.2 in the sense that it is the one that maximizes the similarity between any two given sets of local features. Unfortunately, this optimality comes at the price of loosing the positive definiteness which is mainly due to the *max* terms in the equation which prevents the base kernel from being expressed as an inner product in some space. As explained by Caputo et al. [CJ10], the main issue with non-Mercer Kernels is that they do not guaranty that the SVM optimization is convex, thus the max-margin constraint might not be fulfilled, leading, theoretically, to sub-optimal error risk minimization. However, the authors build on the work of Boughorbel et al. [BTF04] to claim that Match Kernels can be statistically considered as positive definite if they fulfill the property to be diagonal dominant.

The approaches we have reviewed so far correspond to match kernels that evaluate similarities between sets of features regardless of the spatial arrangement of the individual features nor the checking that the two sets of features are geometrically or spatially consistent from an image to another. As mentioned in the previous section about image matching, such (weak or strong) geometrical consistencies would be of great interest to deal with viewpoint invariance especially in the context of fine-grained classification. Some methods [WCG03, BRF10, TFJ14, TBFJ15] considered integrating weak geometry consistency checking in the aggregation process of match kernels. This is done by designing a particular explicit feature map for the base kernel such that the inner product between such embedded descriptors corresponds to kernel of the form:

$$k(\mathbf{x},\mathbf{y}) = k_f(\mathbf{x},\mathbf{y}).k_g(\theta_{\mathbf{x}},\theta_{\mathbf{y}}) \tag{2.62}$$

which is a combination of a similarity evaluation operating in the feature space and a similarity function evaluating how much the respective intrinsic local geometric information $\theta_{\mathbf{x}}$ and $\theta_{\mathbf{y}}$ of the descriptors $\mathbf{x} \in \mathcal{X}$ and their corresponding $\mathbf{y} \in \mathcal{Y}$ are

conserved. For instance, Wallraven et al. [WCG03] define a RBF kernel between the respective position of the two local feature vectors. This implies that for a match to significantly contribute to the global similarity score, both feature vectors and their geometric information have to be consistent. It is interesting to note that if $k_f(.,.)$ and $k_g(.,.)$ are Mercer kernel, then $k(.,.)$ is a Mercer kernel as well as the image level kernel $K(.,.)$ that can then be expressed a dot product of global representation vector such as in 2.59. The main strength of the explicit formulation of such match kernels relies on the fact that the base feature map $\varphi(.) : R^d \rightarrow R^D$ explicitly embed weak geometry consistency information of the individual features such that the dot product between two global image representation $\Phi(\mathcal{X})^T\Phi(\mathcal{Y})$ replaces the complicated and computationally demanding aggregation procedure involved in equation 2.5.2. This leads to a reduction of the complexity from $\mathcal{O}(|\mathcal{X}|\,.\,|\mathcal{Y}|\,.d)$ for the explicit computation of $K(\mathcal{X},\mathcal{Y})$ to $\mathcal{O}((|\mathcal{X}|+|\mathcal{Y}|).D)$ for the computation of the inner product $\Phi(\mathcal{X})^T\Phi(\mathcal{Y})$. This is why explicit embedding is a powerful tool that is often considered while designing computationally efficient match kernels [BRF10, TFJ14, TBFJ15, BS09]. However, it is often the case that the explicit embedding corresponds to an infinite-dimensional mapping where we thus need to produce approximated explicit feature map so as to efficiently compute the inner product while conserving most of the information in the aggregation procedure. For instance, Tolias et al. [TFJ14], inspired by [BRF10], proposed a weak geometrically consistent based kernel $k_g(.,.)$ that can be expressed as a Fourier series of the form:

$$k_g(\theta_{\mathbf{x}},\theta_{\mathbf{y}}) = \sum_{n=0}^{N=\infty} \beta_n \cos(n(\theta_{\mathbf{x}}-\theta_{\mathbf{y}})) \tag{2.63}$$

where N controls the quality of the approximation of the general form given in [TFJ14]. The authors study the impact of the value of N on the retrieval performance and show that the mAP value quickly converges (about $N = 3$ provide very satisfying results). The authors then derived the corresponding explicit feature mapping $\boldsymbol{\alpha}(\theta) : R \rightarrow R^{2N+1}$:

$$\boldsymbol{\alpha}(\theta) = (\sqrt{\beta_0}, \sqrt{\beta_1}\cos(\theta), ..., \sqrt{\beta_N}\cos(N\theta), \sqrt{\beta_1}\sin(\theta), ..., \sqrt{\beta_N}\sin(N\theta))^T) \tag{2.64}$$

where it can be shown that:

$$\boldsymbol{\alpha}(\theta_{\mathbf{x}})^T\boldsymbol{\alpha}(\boldsymbol{\theta}_{\mathbf{y}}) = \beta_0 + \sum_{n=1}^{N} \beta_n(\cos(n\theta_{\mathbf{x}})\cos(n\theta_{\mathbf{y}}) + \sin(n\theta_{\mathbf{x}})\sin(n\theta_{\mathbf{y}})) \tag{2.65}$$

$$= \sum_{n=0}^{N} \beta_n \cos(n(\theta_{\mathbf{x}}-\theta_{\mathbf{y}})) \approx k_g(\theta_{\mathbf{x}},\theta_{\mathbf{y}}) \tag{2.66}$$

Then, they derived the explicit feature mapping $m(\mathbf{x},\boldsymbol{\alpha}(\theta_{\mathbf{x}}))$ of the total base kernel $k(.,.)$ as being the Kronecker product $\otimes$ of any vector $\mathbf{x} \in R^d$ and the explicit feature encoding of its corresponding angle $\theta_{\mathbf{x}}$ such that:

$$m(\mathbf{x},\boldsymbol{\alpha}(\theta_{\mathbf{x}})) = \mathbf{x}\otimes\boldsymbol{\alpha}(\theta_{\mathbf{x}}) = (x_1\boldsymbol{\alpha}(\theta_{\mathbf{x}})^T, x_2\boldsymbol{\alpha}(\theta_{\mathbf{x}})^T, ..., x_d\boldsymbol{\alpha}(\theta_{\mathbf{x}})^T)^T \tag{2.67}$$

The produced mapping $m(.,.) : R^d \times R^{2N+1} \rightarrow R^{(2N+1)d}$ now encodes both visual appearance and local geometric information of the interest points such that the inner product of two embedded interest point is high if and only if they have both visual appearance and intrinsic local geometric information that are consistent:

$$m(\mathbf{x}, \boldsymbol{\alpha}(\theta_\mathbf{x}))^T m(\mathbf{y}, \boldsymbol{\alpha}(\theta_\mathbf{y})) = (\mathbf{x} \otimes \boldsymbol{\alpha}(\theta_\mathbf{x}))^T (\mathbf{y} \otimes \boldsymbol{\alpha}(\theta_\mathbf{y})) = (\mathbf{x}^T \mathbf{y}) \otimes (\boldsymbol{\alpha}(\theta_\mathbf{x})^T \boldsymbol{\alpha}(\theta_\mathbf{y})) \tag{2.68}$$

$$= k_f(\mathbf{x}, \mathbf{y}) k_g(\theta_\mathbf{x}, \theta_\mathbf{y}) \tag{2.69}$$

We can notice that any non-linear explicit feature mapping $\varphi(.) : R^d \rightarrow R^D$ can be used instead of the linear mapping. The resulting image-level kernel $K(.,.)$ is then a Mercer kernel from which we can derive an image explicit embedding $\Phi(.) : E \rightarrow R^D$ given by:

$$\Phi(\mathcal{X}) = \gamma(\mathcal{X}) \sum_{\mathbf{x} \in \mathcal{X}} m(\varphi(\mathbf{x}), \boldsymbol{\alpha}(\theta_\mathbf{x})) \tag{2.70}$$

where the weakly geometrically consistent image similarity now relies on the dot product between the two vectorial representations of the images rather than computationally demanding pairwise aggregation of the matching scores between $\mathcal{X}$ and $\mathcal{Y}$. The authors actually showed that this method significantly outperforms other state-of-the-art image search methods [Del13, PSM10, JPD+12] on several benchmarks such as the OxfordBuilding dataset [PCI+07] and INRIA Holidays dataset [JDS08]. In particular, it provides better results than the related *Covariant VLAD* (CVLAD) [ZJG13] that also considers to integrate the dominant orientation in the vector representation but at the image level. A VLAD vector is expanded into B VLAD sub-vectors of the same size than the original ones but where local features associated with dominant angle $\theta_\mathbf{x}$ are only aggregated into the b-th VLAD sub-vector (with b corresponding to the index of the quantized value of $\theta_\mathbf{x}$). This procedure is actually not very efficient compared to the one of [TBFJ15] as they rely on testing each possible quantized rotation angle to compute the final similarity score. This increases the complexity proportionally to the number of possible angles.

Although such weakly geometrically consistent match kernels have been shown to be very competitive in the context of image search, they do not explicitly integrate strong viewpoint invariance. However, this would be a highly desired property in the context of fine-grained or instance-based classification tasks where small objects are prone to appear in different positions, scales and global orientations. Another kind of methods have been designed to integrate stronger geometric consistency in the match kernel framework. Lyu et al. [Lyu05] consider both spatial consistency and strong geometry consistency between pair-wise semi-groups of local features. The spatial consistency is performed by splitting the feature sets $\mathcal{X}$ and $\mathcal{Y}$ into sub-groups of local features $G_\mathcal{X} = \{g_i^\mathcal{X}\}_{i \leq |G_\mathcal{X}|}$ and $G_\mathcal{Y} = \{g_j^\mathcal{Y}\}_{j \leq |G_\mathcal{Y}|}$ where each sub-group $g_i^\mathcal{X}$ corresponds to the spatial neighborhood of a particular descriptor $\mathbf{x_i} \in \mathcal{X}$ and can potentially overlap with other sub-groups of $\mathcal{X}$. Figure 2.12 illustrates a particular sub-group. The authors then defined a kernel between set of sub-groups

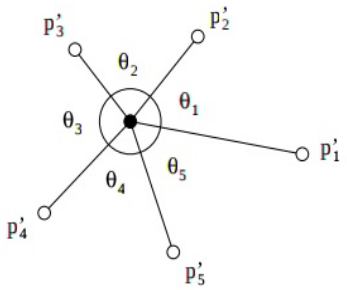


FIGURE 2.12 – Illustration of a sub-group of semi-local constrained feature sets considered in the geometrically consistent match kernel of [Lyu05]. A particular interest point is chosen as well as its spatial nearest neighbors p'_j in the image (here five neighbors are considered). We also consider the angles θ_i formed by the adjacent neighboring interest points connected to the central ones.

$K_S(.,.)$ whose base kernel have the form of the image-level kernel $K(.,.)$ of equation 2.58 which is now applied on sub-groups rather than full groups $\mathcal{X}$ and $\mathcal{Y}$. More formally, the image-level kernel has the form:

$$K_S(\mathcal{X},\mathcal{Y}) = \frac{1}{|G_\mathcal{X}|\,|G_\mathcal{Y}|} \sum_{g_i^\mathcal{X} \in G_\mathcal{X}} \sum_{g_j^\mathcal{Y} \in G_\mathcal{Y}} K(g_i^\mathcal{X}, g_j^\mathcal{Y}) \tag{2.71}$$

where $K(g_j^\mathcal{X}, g_j^\mathcal{Y})$ is the group-level kernel of equation 2.58 applied on pairwise spatially localized groups of neighboring features. The geometry consistency is achieved by combining this sub-group level kernel with a circular-shit invariant kernel that compares the respective sets of neighboring angles modeled with vectorial notations $\mathbf{\Theta}_i^\mathbf{x} \in R^{|g_i^\mathcal{X}|}$ and $\mathbf{\Theta}_j^\mathcal{Y} \in R^{|g_j^\mathcal{Y}|}$ where $|g_i^\mathcal{X}|$ and $|g_j^\mathcal{Y}|$ are constrained to equality. As illustrated in Figure 2.12 these angles are computed from the nearest neighbors of the central point of the groups. Putting it all together, this leads to the geometrically and spatially consistent match kernel formulation:

$$K_S(\mathcal{X},\mathcal{Y}) = \frac{1}{|G_\mathcal{X}|\,|G_\mathcal{Y}|} \sum_{g_i^\mathcal{X} \in G_\mathbf{x}} \sum_{g_j^\mathcal{Y} \in G_\mathcal{Y}} K_\mathcal{F}(g_i^\mathcal{X}, g_j^\mathcal{Y}).K_\mathcal{G}(\mathbf{\Theta}_i^\mathcal{X}, \mathbf{\Theta}_j^\mathcal{Y}) \tag{2.72}$$

where $K_\mathcal{F}(g_i^\mathcal{X}, g_j^\mathcal{Y})$ is the kernel on feature similarities which by:

$$K_\mathcal{F}(g_i^\mathcal{X}, g_j^\mathcal{Y}) = \frac{1}{|g_i^\mathcal{X}|\,|g_j^\mathcal{Y}|} \sum_{\mathbf{x} \in g_i^\mathcal{X}} \sum_{\mathbf{y} \in g_j^\mathcal{Y}} k(\mathbf{x},\mathbf{y})^p \tag{2.73}$$

The kernel on geometric configurations have the form:

$$K_\mathcal{G}(\mathbf{\Theta}_i^\mathcal{X}, \mathbf{\Theta}_j^\mathcal{Y}) = \sum_{l=0}^{n-1} \left((\mathbf{\Theta}_i^\mathcal{X})^T (c(\mathbf{\Theta}_j^\mathcal{Y}, l))\right)^p \tag{2.74}$$

where $l < n$ and $c(\mathbf{\Theta}_j^{\mathcal{Y}}, l) : R^n \times N^+ \rightarrow R^n$ is an operator producing the circular permuted version of $\mathbf{\Theta}_j^{\mathcal{Y}}$ where its original l-th component is set to be the first one in the new vector. The vector of angles of the second subset is then permuted in a circular way such that the angle vectors of the first sub-group is compared to the l circular-shifted versions of the vector of angles of the second sub-group. As for the feature kernel, the parameter p increase the contrast between the different aggregated terms such that the circular permutation of $\mathbf{\Theta}_j^{\mathcal{Y}}$ maximizing its similarity with $\mathbf{\Theta}_i^{\mathcal{X}}$ will be approximately the only one that will contribute to the global matching score. Although such methods rely on both spatial consistency and group-wise rotation invariance, the authors do not define an explicit embedding such as in [BS09, TBFJ15] which severely reduces the computation efficiency of such kernels. This is especially true when considering a lot of sub-groups of descriptors per image as one would expect in a fine-grained classification context.

Another popular research direction toward integrating spatial consistency in match kernel concerns graph-matching methods [SKH08, LH05, DJP11, HH99, FH05, BBM05]. In particular, Duchene et al. [DJP11], proposed a graph-matching kernel which models the images as graphs whose nodes correspond to a dense set of regions associated with a spatial grid and edges model their respective spatial adjacency. The problem of image matching is set as a problem of graph-matching where we want to optimize an energy function maximizing pairwise node assignment while conserving the local spatial adjacency of group of nodes from a graph to another. Although such approaches theoretically provide strong geometry-aware similarity kernel between two images, they are not as efficient as [TFJ14, TBFJ15] as they do not provide finite dimensional explicit feature maps. They require an exhaustive inference-based comparison between every pair of images to build the Graam matrix.

Discussion on raw match kernels Although the generalized match kernel formulation [Lyu05] of equation 2.5.2 satisfies the Mercer condition and has been shown to limit the burstiness effect by increasing the contrast between high and low matching values, the required computation time can drastically increase when considering a lot training images (each described by a high number of local features). Indeed, each image-to-image similarity would require comparing pairwise features similarities leading to a quadratic complexity that could not be used in a context of fine-grained classification where we often need to describe the images in a dense fashion with high numbers of local features. In contrast, raw matching kernels such as the one defined by equation 2.60 rather considers only keeping the best match of each descriptor from one image to another. This explicitly limits the burstiness phenomenon. Unfortunately, this causes the Mercer condition not be satisfied anymore because of the *max* terms involved in the equation which does not guaranty the margin maximization and the convergence of the SVM. Moreover, although some techniques [TFJ14, TBFJ15] provide powerful efficient formulation of weakly geometrically consistent match kernel, integrating strong geometry, *e.g.* involving

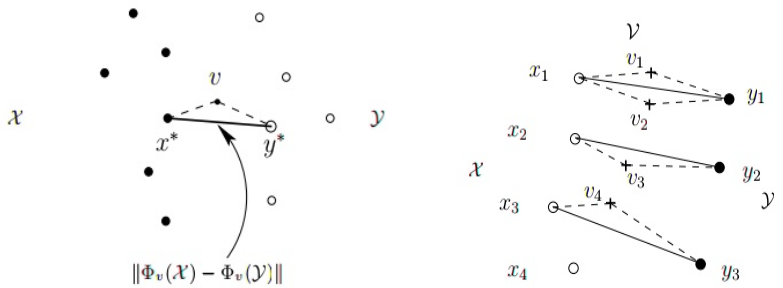


FIGURE 2.13 – Illustration of the main idea of IMK. Left: Visualization of how two local features from two set to be compared a selected to be matched with respect to a given virtual feature. Right: Example of the while computation of the similarity value between two visible set of features. For each virtual features, two features are selected and compared according to the process described on the left figure. Then every matching value are aggregated into a global matching score. Figure taken from [BTB05b].

RANSAC-like algorithms or any other registration algorithms, is simply not possible for the same reason. Despite this, what we aim to do in a context of fine-grained classification is to discriminate the different objects from finely localized details often involving particular spatial arrangement of local features. It then seems of great interest to find a match kernel formulation that would embed such strong geometry in an efficient way while benefiting from the powerful discrimination ability of the kernelized SVM framework at the same time.

Intermediate Match Kernels

Over the 2000's, Image Matching methods was neglected in favor to the upcoming of aggregation-based approaches seen in section 2.3. Approaches such as BoVW-based representations became the dominant approach in a lot of computer vision tasks due to its simplicity and its powerful discriminating abilities when combined with spatially consistency tools [JDS11, CMK03, RDGM10] or other tricks to reduce quantization errors (see section 2.3.1). However, to alleviate the definite positiveness issue induced by former methods, another research line has been considered in the literature which consists in providing a kind of intermediate feature distance between two local descriptors. Two descriptors are considered as similar if both of them are close to the same feature or group of features of an intermediate feature set (that have possibly nothing to do with the two feature sets that are being compared).

One of the most popular pioneering work in this research direction is the so-called *Intermediate Match Kernel* (IMK) proposed by Boughorbel et al. [BTB05b].

The authors proposed to introduce a set of virtual local features $\mathcal{V}$ such that each local feature $\mathbf{v} \in \mathcal{V}$ is used to select the pair of local features from $\mathcal{X}$ and $\mathcal{Y}$ to be compared with a given local kernel and pooled in the aggregation process. More concretely, for each descriptor $\mathbf{v} \in \mathcal{V}$, we compute the respective values of $\Phi_{\mathbf{v}}(\mathcal{X})$ and $\Phi_{\mathbf{v}}(\mathcal{Y})$ where $\Phi_{\mathbf{v}}(.)E \rightarrow R^d$ is a mapping function associated to particular *virtual* feature $\mathbf{v}$ such that:

$$\Phi_{\mathbf{v}}(\mathcal{X}) = \mathbf{x}\star = arg\min_{\mathbf{x}\in\mathcal{X}} ||\mathbf{x} - \mathbf{v}|| \tag{2.75}$$

and the final Intermediate Matching Kernel is defined as follow:

$$K_{\mathcal{V}}(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{v}\in\mathcal{V}} e^{\frac{1}{2\sigma^2}||\Phi_{\mathbf{v}}(\mathcal{X})-\Phi_{\mathbf{v}}(\mathcal{Y})||^2} \tag{2.76}$$

It is easy to prove that $K_{\mathcal{V}}$ is a Mercer kernel as it is a sum of RFB kernels between the intermediate mapping $\Phi_{\mathbf{v}}(\mathcal{X})$ and $\Phi_{\mathbf{v}}(\mathcal{Y})$ such that the local kernel $K_{\mathbf{v}}$ associated to a particular virtual feature $\mathbf{v} \in \mathcal{V}$ can be expressed as:

$$K_{\mathbf{v}}(\mathcal{X}, \mathcal{Y}) = K_{RBF}(\Phi_{\mathbf{v}}(\mathcal{X}), \Phi_{\mathbf{v}}(\mathcal{Y})) = \Phi_{RBF}(\Phi_{\mathbf{v}}((\mathcal{X}))^T \Phi_{RBF}(\Phi_{\mathbf{v}}(\mathcal{Y})) \tag{2.77}$$

which satisfies the Mercer condition because $\Phi_{\mathbf{v}}(\mathcal{X})$ and $\Phi_{\mathbf{v}}(\mathcal{Y})$ do not depend on each other. As illustrated on Figure 2.13, the intermediate features allow to select local features to be matched rather than comparing them in a pairwise exhaustive fashion. Intermediate match kernels then seem to be a powerful alternative to design match kernels while assuring the positive definiteness property thus benefiting of powerful generalization abilities of SVM. The other noticeable strength of such kernels is that the virtual features can be chosen to be located in highly informative region of the space. In [BTB05b], the authors chose to set $\mathcal{V}$ as an abstraction of the training features by learning a visual vocabulary on top of them thanks to an unsupervised clustering algorithm. More concretely, they use a Fuzzy C-Means algorithms [Bez81] that can be seen as soft assignment version of the K-means algorithm.

As noticed by the authors, this suggests alternative ways of designing match kernels where we could extend this principle to any clustering or coding algorithms. For instance, [BS09] noticed that the linear product between the BoVW representation [SZ03] of two images is a specific case of match kernel consisting in pooling the matches in the k-means clusters and we can thus rewrite it as follows:

$$K_{BOW}(\mathcal{X}, \mathcal{Y}) = \Phi_{BOW}(\mathcal{X})^T \Phi_{BOW}(\mathcal{Y}) = \sum_{\mathbf{v}} \sum_{\mathbf{x},\mathbf{y}} \delta_{\mathbf{v}}(\mathbf{x}, \mathbf{y}) \tag{2.78}$$

where

$$\delta_{\mathbf{v}}(\mathbf{x}, \mathbf{y}) = \delta_{\mathbf{v}}(\mathbf{x})\delta_{\mathbf{v}}(\mathbf{y}) = \begin{cases} 1 & Q(\mathbf{x}) = Q(\mathbf{y}) = \mathbf{v} \\ 0 & else \end{cases}$$

Thus, this is a positive definite local kernel measuring the similarity between two vectors as equal to 1 if they belong to the same Voronoi cell and equal to 0 else.

Learning a linear SVM on top of a BoVW explicit embedding is then equivalent to learning a kernelized SVM on top of two feature sets by considering an intermediate feature set being equal to the visual vocabulary learned by a K-means clustering algorithm. The main differences between this match kernel formulation and the one of [BTB05b] relies on the use a Fuzzy C-Means clustering, the RBF kernel between pairwise intermediate mapping is replaced by a linear kernel. Finally, the $arg\min$ operator in equation 2.75 is replaced by a sum pooling operator over the features in $\mathcal{X}$ and $\mathcal{Y}$ hard assigned to $\mathbf{v}$. Then, the Bag-of-Visual-Word seems to be an interesting alternative way to design efficient match kernels over two sets of points as they can be expressed with a finite-dimensional explicit embedding while satisfying the Mercer condition. Moreover, such intermediate match kernel also benefit from powerful abstraction-based models where the virtual features (here visual words) can be considered as a set of latent variables explaining the visible data. Two features are matched and pooled if they somehow *share* or are explained by the same latent variables.

As noticed in [BS09], although this allows us to provide much more informative matching score than aggregating raw pairwise similarities of local features, the BoVW match kernel is highly prone to lose a lot of information of the original matches due to quantization error involved in the local kernels $\delta_{\mathbf{v}}(\mathbf{x}, \mathbf{y})$. The authors then advocate the use of continuous local kernel function such as the one depicted in equation 2.58 while benefiting of an intermediate vocabulary and an explicit embedding formulation to provide an *Efficient Match Kernel (EMK)*. To this aim, the authors proposed a way to build an approximate feature mapping $\varphi(\mathbf{x}) : R^d \rightarrow R^D$ based on low dimensional projection on a learned dictionary such that their inner product approximates a given local kernel function. More concretely, they consider the original explicit feature mapping $\psi(\mathbf{x}) : R^d \rightarrow R^{D'}$ (possibly infinite-dimensional) induced by the target local kernel $k(\mathbf{x}, \mathbf{y})$ and look for a linear approximation of it such that:

$$k(\mathbf{x}, \mathbf{y}) = \psi(\mathbf{x})^T \psi(\mathbf{y}) \approx (\mathbf{H}\mathbf{v}_{\mathbf{x}})^T (\mathbf{H}\mathbf{v}_{\mathbf{y}}) \tag{2.79}$$

where $\mathbf{H}$ is a codebook matrix given by $[\psi(\mathbf{z_1})^T, ..., \psi(\mathbf{z_D})^T]^T$ corresponding to the non-linearly embedded codebook vectors. $\mathbf{v_x}$ is a low dimensional code vector and the problem now amounts to minimizing the reconstruction objective function:

$$J(\mathbf{H}, \mathbf{v}_{\mathbf{x}}) = \sum_{\mathbf{x}} ||\psi(\mathbf{x}) - \mathbf{H}\mathbf{v}_{\mathbf{x}}||_2^2 \tag{2.80}$$

This is similar to the typical minimization term in coding-based methods where it is possible to generalize to any other regularized coding-scheme such as sparse coding [GTC10]. This is not an easy problem at first glance because the target explicit mapping $\psi(.)$ to be approximated is prone to be infinite-dimensional (*e.g.* if we want to approximate the RBF kernel) and we thus need to operate with the kernel trick to

properly minimize equation 2.79 to find $\mathbf{H}$ and the code vectors $\mathbf{v_x}$. To this aim, the authors proposed a modified version of Kernel PCA (KPCA) [Bül, KFS05] based on stochastic gradient descent optimization to derive the codebook matrix $\mathbf{H}$. Such methods are powerful tools that allow to perform spectral decomposition using the kernel trick such that a great proportion of the energy will be concentrated in a few eigen directions in the kernelized space. In this case, this will allow to produce low dimensional code vector that will well approximate the original targeted local kernel. Once the codebook vectors have been learned the low dimensional approximation of the target explicit embedding can be easily computed by minimizing equation 2.80 with respect to $\mathbf{v_x}$. This is a simple linear regression problem applied to a convex function leading to the optimal code:

$$\mathbf{v_x}* = (\mathbf{H}^T\mathbf{H})^{-1}(\mathbf{H}^T\psi(\mathbf{x})) \tag{2.81}$$

The targeted local kernel can then be approximated as:

$$k(\mathbf{x},\mathbf{y}) = (\mathbf{H}\mathbf{v_x}*)^T(\mathbf{H}\mathbf{v_y}*) = k_Z(\mathbf{x})^T(\mathbf{H}^T\mathbf{H})^{-1}k_Z(\mathbf{y}) = k_Z(\mathbf{x})^T(\mathbf{G}^T\mathbf{G})k_Z(\mathbf{y}) \tag{2.82}$$

where $k_Z(\mathbf{x}) \approx \mathbf{H}^T\psi(\mathbf{x}) \in R^D$ is a similarity vector whose component are given by $k_Z(\mathbf{x})_i = k(\mathbf{x},\mathbf{z_i}) = \psi(\mathbf{x})^T\psi(\mathbf{z_i})$. $\mathbf{G}$ is such that it can be considered as the square root matrix of the inverse of matrix $(\mathbf{H}^T\mathbf{H})$ which does not need to be explicitly computed as a product as it simply corresponds to the Graam matrix K_Z whose term are of the form $K_Z^{ij} = k(\mathbf{z_i},\mathbf{z_j})$. Finally, we can derive the approximated image-level explicit embedding of the form:

$$\Phi(\mathcal{X}) = \frac{1}{|\mathcal{X}|}\sum_{\mathbf{x}\in\mathcal{X}}\varphi(\mathbf{x}) = \frac{1}{|\mathcal{X}|}\mathbf{G}\sum_{\mathbf{x}\in\mathcal{X}}k_Z(\mathbf{x}) \tag{2.83}$$

that can be seen as a kernelized and non-quantized version of a Bag-of-Word representation where $k_Z(\mathbf{x})$ is soft-assignment representation with respect to a codebook learned in a non-linear kernelized space (rather than a hard-assignment representation $Q(\mathbf{x}) = \mathbf{D}^T\mathbf{h}$ with respect to a codebook produced by a traditional clustering algorithm in the raw input space). This makes such representations more continuous and potentially limits the number of irrelevant matches due to error quantization.

During past decades, a lot of methods has been proposed to reduce such quantization error involved in intermediate match kernels. We could cite as examples all the soft-assignments extension of the Bag-of-Words model [VGGVS08, LWL11] that we mentioned earlier in section 2.3.1. Another popular intermediate match kernel formulation is the Pyramid Match Kernel [GD05] proposed by Grauman and Darrell, which is an algorithm extending the BoVW model by calculating multi-resolution histograms in the feature space. To this aim, the space of descriptors is quantized using increasing cell size allowing certain resolutions to group descriptors that would not have been grouped otherwise. This makes it less sensitive to quantization errors. The global similarity between two set of features is obtained similarly

to the match kernel formulation of the Bag-of-Words where pairwise feature points that belong to the same bin at a particular resolution l are matched together and pooled in the aggregation process. The global kernel is then obtained by summing the kernels of each resolution. As for the BoVW model, this match kernel has an explicit embedding formulation where each set of points is represented by the concatenation of histograms of features (with hard assignment like BoVW) obtained for the different resolutions. The Histogram Intersection kernel is then used to compute the image-level similarity kernel. The main drawbacks of such methods is that we lose the knowledge about the underlying structure of the input space that could be modeled be any generative algorithm such as the unsupervised clustering algorithm used in BoVW.

Spatial Pyramid Match Kernel (SPMK) [LSP06, YYGH09] that we mentioned earlier in section 2.3 have actually been originally formulated [LSP06] in a match kernel perspective as motivated by the extension of the Pyramid Match Kernel considering multi-resolutional histograms in the image space rather than in the feature space. The match kernel formulation simply consists in matching the features assigned to the same visual word and belonging to the same spatial bin at a particular resolution. The global kernel is, then, obtained by pooling over the kernels obtained for different spatial resolutions. As for Bag-Of-Visual-Words, the explicit embedding formulation is used and a noticeable particularity is that considering a SPMK for only one resolution equal to 0 is equivalent to the Bag-of-Word formulation. Integrating such geometry consistency in intermediate match kernel can also be considered in the same way of Tolias et al. [TFJ14, TBFJ15] as discussed earlier in the previous subsection. In particular, as we have seen in equation 2.70, it is possible to consider any explicit feature embedding $\varphi(.) : R^d \rightarrow R^D$ (such as the Fisher encoding [PSM10], VLAD [JPD+12] or Sparse Coding based methods [WYY+10]) to efficiently combine properties of efficient intermediate match kernel and weak geometry consistency. Actually, the authors showed that using the VLAD embedding significantly increase the retrieval performance. Indeed, as the Bag-of-Word model, Fisher Vector and VLAD can have an intermediate match kernel interpretation where it is originally derived from the Fisher similarity metric also known as the *Fisher Kernel* [JH+99] defined between two feature vectors:

$$k_{FV}(\mathbf{x}, \mathbf{y}) = (\nabla_\theta \log p(\mathbf{x}|\theta))^T \mathbf{F}^{-1} (\nabla_\theta \log p(\mathbf{y}|\theta)) \tag{2.84}$$

The vector for a given local feature vector is then given by equations 2.29, 2.30 and 2.31 in section 2.3 and corresponds to the explicit embedding from the base Fisher kernel such as:

$$\varphi_{FV}(\mathbf{x}) = \mathbf{L} \nabla_\theta \log p(\mathbf{x}|\theta) \tag{2.85}$$

leading to the following Fisher match kernel formulation:

$$K_{FV}(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}} k_{FV}(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}} \varphi_{FV}(\mathbf{x})^T \varphi_{FV}(\mathbf{y}) \tag{2.86}$$

As noticed by the very inspiring paper of Tolias et al. [TAJ16], similar analogy with intermediate match kernels can be made with VLAD vectors [JPD+12] and Hamming Embedding [JDS08][WZNM14, JBJG12]. In this paper, they proposed a particular kind of match kernel so-called *Aggregating Selective Match Kernel* (ASKM). It consists in combining the attractive properties of both matching-based approaches (that provides selectivity) and the aggregation-based representation such as the VLAD representation that provides compact global representation suited for classification and retrieval tasks. To do this, the authors consider two kinds of new kernels: a non-aggregated *Selective Match Kernel* (SMK) and an *Aggregated Selective Match Kernel* (ASMK). The *Selective Match Kernel* between two image is defined as:

$$SMK(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma(\mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{y})) \tag{2.87}$$

Where $\sigma(.) : R \rightarrow R$ is a selectivity function, $\mathbf{x}_c$ and $\mathbf{x}_c$ are respectively the set of descriptors of the two images to be compared that has been quantized by the c-th word of the visual dictionary and $\mathbf{r}(\mathbf{y})$ and $\mathbf{r}(\mathbf{y})$ respectively correspond to the normalized residual vector of local descriptors $\mathbf{x}$ and $\mathbf{y}$ with respect to their quantized value. The role of this non-linear function is to provide suitable normalization of the matching scores so as to reduce the impact of irrelevant matches. The authors actually chose to use a power normalization selectivity function such as those used as post processing of Fisher Vector embedding that we have described in section 2.3. The Aggregated Selective Match Kernel rather consists in previously aggregating the local features of the respective images into a normalized VLAD representation and then applying the dot product between these representations followed by the same selectivity function than for SMK:

$$ASMK(\mathcal{X}_c, \mathcal{Y}_c) = \sigma(\Phi_{VLAD}(\mathcal{X}_c)^T \Phi_{VLAD}(\mathcal{Y}_c)) \tag{2.88}$$

where $\Phi_{VLAD}()$ is the normalized VLAD embedding described in section 2.3. In their experiments, the authors show that this new kind of kernel exhibits superior performance with respect to state-of-the-art methods for both retrieval and place recognition tasks.

Memory-based Match Kernels

In this section, we are going to discuss a particular kind of kernel models where the virtual set no longer corresponds to an abstraction of some intermediate set but rather keeps the original feature vectors of this intermediate set. As the intermediate set corresponds to *real* and previously observed examples, these methods are often denoted [Bis06] as *memory-based* models or *instance-based* models because the decision function consists in recognizing an object by the combination of instances we have already encountered.

The first kind of techniques corresponding to this category are strongly related to matching-based K-NN classifier that we mentioned earlier in this section when introducing image matching principles. These techniques are primarily aimed at retrieving instances of a given query object in an unsupervised way but any of them can be used for classification purposes when the search is performed on a labelled set of pictures (typically by voting on the top-K retrieved images or through any other instance-based classifier). One of the simplest K-NN-based matching kernel method is the one proposed by Boiman et al. [BSI08], who use a Nearest Neighbor (NN) classifier as a nonparametric model that does not require any training phase. This is the so-called *Naive Bayesian Nearest Neighbor* (NBNN) classifier where the authors cast the NN-based classifiers as a particular cases of Bayesian Learning leading to a very simple classification scheme. Considering a set $\mathcal{Y}$ of descriptors extracted from training images associated with labels $c \in \mathcal{C}$, the predicted label $\hat{c}$ of a test image represented by a set of local feature vectors $\mathbf{x}$ is given by:

$$\hat{c} = arg\min_{c\in\mathcal{C}} d^c_{\mathbf{x}} = arg\min_{c\in\mathcal{C}} \sum_{\mathbf{x}\in\mathcal{X}} ||\mathbf{x} - NN^c(\mathbf{x})||_2^2 \tag{2.89}$$

where $NN^c(\mathbf{x})$ correspond to the nearest neighbor of descriptors $\mathbf{x}$ among $\mathcal{Y}^c$ which is the subset of $\mathcal{Y}$ containing the descriptors extracted from images of class c. This is equivalent to finding the class with the minimum image-to-class distance $d^c_{\mathbf{x}}$. The authors showed that this simple formulation is an approximation of a Naive Bayes classifier with uniform prior distribution over the classes such as:

$$\hat{c} = arg\max_c \log\left(p(c|\mathbf{x})\right) = arg\max_c \log\left(p(\mathbf{x}|c) = arg\max_c \log\left(\prod_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}|c)\right) = arg\max_c \sum_{\mathbf{x}\in\mathcal{X}} \log\left(p(\mathbf{x}|c)\right)\right. \tag{2.90}$$

which holds if we assume the independence between the descriptors of $\mathbf{x}$. Considering a Parzen Window estimator for $p(\mathbf{x}|c) = \frac{1}{|\mathcal{Y}^c|} \sum_{\mathbf{y^c}\in\mathcal{Y}^c} e^{-\left(\frac{||\mathbf{x}-\mathbf{y^c}||_2^2}{2\sigma^2}\right)}$, and considering the approximation that the nearest neighbor term $\hat{\mathbf{y^c}} = arg\min_{\mathbf{y^c}} ||\mathbf{x} - \mathbf{y^c}||_2^2 = NN^c(\mathbf{x})$ is the one that contributes the most to the estimator, we get back the approximation of equation 2.89. More recently, Tuytelaars et al. proposed a kernelized version of the NBNN model and demonstrate some significant gain on several benchmark compared to the original method. Instead of only taking the class with the minimum distance to the test image, they keep all the distances values from the test image to the different classes and produce a vectorized representation of an image by concatenating these distances. The final image representation is then given by $\Phi(\mathcal{X}) = \left[d^1_{\mathbf{x}}, d^2_{\mathbf{x}}, ..., d^{|\mathcal{C}|}_{\mathbf{x}}\right]^T \in R^{|\mathcal{C}|}$. This gives rise to an intermediate match kernel of the form:

$$K(\mathcal{X},\mathcal{Y}) = \Phi(\mathcal{X})^T\Phi(\mathcal{Y}) = \sum_{c\in\mathcal{C}}\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{y}\in\mathcal{Y}} d^c_{\mathbf{x}}.d^c_{\mathbf{y}} \tag{2.91}$$

Later, McCann et al. [ML12] proposed another version of the NBNN kernel to reduce the computation cost while not loosing classification accuracy. Their proposed method, so-called *Local NBNN*, consists in computing for each descriptors $\mathbf{x} \in \mathcal{X}$ its k nearest neighbors in the full set of descriptors $\mathcal{Y}$ and then recompute on top of the set of retrieved descriptors the original NBNN scores $d^c_{\mathbf{x}}$ before taking the class with minimum distance. This method is different from NBNN in the fact that some classes may not be represented in the retrieved descriptors in which case their distance-to-class value $d^c_{\mathbf{x}}$ is set to zero. The authors showed that this method allows us to reduce the computation time by scaling up the number of classes to be managed. Indeed, using approximate knn search schemes in the context of raw NBNN is prohibitive for large number of classes because such algorithms are often linear in the number of samples which implies than performing knn search procedures on $|\mathcal{C}|$ subsets is more expensive than performing it on the whole set of descriptors even if the total number of descriptors that is considered is the same. Moreover, this method somehow allows us to reduce classification noise as we give more confidence to the classes that are most represented in the direct neighborhood of the local descriptor.

Another related method was proposed by Krapac et al. [KPFJ14] which is based on a feature-wise prototype selection approach where a distance-adaptive prototype is trained in a supervised way for each local descriptors of the training set. Each prototype corresponds to a set of balls with different radii and each ball is assigned to a particular class-specific precision score (computed for the class of the training descriptors prototype's balls are centered on) roughly corresponding an Average Precision (AP)-like score computing on the training feature vectors that falls into the ball, *i.e.* if all the descriptors inside the ball are of the same class than the center feature of the prototype, then the ball's score will be one. The decision function then coarsely consists in finding, for each feature vector of the test image, the prototype's balls they fall into. Each ball is affected to a score given by the sum of all descriptors that falls into it weighted by the AP-like scores of the corresponding balls. Each prototype score is obtained by pooling together its ball's scores and the score for the class is obtained by pooling the scores of the prototypes corresponding to that class. Although some of the previously seen methods are not directly related to match kernels, they still correspond to categories of methods consisting in aggregating the similarities of the test image's features with respect to all the descriptors of the training set. Some of these methods also rely on an approximated version of their respective main formulation by considering scaling up their pooling procedure with approximate k-nearest neighbor techniques. This is also the case of methods that are more related to intermediate match kernels denoted as *similarity-based representations* [LV09, JBJG12]. They consist in representing an image by concatenating its similarities to the other training images into a global representation. This is a particular case of intermediate match kernel but more related to memory-based match kernels as the intermediate set is composed of the training images. In [JBJG12], the authors define a similarity-based embedding by pooling local similarity scores

at image level by summing Hamming Embedding based similarities between the local features of the query vector and those of images of the training database. This gives, for each image of the database, a similarity score with respect to the query. The obtained scores are then aggregated by concatenating them which gives rise to N-dimensional representation where N is the number of images in the training database. More formally, the image representation is given by:

$$\Phi_{Hamming}(\mathcal{X}) = \sum_{i=1}^{N} HE(\mathcal{X}, \mathcal{Z}_i)\vec{e_i} \tag{2.92}$$

where $\mathcal{X}$ and $\mathcal{Z}_i$ are respectively the descriptors set of the query image and the descriptors set of the i-th image of the training database and $HE(.)$ is the Hamming Embedding similarity proposed in [JDS08]. This representation is then power normalized and L_2 normalized as it has been proved to be an efficient way to deal with the burstiness phenomenon for both classification and retrieval performance.

2.6 Discussion

In this chapter, we have reviewed the most popular image representation schemes of the literature. We can group these approaches into three main categories. First, we have presented coding schemes and aggregation methods that consist in learning in an unsupervised way how to extract relevant information from a training set of local vectors. Coding schemes learn a set of basis vectors whose linear combination maximize the reconstruction of the original samples. Most of the time, this procedure is applied on top of a handcrafted feature extraction scheme (such as SIFT). The resulting encoded low-level features are then aggregated giving rise to a global vectorial representation that characterizes the whole image. The second category of methods corresponds to deep learning approaches that consist in learning a multilayer and progressive embedding with spatial aggregation between the layers so as to be adapted to the targeted task. The mostly used approach in the literature is the Convolutional Neural Network that now exhibits astounding results on a wide range of computer vision tasks. The main disadvantage of this method lies in the fact that it needs a very large number of training examples or, at least, a lot of training samples of a related task. Finally, the third category of methods was kernel methods which consist in a decision function based on the similarity between the training images. There exists a wide range of kernel methods and we have particularly focused on the study of Match Kernels that rely on the pooling of local similarity scores of matched local features between the images.

The major drawback of aggregation-based methods and Match Kernels is the loss of spatial information in the aggregating process and the fact that they ignore the local geometric consistency between local descriptors. However, this would be of great interest in a context of fine-grained classification where semantically different objects

can be composed of similar sets of lower-level features that respectively appear with different spatial configurations. Although some extensions of aggregation-based and Match Kernel methods proposed some spatial binning techniques to integrate partial geometric information into the representation space, these techniques often rely on encoding the position of the local descriptors in the image that we want to represent. This does not guarantee invariance from an image to another. For instance, this is the case of the popular Spatial Pyramid Match Kernel where local descriptors are aggregated inside several regions of the query image or the Spatial Coordinate Coding (SCC) method test each local descriptor is augmented with additional components corresponding to their position in the query image before aggregating them in the final global representation. Although these methods are based on local descriptors, the lack of spatially localized information implies that a lot of details are lost in the final image representation. This also implies that these representations will not be able to recognize small objects that can appear in possibly highly cluttered background.

The major contribution that we propose in this work consists in solving the lack of spatially localized details of the above-mentioned methods by making use of large scale and geometrically consistent content-based retrieval methods. In the following chapter, we present a first contribution to design a classification system that relies on a *Matching-based Spatially Consistent K Nearest Neighbors classifier*. In chapter 4, we present a more general Match-Kernel formalism and its corresponding explicit embedding. We show that applying supervised linear classification on top of it allows us to learn good combinations of geometrically consistent patterns resulting in significant gains compared to our baseline of Chapter 3.

Chapitre 3

Toward Fine-Grained Classification through Spatially Consistent Approximate Nearest Neighbors

Contents

3.1 Introduction

In the previous chapter, we saw that most state-of-art methods hardly handle small objects as well as the geometric relationship between the different parts of these objects whereas this is particularly important in the scope of fine-grained visual classification. The contribution presented in this chapter is a first attempt to address this issue by considering an instance-based classification model using localized sets of feature vectors and strong geometry to measure the similarity between

these sets. Such model implies a huge volume of local descriptors in the training database. This leads to a non-negligible scalability issue if we want to perform exhaustive similarity between each feature of the query images and the ones of the training dataset. In this work, we propose to alleviate this scalability issue with the use of state-of-the-art large scale approximate nearest neighbor search techniques.

Figure 3.1 illustrates an overview of our instance-based geometrically-consistent classification scheme which can be summarized as follows: We consider a training set S of $|S|$ pictures weakly annotated with one or several labels among $|C|$ classes (corresponding to entities presumably recognizable in the picture through specific small visual patterns such as logos, monuments, products, etc.). Each picture is described by a set of local descriptors forming a reference dataset $\mathcal{Z}$ of N local features $\mathbf{z}_i$. Similarly, each query image I_q is represented by densely extracted local descriptors that are searched independently in the reference set using an efficient approximate K-NN search scheme. A local strong geometry consistency checking is then performed at every local feature position using a newly introduced sliding RANSAC procedure. The resulting lists of checked patches are then back-propagated in the query image and merged in order to produce pixel-wise saliency maps for each of the retrieved label. A voting-based classifier is finally derived from the class-specific saliency maps through a max-pooling strategy.

The remainder of this chapter will be organized as follows: First, we will review the state-of-the-art methods for approximate nearest neighbor search to justify which one we use to leverage the scalability issue involved in our matching based knn classifier. We will then give the details of our classification algorithm designed on top of this approximate knn search technique. Thereafter, we will present some experiments that we led on different benchmarks in terms of classification performance and object localization. Finally, we will interpret these results and highlight weaknesses of this approach so as to introduce the contribution of the next chapter of this thesis.

3.2 Scalability issue and hash-based approximate K-NN search

As the classification scheme is based on the pooling of local similarity scores between the test image and local regions of training images, we need a system that first retrieve similar local features of the training images with respect to those of the test image. Here we consider the use of an efficient K-NN search scheme that will allow retrieving the k nearest neighbors of each local descriptor. In the experiments presented in this work, we consider SIFT descriptors [Low04] as local handcrafted features extracted on image location detected by a Hessian-Affine interest point detector [MS02]. On average, $20,000$ feature vectors are extracted per image. This

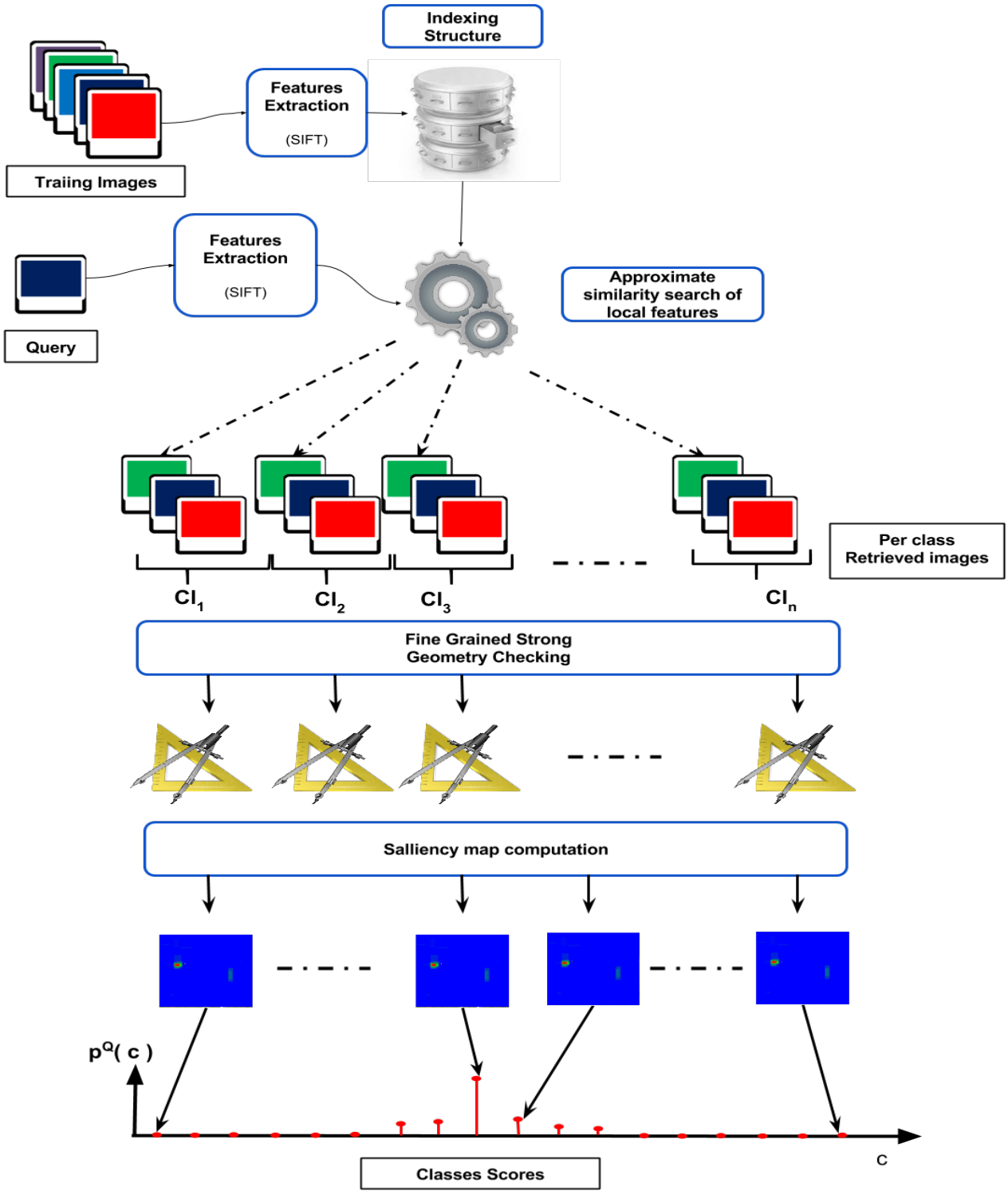


FIGURE 3.1 – Overview of the proposed approach.

corresponds to an order of about $100,000,000$ feature vectors for a training dataset composed of about $5,000$ images. This makes intractable the exhaustive search of the k-nn of the $20,000$ feature vectors for only one prediction. In the following subsections, we are going to give the details and justify which scalable NN search scheme we did use in our classification system by analyzing and comparing the state-of-the-art techniques of the literature.

3.2.1 Nearest Neighbor search methods

Searching in high-dimensional spaces consists in returning for a particular query vector $\mathbf{x_q} \in R^d$ the most similar d-dimensional vector samples $\mathbf{z_j} \in R^d$ stored in a database $\mathcal{Z}$ given a particular metric (*i.e.* L_2 distance, Earth Mover distance, inner product, etc.). In the literature, we denote two main kinds of query paradigm for searching in high-dimensional spaces: range queries and K-nearest neighbors queries. Range query consists in finding all elements $\mathbf{z_j}$ that have a distance to query $d(\mathbf{x_q}, \mathbf{z_j}) \leq r$ where $r \in R$ is a given radius and K-nearest neighbors search corresponds to finding the top-K more similar elements $\mathbf{z_j}$ with respect to the similarity function $d(\mathbf{x_q}, \mathbf{z_j})$. Range-queries are much less considered than the more popular knn-query paradigm because natural distribution does not tend to be uniform and so it is non-sense to query an isotropic radius where some region of the space will be less populated than others (possibly even empty). Several problems occur when considering large-scale search in high-dimensional spaces. The *curse of dimensionality* (see section 2.1.2) causes regular distance metric such as L_2 to be irrelevant in high dimensions due to the exponential number of possible configurations and as explained in section 2.1.2 all distances are likely to be very close implying that every samples are likely to be located in the same region of the space.

This problem and the scalability issue of massive datasets make intractable the exploration of relevant regions of the space in reasonable computation time. Indeed, the naive approach to solve this problem would be to exhaustively scan the entire database and to maintain the computed distances in a MaxHeap structure (implying a search time complexity that is linear in the size of the database). Although this sounds a reasonable way of solving the task, real-world applications often involve very huge datasets of features lying in high-dimensional spaces (128 for SIFT descriptors, 960 for GIST descriptors, 1024 for classical global CNN features and up to millions for aggregation-based representation such as Fisher vectors or VLAD) where we need to retrieve for each query similar documents in very short time (less than 1 second for real-world search engines). The first proposed similarity search methods mostly consisted in designing efficient tree-based indexing structure to organize the features such as Kd-tree [Ben75, FBF77] R-tree [Gut84], M-tree [CPZ97] or cover-tree [BKL06]. These structures have shown very good retrieval performance but only with feature spaces with a low number of dimensions above what the exhaustive scan become more efficient. This is a direct consequence of the curse of

dimensionality where similarities are highly sensitive to quantization error leading to a too high required number of cells to estimate accurate distances. This makes traditional methods intractable for both memory and efficiency issues.

To overcome these limitations, a very interesting line of research that has been considered in the literature corresponds to Approximate Nearest Neighbors (ANN) search methods. These techniques do not consider exhaustive scan of the space but rather finding as much as possible of the nearest neighbors of the query. One of the pioneering work of ANN search was the *Vector Approximation File (VA-File)* [WB97] whose main principle relies on speeding up a linear search procedure on vectors approximated with a raw dimension-wise quantification. More concretely, the input feature space is uniformly quantized into 2^b regions leading to approximated vectors of b binary components. A linear scan is first performed on these newly obtained binary signatures so as to pre-filter the potential nearest neighbors. A second scan is then performed on the uncompressed vectors with exact distance computation to filter out the irrelevant neighbors brought in the first stage. Other strategies have been proposed to provide sub-linear search time by making use of multi-dimensional indexing structures [PCS09, JB08, JDS08, JDS11] (such as trees, inverted lists or hash tables). Such methods consist in quantifying the space into discrete regions (such as for *VA-File*) and storing the vectors falling into the same region in a particular *short-list*. The indexing structure then corresponds to a key/value structure where each *short-list* (the value) is accessible by the index of the corresponding discrete region (the key). One of the first schemes proposing such search strategy was the approximate extension of tree-based search algorithms. One of the most popular is probably the approximate kd-tree search algorithms [SAH08, ML14]. The principle of this algorithm consists in building multiple randomized kd-tree that are searched in parallel and a single priority queue is maintained for all the trees. More, concretely, all vectors are stored in a regular kd-tree partitioning by randomly splitting axis-aligned directions of the input space (those among highest variance). The query is searched on each tree by visiting the cells contained in the priority queue which is ordered by increasing distance of the query to the decision boundary of the cell. The search algorithm first retrieves the closest leaves to the query and stops when the maximum number of leaves to visit in all the trees has been reached. The higher this parameter, the more accurate the returned nearest neighbors but the higher the computation time is. One of the main drawbacks of this method is that it relies on a partitioning technique that is poorly conditioned by the input data distribution. Other strategies have been employed so as to use a tree-based indexing structure whose cells better reflect the underlying structure of the input space. This is the case of hierarchical clustering techniques such as the popular *Hierarchical K-Means (HKM)* of Nister et al. [NS06] or the *Priority Search K-Means Tree* algorithm proposed by Muja et al. [ML14] which consists in combining the benefit of their proposed kd-tree based priority search algorithm and hierarchical clustering. Their method relies on efficiently exploring a K-mean tree with a *best-bin-first* strategy. The K-

mean tree is built by recursively partitioning the data into K different clusters, each of the obtained K discrete regions is then partitioned into K other clusters, etc. The process is repeated until the number of vectors in a leaf cell is lower than K. The search algorithm is then similar to the approximate kd-tree algorithm: *i)* a query vector is recursively assigned to the closest cluster while traversing the tree, *ii)* branches whose boundary is closest to query point are added to the priority queue and *iii)* the search procedure is repeated on the top branches stored in the queue. This method was found to significantly improve the performance of the approximate kd-tree search algorithm. These two methods are actually part of the popular open source *FLANN* library (*Fast Library for Approximate Nearest Neighbors (FLANN)*) [ML09, ML14]. Although such methods provide sub-linear search time, one of their main drawbacks is that they require multiple indexing structures to achieve good retrieval performance which implies storing multiple times the vectors of the database and hence requires a lot of memory. Within FLANN, the maximum dataset set size is about 10^9.

Similarly, to K-mean-tree-based partitioning techniques, other strategies arose from the need to devise indexing structure that take care of the underlying structure of the input space. One of the most popular kind of methods in this research line are those that consider flat quantization (K-mean) combined with inverted lists rather than hierarchical K-means combined with multi-dimensional tree structures [SZ03, JPD+12, JDS11]. Inverted files structures have been initially designed for text retrieval engines and well popularized by Sivic et al. [SZ03] in the context of image retrieval with the introduction of the Bag-of-Visual-Words representation. The basic principle of such structure consists in storing each document's index in a key/value table. The key entries correspond to the respective indexes of the words contained in the vocabulary. Each key entry is associated to a list of indexes of documents that contains the corresponding word. Retrieving similar documents now amounts to look at the word contained in the query document and looking at the respective entries of the structure to see which documents contain these words. The most similar documents will correspond to those having the higher number of common words with the query. Although BoVW-based methods has been widely used in the context of object retrieval, these techniques highly suffer from quantization errors involved by the underlying visual dictionary learning procedure. Most approximate search methods in this research line generally rely on, first, quantizing the space coarsely and the produced quantization indexes are, then, used as key entries of an inverted list structure. Then residual vectors belonging to each particular Voronoi cell are embedded in a compressed domain with alternative strategies of the BoVW model (such as *HE* [JDS08] or *PQ-code* [JDS11] reviewed in section 2.3.1). This allows to reduce the quantization error by refining their quantized position in the feature space defined by the coarse quantizer. The search algorithm often relies on, first, restricting the search in the region described by coarse quantization and, then, efficient similarity evaluation is performed by making use of approximate similarity

metrics between the residual quantized vectors. These distances are computed directly in the compressed domain rather than computed without loss in the original input space. For example, the *Hamming Embedding* (HE) method proposed by Jegou et al. [JDS08] consists in a Hamming distance computation between compact binary codes co-localized in the same Voronoi cell. In the same spirit, [JDS11] uses the PQ-Code representation described in section 2.3 rather than Hamming Embedding. Once again, the short binary codes are computed on residual vectors and the inverted list entries corresponds to indexes produces by the coarse quantizer. The main contribution of their work relies in both the product quantizer based representation and the approximate vector-to-code distance that allows efficient similarity computation in the compressed domain. The authors also proposed to use approximated asymmetrical distance-computation (between original residual query vector and quantized code vectors of the database) and a soft assignment strategy where they visit several inverted list entries corresponding to the coarse cluster centers nearest to the query rather than only the closest one.

Contrary to quantization-based partitioning methods where the access of the short lists relies on nearest cluster assignments, another family of methods, so-called *Hash-based partitioning methods*, rather relies on binary transformations to fastly compute the key entries of the vectors. These binary embedding $h(\mathbf{x}) : R^d \rightarrow \{0,1\}^B$ are designed to embed vectors lying in the input space in a binary space such that close vectors in the input space are more likely to be *hashed* with the same hash key. A pioneering work in this field is the popular *Locality Sensitive Hashing (LSH)* introduced by Indyk et al. [IM98, GIM+99]. These methods were the first ones to index very high-dimensional vectors by using families of hash functions defined by random processes. A popular LSH partitioning function is the linear LSH that is sensitive to the dot product. To produce B-dimensional binary codes, B random projections $\mathbf{w_b} \in R^d$ are drawn from a multi-nomial normal distribution $\mathcal{N}(0, \mathbf{I_d})$ leading to B hash functions $h_b(\mathbf{x}) : R^d \rightarrow \{0,1\}$ of the form:

$$h_b(\mathbf{x}) = sgn(\mathbf{w_b}^T \mathbf{x}) \tag{3.1}$$

and the final B-dimensional binary vector finally produced by simply concatenating the B unitary hash keys obtained from $\mathbf{x}$. The basic principle of the LSH indexing method consists, for a given query $\mathbf{x_q}$, in computing the query hash code and generating a set of candidate neighbors by taking all the vectors contained in the bucket the query is hashed to. The second step ranks the candidate vectors according to their distances to the query and finally returns the top K neighbors. The approximate search algorithm is actually quite similar to the one used in quantization based partitioning methods combined with inverted list. The hard assignment of clusters is replaced by the hashing scheme which allows to avoid the exhaustive comparison of the query with respect to all the cluster centroids learned by the K-means algorithm. To avoid the undesired effect of hard partitioning involved in such hash-based partitioning, especially in high dimension, several alternative ANN search strategies have

been considered for LSH. One of the most popular is to use multiple hash tables to store the vectors. The ANN search procedure now amounts to compute for each hash table the hash keys of the query and the candidate neighbors (to be re-ranked) now correspond to the vectors contained in all the buckets the query has been hashed to. This what is used in the LSH implementation provided by the FLANN library. Once again, the main drawback is that we need to store multiple times the vectors which requires a lot of memory. Although the LSH indexing provides sub-linear search time, another drawback of this technique is that the descriptors are stored on disk and the re-ranking step thus requires high access time to refine the retrieval results.

LSH has also been considered as a compression (a binarization) scheme that allows producing approximated binary vectors that can be fitted in memory to perform fast approximate distance computation. Producing such binarized hash codes have actually been a very active research area in the ANN search literature where most of the proposed methods were motivated to cast L_2 distance computation to approximate distance computation in the Hamming space. One of the most popular and effective method in this research line is the so-called *Iterative Quantization (ITQ)* proposed by Gong et al. [GL11]. Contrary to Hamming Embedding where we use a randomized rotation projection to provide better balancing in the dynamic of the feature components, ITQ proposes to learn in an EM fashion an optimal rotation matrix so as to explicitly minimize the quantization error. As in Hamming Embedding, this rotation is applied on top of a PCA-based dimensionality reduction. Contrary to the LSH scheme presented above, such binarization techniques are often denoted as *data dependent* hashing methods. In this type of methods, hash functions are not deterministic or purely randomized anymore but rather learned from a training set. The objective is to better fit the data distribution while maintaining a good selectivity/locality-sensitivity trade-off. Another approach that has been considered for data-dependent hashing is Spectral Hashing (SH) proposed by Weiss et al. [WTF09]. This method is motivated by graph partitioning theory and consists in applying a PCA embedding before quantizing each component to produce binary codes. Although this formulation seems very simple, the authors generalize their methods to any eigen-function computed along the PCA directions of the data. This technique has the advantage to encode more informative component with more bits than the less informative ones and, thus, reduces the quantization error. One of the main drawback of this methods is that it makes the assumption that the input distribution is uniform and rectangular which is not realistic. Another popular method is the so-called *Semantic Hashing* proposed by Salakhutdinov et al. [SH07a] based on Deep Restricted Boltzman Machines [Smo86, Hin02, HOT06] (Deep RBM). The very interesting property of their algorithm is that it can be turned into a semi-supervised learning algorithm which aims to embed data into a latent space where input samples having the same label will be close to each other. It actually consists in a deep RBM-based autoencoder framework where they add to the reconstruction objective function a similarity-based penalization term similar to

the one of *Neighborhood Component Analysis (NCA)* [GHRS04, SH07b]. This penalization term acts as a kind of metric learning forcing similar samples to be attracted with respect to each other in the latent space. As deep learning methods allows learning a progressive embedding of the input data into higher and higher semantically meaningful spaces, these methods have been successfully applied in image retrieval schemes [BH07, WWH+14, WLKC16]. Other strategies like *Random Hashing Maximum Margin (RMMH)* proposed by Joly and Buisson [JB11] introduces a new hash functions applicable to any type of kernel by, somehow, guarantying a form of independence between the hash functions. The particularity of RMMH relies on the fact that it aims at privileging balanced partitioning to avoid collision while favoring independence between the hash buckets. The algorithm is very simple as it consists in randomly selecting a subset of M feature vectors, where $M << N$, and randomly labelling $\frac{M}{2}$ of them with $+1$ and the $\frac{M}{2}$ remaining ones with -1. Then a max margin classifier (SVM) is learned and the resulting separating hyperplanes are used to produce one binary hash function. The two previous steps, random selection and SVM learning, are then independently repeated b times to produce the final embedding. Intuitively, the random selection part of the algorithm is twofold. First, it allows to take part of a desirable property in representation learning that consists in defining a latent space where the different components are as much mutually independent as possible. Secondly, as far as M is much lower than the total number of feature points N, the probability to randomly select two points that would come from the same *natural cluster* (if such thing would exist in high-dimensional feature space) is very low. Then, contrary to what we could think at first glance, the SVM learning will not force to separate the space into non-natural clusters of feature vectors. As we can guess from what we have said in section 2.1.2, this is especially true in high-dimensional spaces and this implies that such random grouping combined with a max-margin learning algorithm allows to perform some complex metric learning guarantying balancing of the data and some independence in the learned representation. Later, Heo et al. [HLH+12] proposed to investigate another way of building independent hash functions with their so-called *Spherical Hashing* scheme currently has state-of-the-art performance on large scale retrieval. It consists in learning a set of prototypes $\mathbf{p_j} \in R^d$ lying in the input space and an associated set of radii $r_j \in R$ such that each unitary hash function $h_j(\mathbf{x}) : R^d \rightarrow \{0, 1\}$ is of the form:

$$h_j(\mathbf{x}) = \begin{cases} 1 & ||\mathbf{x} - \mathbf{p_j}||_2 < r_j \\ 0 & else \end{cases}$$

To favor some form of independence between the hash functions, the pivots are learned in an iterative manner in such way that a repulsive force pushes away from each other pivots that highly overlap in terms of local descriptors assigned to them. On the other hand, pivots that poorly overlap are attracted. The recent literature of hashing methods for content-based retrieval is very rich and we will thus not entirely cover it. However, we advise the interested reader to look at the very inspiring

survey of Wang et al. [WSSJ14] for a more exhaustive overview of hashing schemes.

Note that most of these binarization techniques [GL11, WTF09, SH07a] are denoted as hashing schemes because they produce binary vectors. But, most of them are often used without hash tables or other indexing structures. The retrieval performance of such approaches have actually been evaluated only by using linear exhaustive search on the compressed domain thanks to efficient assembly instructions to compute hamming distances. This is not to be confused with hash-based partitioning (or indexing) techniques which rely on designing indexing structures based on hashing schemes to produce binary hash keys that are sensitive to localization of the vectors in the input space. Such indexing techniques, as the implementation of LSH in FLANN, often does not compress the vectors lying in the buckets but rather keep the original ones and re-rank them with exact distance computation. Actually, *Hamming Embedding* has been originally motivated by combining indexing structures and binarization techniques to provide both sub-linear search time by using coarse quantizer (and inverted lists), and efficient approximated distance computation with binarization techniques that allows to fit all the compressed vectors in memory. Similar strategies have been considered for hash-based indexing methods such as the work of Joly and Buisson [JB11] when using the RMMH scheme presented above. Such hash-based indexing techniques consider the first bits of the produced binary signatures as key entries of a hash table similarly to the coarse quantizer used in Hamming Embedding. The remaining bits can be seen as additional information to refine the position of the original vectors and then correspond to the compressed refined vectors such as those used in Hamming Embedding. Approximate search algorithms associated to such hashing schemes are then quite similar to the one proposed for quantization-based methods, *i.e.* the first bits of the hashed query vector give the key entry to look at in the hash table. A Hamming distance is computed between the compressed query vector and the compressed vectors listed in the corresponding hash table entry. This corresponds to the so-called *mono-probe* access paradigm which is the basic approximate search algorithm used in LSH-like framework that has also been extended to a version with multiple indexing structure. The interest of such hash-based indexing methods is that they allow to simultaneously deal with the embedding, quantification and compression steps listed above. However, as explained before, such mono-probe access methods can be quite sensitive to quantization error and mono-probe access in multiple indexing structures can severely increase the memory size needed to store the indexed data. Then, some methods rather focused on the multi-probe query paradigm such as [LJW+07, JB08, YBV13]. Multi-probe search strategies consist in looking in one or more entries per table so as to visit parts of the space near the one that contains the query. This is similar to soft assignment strategies employed in PQ Code [JDS11] that can be considered as a data dependent hashing schemes relying on a quantization-based indexing technique. Multi-probe search has the very interesting property that it allows to increase both memory efficiency and probability to retrieve relevant neighbors. The

difficulty in such strategy relies on determining the regions of the space that have the best probability to contain similar vectors. Particularly, Joly and Buisson [JB08] proposed a probabilistic multi-probe search algorithm, so-called *A Posteriori Multi-Probe LSH (APMP-LSH)* to select the buckets of the hash table that are the most likely to contain exact nearest neighbors. This is done by using a probabilistic search model that is trained offline on the exact m-nearest neighbors of M sampled features $\mathbf{z} \in \mathcal{Z}$. Another very interesting property of their algorithm relies on the fact that they provide a quality control parameter so as to be able to trade quality of search for time efficiency. APMP-LSH has been shown to give very good retrieval performance when coupled with RMMH [JB11].

3.2.2 Discussion and justification of the chosen scheme

As we are in a context of fine-grained classification, several constraints should be taken into account to choose a ANN search scheme that is well adapted to our needs. First, as explained in the beginning of this section 3.2, we need to extract a lot of descriptors in the images which leads to a very large database of feature vectors (up to 100 million or 1 billion of descriptors). So, what we would like to have is a search procedure that is sub-linear in the size of the database. This property is actually available in methods such as approximated tree-based search indexing or LSH that are implemented in the open source FLANN library. However, the main drawback of such methods is that they use multiple indexing structures and they do not compress the vectors which lead to high memory inefficiency and a lot of disk accesses for the re-ranking step. Then, a better solution would be to consider ANN schemes using both indexing structures and binarization techniques so as to provide both sub-linear search time and efficient re-ranking step from compressed vectors that can be fitted in memory. One of the main advantages of hashing-based partitioning methods compared to quantization-based indexing techniques (such as HE or PQ code) is the fast inference property, *i.e.* the computation of the keys to access the elements of the structure. Indeed, quantization-based partitioning methods often rely on nearest neighbors affectation whereas the inference complexity of hash function is close to $O(1)$ as we just have to apply the transformation to the input vector. The resulting hash key can directly be used as an entry of the indexing structure. As we have seen, if the hash functions are cleverly designed, it is possible to simultaneously deal with the indexing and compression steps thus providing both sub-linear search time and efficient distance computation of compressed vectors that can be fitted in memory. The indexing structure can be used in the same spirit than coarse to fine quantization-based indexing structures by considering the first bits of the hash keys as coarse quantizer key entries and the remaining bits as the short binary signatures to be compared with the query. Also, the adapted quantification schemes associated to such hashing methods has the advantage to reduce the undesirable effect of hard quantization involved in traditional clustering-based quantizers. This prevents mutually exclusive explanatory factors, *i.e.* when each vector is hard

assigned to only one cluster which is assumed to not depend on the others. On the contrary, data dependent hashing schemes rather benefit from multi-clustering properties where each binary hash function participate to the refinement of the original information (*i.e.* location) of the original vectors. The different projections learned in data dependent hashing schemes correspond to non-mutually exclusive explanatory factors explaining the original input vectors which is less prone to suffer from the curse of dimensionality. Moreover, as we have seen, some search algorithms based on multi-probe hashing partitioning methods provide quality search vs time efficiency trade-off which is a very desirable property for large scale retrieval methods as we can adapt to the particular needs of performance. Such properties are not available in indexing methods implemented in FLANN nor in the popular PQ code method.

For those reasons, we decided to make use of an approximate nearest neighbors scheme highly inspired by the state-of-the-art approaches of Joly and Buisson [JB11, JB08] to speed up the matching step of our classification approach. The RMMH [JB11] scheme is used for feature hashing and the APMP-LSH [JB08] scheme is used to perform large scale approximate search. However, we used a simpler search model than the one of [JB08]. We define the probabilistic search model by fitting an isotropic normal distribution parametrized by a single vector $\boldsymbol{\sigma}$ that is trained over the exact nearest neighbors of the training samples. We compress the local features of the training set $\mathbf{z} \in \mathcal{Z}$ into compact binary hash codes $\mathbf{h}(\mathbf{z})$ of length b thanks to the RMMH scheme. The obtained hash codes $\mathbf{h}(\mathbf{z})$ are then indexed in a hash table whose keys are the t-length prefix of the hash codes $\mathbf{h}(\mathbf{z})$. At search time, we also used a slightly different probabilistic multi-probe algorithm trading stability for time. Instead of probing the buckets by decreasing probabilities, we rather use a greedy algorithm that computes the probability of neighboring buckets and select only the ones having a probability greater than a threshold ζ that is fixed over all queries. The value of ζ is trained offline on M training samples and their exact nearest neighbors so as to reach on average a cumulative probability α over the visited buckets. In the experiments that we led in this chapter, we always used $\alpha = 0.80$ meaning that on average we retrieve 80% of the exact nearest neighbors in the original feature space. Once the most probable buckets have been selected, the refinement step computes the Hamming distance between $\mathbf{h}(\mathbf{x})$ and the $\mathbf{h}(\mathbf{z})$'s belonging to the selected buckets and keep only the top-K matches thanks to a max heap.

At this stage, we are now able to retrieve the similar regions with respect to the feature vectors of the query. This corresponds to retrieving similar images from local invariant of the image without taking care of their respective spatial arrangement. In the next section, we present a newly introduced method to finely post-check strong localized geometry consistency between the query image and local patches contained in the *coarsely* retrieved training images.

3.3 Localized weak and strong geometry consistency

Post-checking the geometric consistency of the raw visual matches is an efficient strategy to filter false positives and consolidate good matches by integrating spatial configuration information between local features in the global similarity score. The RANdom SAmple Consensus (RANSAC) [FB81] algorithm has notably been successful in rigid objects retrieval [PSZ11] in particular logos [JB09]. RANSAC does not estimate the transformation between two subsets of points with an optimal regression but rather estimate multiple weakly optimal transformations of sampled pairs of matched points and keeps the ones corresponding to the best consensus. The algorithm stops when an acceptable consensus is reached or when a defined number of iterations is reached, or when all the possible combinations of samples are tested. Numerous variants of RANSAC have been presented since the publication of the original algorithm. In SCRAMSAC [SLK09], a spatial coherence checking step is performed to reduce the number of matches, retaining the most consistent. The PROSAC algorithm [CM05] promotes the samples that matched with the best confidence score. Moisan et al. [MS04b] introduced an *A Contrario* RANSAC method called *AC-RANSAC*, to get a better robustness to noise by comparing the RANSAC hypothesis with a noise hypothesis. Finally, the MAC-RANSAC algorithm [RDGM10] generalizes AC-RANSAC by providing even better results in the context of multiple localized objects by either constraining the distance of the randomized candidate pairs or by iteratively removing validated matches from the pool of candidates. The main drawbacks of these methods is that they often require a higher computation time and this is why some methods proposed to reduce this effect by limiting the number of hypothesis to be considered by the RANSAC algorithm such as SCRAMSAC [SLK09] and LO-RANSAC [CMK03]. But even with such improvements, a global RANSAC algorithm applied at the image-level is not adapted to the detection of very small objects in highly cluttered images for which the percentage of inlier pairs of matches can be typically lower than 0.1% of the whole set of possible pairs. Furthermore, as it is computed on the retrieved images one by one, it does not allow consolidating locally the matches from different training images.

Sliding-RANSAC To address these issues, we introduce a *sliding* RANSAC strategy aimed at checking the geometric consistency locally for each of the N_Q query features of the query image I_Q. More precisely, for a given local feature $\mathbf{x}_j^Q \in I_Q$, its m spatial nearest neighbors are computed so as to define a candidate region of interest to be geometrically checked in all the retrieved pictures (*i.e.* in the ones having some visual matches within the $m+1$ lists of K-NN's). For a given candidate region of interest and a given retrieved image, we use a modified version of the RANSAC algorithm that we describe in the next paragraph. Note that the support of both the *random sampling* and the *consensus* phases is bounded by the set of local features belonging to the current region of interest. This allows improving the recall and the precision of the inliers compared to the classical global RANSAC algorithm. The

parameter m controls the locality constraint of the geometry consistency analysis. Ideally, it should fit the size of the targeted objects of interest. Too large values of m would lead to the same problem than a global RANSAC. Too small values of m would degrade the dynamic of the number of inliers and possibly miss some consistent matches. In our experiments, m was trained by cross-validation.

Modified RANSAC Algorithm Let us now give the details of the modified RANSAC algorithm that we use in the Sliding-RANSAC procedure. Each raw visual correspondence $\{\mathbf{x_q}, \mathbf{x_q}\}$ is associated with a rank $r_q(\mathbf{x_i})$. This allows two things: (i) to restrict the generation of the hypothesis of the RANSAC algorithm to the best match of each query feature $\mathbf{x_q}$ in the targeted image. The number of evaluated hypothesis is consequently reduced, particularly in the presence of numerous repeated visual patterns (the burstiness phenomenon [JDS09]) (ii) the ranking can be used in the computation of the final score by weighing the contribution of each inlier according to its rank in the whole dataset. Closest points are then favored to the detriment of the farthest ones, independently from the feature space density in the neighborhood of $\mathbf{x_q}$. Finally, the geometrically consistent score of a retrieved image I is computed as:

$$S_Q(I) = \sum_q I\{||\mathbf{P_q} - \mathbf{A}\mathbf{P_q^I} + \mathbf{B}|| < \theta\}.\varphi(r_q(\mathbf{x_q^I})) \tag{3.2}$$

where $(\mathbf{A}, \mathbf{B})$ are the parameters of the best transformation estimated by the RANSAC algorithm for the image I, $\mathbf{P_q}$ and $\mathbf{P_q^I}$ are the spatial positions of respectively the query feature $\mathbf{x_q}$ and its best match $\mathbf{x_q^I}$ in I, $\varphi(.)$ is a decreasing weighting function on the rank $r_q(\mathbf{x_q^I})$ (typically the inverse or a linearly decreasing function), $I\{.\}$ is an indicator function equals to 1 if the assumption in the braces is true and zero otherwise.

Weak Geometry Consistency One of the main drawback when using such localized strong geometry consistency checking scheme is that it demands higher computation time which prevent from performing prediction in reasonable time. As previously explained in section 2.5.2 of the previous chapter, another way of considering geometry while reducing the post checking process time is to make use of so-called *Weak geometry consistency* that consists in computing distortion between two potential matched descriptors by comparing their respective geometric information, *i.e.* spatial position, angle, scale, without taking care of the others descriptors. This principle has been used in retrieval systems such as [JDS08, JDS10] where we need to deal with a huge number of documents in the database. Another possibility is to use weak geometry as a pre-filtering step reducing the number of documents to be compare more finely with strong geometry schemes allowing to benefit from both lower computation time and precise post filtering to remove irrelevant matches as in LO-RANSAC.

Similarly, to the LO-RANSAC algorithm, the spatial verification we use in the sliding-RANSAC is a variant of the RANSAC algorithm making use of weak geometry rules generated from the region shape characteristics. We however do not use the weak geometry to directly generate a hypothesis from a single visual correspondence. We rather use it to filter the exact hypothesis generated by the classical RANSAC algorithm. Concretely, if we restrict our class of transformations to rotation and scaling, the RANSAC algorithm can generate a hypothesis from any pair of visual correspondences. To quickly decide whether this hypothesis is relevant or not, we check its consistency with regard to the two approximate hypothesis generated from the shape characteristics of each visual correspondence. If any of the two approximate models does not fit the RANSAC hypothesis, we reject that solution without computing the costly consensus phase. In practice, up to 99% of the RANSAC hypothesis can be rejected in that way.

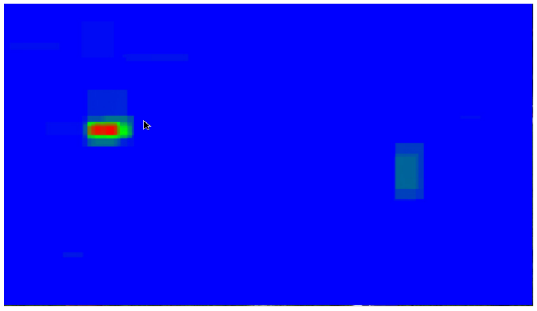
3.4 Class-specific geometry consistency maps

The output of the sliding RANSAC algorithm is a set of N_Q lists of consolidated results (i.e. one list per query feature $\mathbf{x}_j^Q$). Each consolidated result $R_{j,t}^Q$ is itself defined as a set of individual matches of the form $(\mathbf{x}_{j'}^Q, \mathbf{x}_{t'})$ where the $\mathbf{x}_{j'}^Q$ belong to the m spatial neighbors of $\mathbf{x}_j^Q$ and the $\mathbf{x}_{t'}$ belong to an image I_t of the training set. In order to construct saliency maps, we first associate each consolidated result $R_{j,t}^Q$ with an individual geometry consistency score $f_{j,t}^Q$ and a bounding box $B_{j,t}^Q$ in the query image. Rather than simply counting the number of inlier matches, the score $f_{j,t}^Q$ is computed as the sum of the inverse rank of the matched features $\mathbf{x}_{t'}$ (rank in the K-NN's of $\mathbf{x}_{j'}^Q$). This allows giving more importance to the most confident visual matches. The bounding box $B_{j,t}^Q$ is defined as the minimum bounding rectangle containing all the individually matched features $\mathbf{x}_{j'}^Q \in R_{j,t}^Q$.

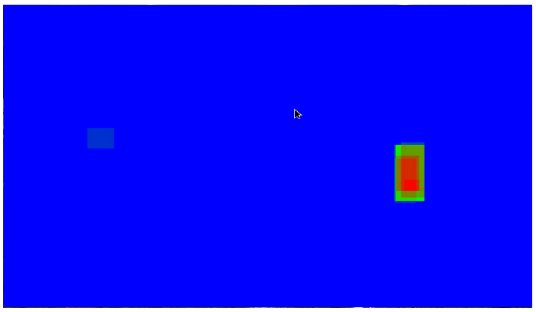
The pixel-wise consistency score $g_c^Q(w,h)$ of a pixel (w,h) according to class label c is then computed by (i) selecting the consolidated results $R_{j,t}^Q$ whose bounding box $B_{j,t}^Q$ intercepts (w,h) (ii) grouping them according to the provenance image I_t and averaging the scores $f_{j,t}^Q$ for each group (iii) summing the averaged scores of the groups whose provenance images I_t are labeled with c. This allows voting on the number of pictures retrieved for label c and weighting each vote by an average geometry consistency in each image. Figure 3.3 displays two saliency maps $g_c^Q(w,h)$ computed for two distinct class labels in a single query image.

FIGURE 3.2 – Some examples of geometrically consistent matches between query images and training images.

(a) Raw query image



(b) Class *Base*



(c) Class *Quick*

FIGURE 3.3 – Class-specific geometry consistency maps. For each class, local regions of the query that correspond to frequent geometric patterns are set to high saliency values. A max pooling operation is then performed for each class-map so as to assign a detection score to the corresponding class.

3.5 Label Scoring

As illustrated by Figure 3.3, the saliency maps produced by the previous step could be easily used for a precise localization of the visual patterns recognized for each of the retrieved entity. We will present some example of object localization later in section 3.6.4. To perform classification, we use these maps to build a strong classifier at the image level. This is done by simply taking the value of the most salient pixel in each map (i.e. for each returned label):

$$s^Q(c) = \max_{(\mathbf{w},\mathbf{h})} g_c^Q(w,h) \tag{3.3}$$

where $s^Q(c)$ is the detection score of the label c.
As shown in Figure 3.3, this last step also acts as a disambiguation procedure where different classes can co-occur in different images and bring geometry consistency in other class-specific saliency maps.

To decide whether a given legal entity is detected or not, a threshold τ_s is applied on the $s^Q(c)$ scores (several annotations can thus be produced for each image). To better model the density distribution over the classes, a normalization is then applied according to:

$$p^Q(c) = \frac{s^Q(c)}{\sum_{c'} s^Q(c')} \tag{3.4}$$

where $p^Q(c)$ is the probability estimation of the presence of the label c.

Discussion The main line of the work in this chapter is to use *online* geometry consistency checking to disambiguate instance-based matches rather than training discriminative models offline. This is justified in several ways. First, our training phase is reduced to a simple indexing process with a linear time and space complexity $O(N)$. The prototype selection technique of [KPFJ14] requires computing the $20,000$-NN's of each of the N features of the training set, leading to a much more important training time (over-linear in N). Concerning the memory storage, their method requires at least 8 times more RAM to store the original SIFT features. Besides, the complexity of other state-of-the-art methods making use of pooling and SVM's is typically $O(N + |C|.|S|^2)$ such that they are less scalable in both the number of classes and the number of images. Beyond scalability, our method has several other advantages including the easy management of multi-labeled images, the fine grained localization of the recognized patterns making them highly interpretable and the possibility of dynamically inserting additional training images or even classes in an incremental way.

3.6 Experiments

3.6.1 Instance classification: comparison to state-of-the-art methods

In this section, we evaluate the performance of our approach on different classification tasks (multi-label or not) and compare our instance classification results to the other state-of-the-art approaches.

Datasets

The evaluation is performed on 2 datasets illustrated in Figure 3.4:

— ***FlickrLogos32*** was originally introduced by Romberg et al. [RPLvZ11] in the context of object retrieval. It is composed of 8,240 images divided into 2,240 images labelled with 32 logo classes and 6000 distractors considered as a no-logo class.

— ***Vehicles29*** [KPFJ14] is composed of 10,622 images labelled with 29 car models of 7 brands (divided into 5,266 training and 5,356 test images). As explained in the introduction of this chapter, the different instances of car models are represented in different view point which make the classification task somehow more difficult with respect to the other ones involving rigid planar objects such as logos.

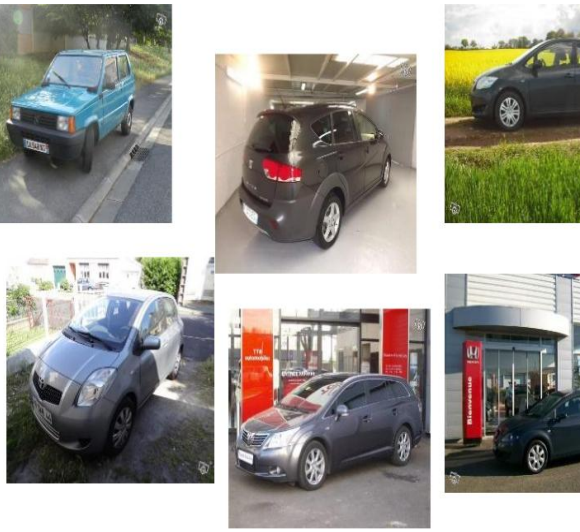
Features and parameters setting.

Local features are SIFT features [Low04] extracted around interest points detected by the rotation-invariant Harris-Hessian-Laplace detector [MS02]. The two main parameters of our method, i.e. the spatial neighborhood size m and the detection threshold τ_s were trained by cross-validation on the training sets of *FlickrLogos32* and *Vehicles29*. m was set to 5% of the number of descriptors for each query image and τ_s was set to 5.

Results

Table 4.2 reproduces the results of [KPFJ14] and reports our own results using the same evaluation protocol. It can be seen that our method outperforms the previous baseline of [KPFJ14] (and de facto the other state-of-the-art classification methods evaluated in their paper [KPFJ14]) on the two experimented datasets, whereas the training stage of our method is much more scalable. It took respectively 13 minutes and 22 minutes to index and to compute the a posteriori multi-probe search model of the 51,054,054 descriptors of the Vehicles29 training set and the 91,800,540 descriptors of the FlickerLogos32 training set (including distractors images). As might be expected, although our approach slightly outperforms other state-of-the-art methods on the Vehicles29 datasets, the results obtained on this

(a) FlickrLogos32



(b) Vehicles29

(c) BelgaLogosII

FIGURE 3.4 – Instance classification datasets.

Method	FlickrL32	Vehicles29
Fisher Vectors (128x4,096) [PSM10]	0.866	0.497
Prototype voting [KPFJ14]	0.914	0.557
Our method (S-Ransac)	**0.928**	**0.597**

TABLE 3.1 – Classification performance.

benchmark are significantly lower than those obtained on logos and are thus not up to what we might expect from a system that could be used in real-world fine-grained classification tasks. This motivates the design of a more effective system that could learn to optimally combine local geometric invariant captured by our underlying voting algorithm.

3.6.2 Multi-label classification

To evaluate our method in the context of multi-labeled images, we used the challenging BelgaLogos dataset [JB09] which corresponds to 10,000 images containing 2695 instances of 37 different logos. The main difference with FlickrLogos32 is that images are multi-labeled and that that the average size of the logos is much lower. For this experiment, we randomly chose 1,000 images of the dataset that contain at least one labeled logo for building the test set and we used the remaining images as the training set. Our method achieves a mAP of 82.30 which is quite impressive knowing that this dataset involves very small objects in highly cluttered contexts. To better illustrate the relevance of the produced labels, Figure 3.5 displays the precision-recall curve of our method when varying the detection threshold τ_s. It shows that very high precision or recall values might be obtained depending on the applicative constraints.

3.6.3 Large Scale Legal Entity Recognition

Legal entities (such as firms, government bodies, political parties, societies, associations, etc.) are entities other than natural persons (human being) created by law and recognized as having duties and rights. It does not exist any estimation of the number of such legal entities but they are omnipresent in our all day life as well as in all media contents. Beyond their legal identity, most of them also have a corporate visual identity, that is a set of graphical rules and elements providing an organization with visibility and recognizability (graphic charter, logotype, insignia, colors, polices, fonts, etc.). As for natural persons, it is therefore possible to recognize them automatically in visual contents in order to provide automatic annotations. This is of high interest for many applications involving huge amounts of weakly annotated image or video contents (YouTube, social media or massive TV archives such as at INA). Whereas legal entity recognition is considered as an important challenge in the text community (e.g. the freebase repository contains 741K organizations, 160K

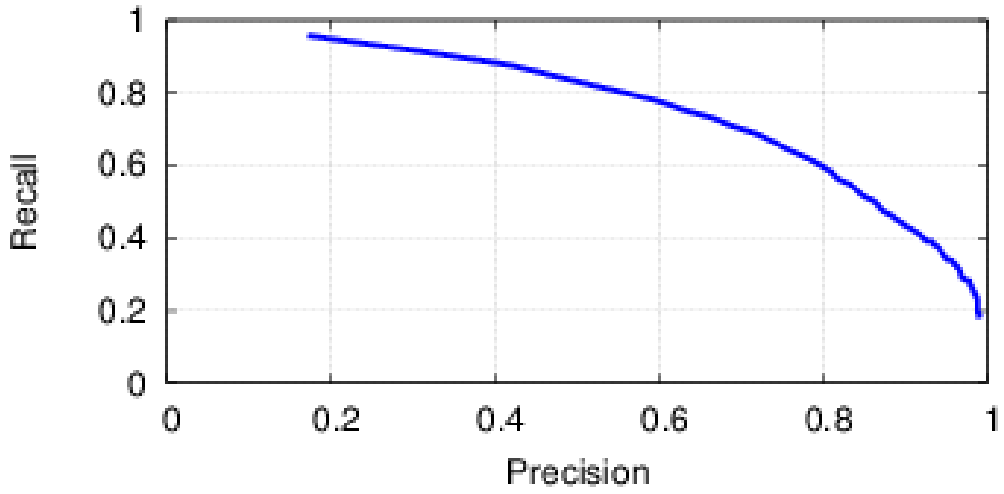


FIGURE 3.5 – Recall-precision curve of our method on the multi-labeled dataset BelgaLogos.

Benchmark	mAP
2.5K images / LegalEntities5K	**0.686**
FlickrLogos / LegalEntities5K	**0.648**

TABLE 3.2 – Classification results and computation time on the *LegalEntities5K* dataset (Intel(R) Xeon(R) E5-2650 CPU 2.00GHz).

educational institutions or 31K sport teams), the problem has received little attention in the multimedia community.

In this section we propose to show the performance of our model on a newly introduced dataset *LegalEntities5K* composed of 371,924 images noisily labelled with 5,824 legal entities. This dataset was automatically created by querying Google Image search engine with the entities names. The list of the entities is the union of several thesauri found on the web and contains world-wide companies, associations, organizations and sport teams. To try limiting noise, we kept only the top-20 to top-1000 results as a linear function of the popularity of the tag (number of pages returned by Google). A test set was built by randomly sampling $2,500$ images and removing them from the training. A second test set was created by intersecting the FlickrLogos32 dataset with the labels of LegalEntities5K (540 test images belonging to 18 classes).

It took 1 hour and 55 minutes to index the 500,957,407 SIFT features of the *LegalEntities5K* dataset and to compute the a posteriori multi-probe search model which can be seen as the training phase of our model. Table 3.2 reports the results achieved on the two test tests in terms of mAP and search time. Comparing the

FIGURE 3.6 – LegalEntities5K Dataset.

performance with that of Table 4.2 shows that the effectiveness of our method is still very satisfactory considering that (i) the number of classes in the training set is two orders of magnitude higher (ii) the training set was built automatically without any human validation and therefore contains a high level of noise.

3.6.4 Saliencies and object localization

One of the desirable effects of such methods lies in the fact that it is possible to know where things matched and how much it matched. This allows us to use our geometry consistency maps to design an object localization system whose qualitative results are presented in this section. Figure 3.8 first shows few saliency maps similar to those presented in Figure 3.3 (with the same colormap) except that these ones correspond to the multi-label maps which simply consists in the aggregation of the class-specific maps (normalized sum pooling with respect to the saliency value of the most salient pixel). In these figures, the saliencies are superposed on the original image and only the pixels whose score exceeds a given threshold (0.5 in our experiments) are associated with their saliency value while the others pixels are associated to their original value (corresponding to the content of the image). This allows us to visualize regions of the image that have a high probability to correspond to an object of interest belonging to the visual dictionary. Figure 3.7 shows the result of a simple detection scheme based on pooling on top of the saliency maps where we simply calculate the bounding boxes of each connexe salient regions. Despite the simplicity of this approach, detection and localization results are pretty good on logos and could be improved with more sophisticated localization process.

FIGURE 3.7 – Some examples of logos detections.

FIGURE 3.8 – Some examples of geometrically consistent saliencies.

3.7 Conclusion

The work presented in this chapter allowed us to set the baseline of the methods introduced in the next chapter. We proposed a spatially consistent matching-based k-nn classifier relying on the pooling of localized similarity scores of images with respect to the query image. We tested our approach on several benchmarks involving logos in natural images (FlickersLogos32 [RPLvZ11], Belgalogos [JB09]), car models (*Vehicles29* [KPFJ14]) and, finally, a large scale corpus and corresponding to over 300,000 images representing 5,000 classes of legal entities crawled on the web. The main contribution lays in the integration of localized strong geometry. The geometry consistency maps clearly show that the integration of the geometry drastically reduces the number of false alarm and focuses the attention on regions of interest. We show that this approach exhibits promising instance classification and localization performance but it still has difficulties to discriminate visual patterns such as those involved in the *Vehicles29* dataset. This is mainly due to the fact that the pooling-based classification does not correspond to good combinations of local information that allows discriminating some classes. Another noticeable drawback about this approach is that, for such classification tasks, some object belonging to different classes are prone to share a high number of common features. This implies that the classifier gives to each localized pattern as much importance as for the others ones whereas only some of them correspond to discriminant features. For instance, every car models share the visual concept of wheel and only some subtle localized details (such as brand logos or shapes of the front car) allow to discriminate between two models. Thus the overall similarity score is composed of a lot of task-irrelevant sub-scores that we would like to get rid of.

This suggests that the pooling should take into account the relative discriminating importance of the different local patterns of the visual dictionary that have been matched. In the following chapter, we will see how to build a new image representation based on the aggregation of geometrically consistent matches and how it can be seen as a new kind of Match Kernel with its associated explicit embedding.

Chapitre 4

Fine-Grained Classification through Shared Nearest Neighbor Embedding

Contents

4.1 Introduction

The work presented in this chapter is positioned in the same research direction than the previous one, it is primarily inspired by instance-based image retrieval methods. It still makes use of efficient indexing and approximate k-nearest neighbors search techniques to match individual local features (such as in [JDS11, JB08]) and it uses a localized parametric spatial consistency checking such as in the previous chapter.

The main contribution, however, is to kernelize such methods to build high-dimensional image representations suitable for advanced machine learning algorithms (rather than simply relying on an instance-based classification scheme such

as the one presented in the previous chapter). Thus, the method introduced in this chapter has some strong connexions with the kernelized and similarity-based image representations in [JBJG12] discussed in the state-of-the-art chapter (see section 2.5.2). Similarity-based embedding is elegant in that it builds representations while keeping trace of the similarity between the training samples (which makes it belongs to the category of memory-based methods). This gives them an interpretability property suitable for understanding which visual patterns of the training set are useful to train the model. However, many state-of-the-art similarity-based and match kernels methods have the main drawback that they do not take into account the spatial arrangement of the local features in the image. What we would need is a representation whose components correspond to localized visual patterns so that the back-end classifier could learn which local details can discriminate the different classes. Although some kernelized methods such as the popular Spatial Pyramid Matching (SPM) of Lazebnik et al. [LSP06] encode the absolute position of local features in the image to be predicted, they do not encode the spatial arrangement between the visual patterns. As a consequence, they are not invariant to translation of small objects or small details which is crucial in a fine-grained classification context.

Kernelized representations offer a nice formalization framework from which we can derive an explicit image representation space tractable by linear classifiers. Our three main contributions over existing methods in this research line are *i)* the use of an intermediate match kernel to compare pairs of features, *ii)* a rank-based similarity function used to pool the individual local matches and *iii)* the embedding of the localized geometric consistency in the kernel which provides us both invariance to translation and local geometric distortions. Figure 4.1 illustrates how we concretely design our explicit image embedding based on this underlying principle. For each image, local features are densely extracted and indexed via a data-dependent hashing scheme. Then, for each image, similar images are searched thanks to a fast approximate K nearest neighbor algorithm. Retrieved images are uniformly subdivided forming a spatial grid at a given resolution. The similarity between the query image and the sub-regions of the similar retrieved images are evaluated and aggregated into a global vector that can be fed to a linear classification algorithm. In order to allow the system to learn spatially consistent visual patterns, we perform local geometry consistency analysis in a RANSAC fashion to refine the similarity score of the retrieved sub-regions.

In this chapter, we first give a detailed explanation of the different concepts involved in our new representation. We then present a study on the impact of the different parameters of our method. We show that learning from these representations yields very strong performance gains with respect to the baseline of the previous chapter on the Vehicles29 [KPFJ14] and FlickersLogos32 [RPLvZ11] datasets. Finally, we discuss the interesting properties of the SNN representation and we see that it is related to popular aggregation-based methods (such as BoVW, Fisher Vectors,

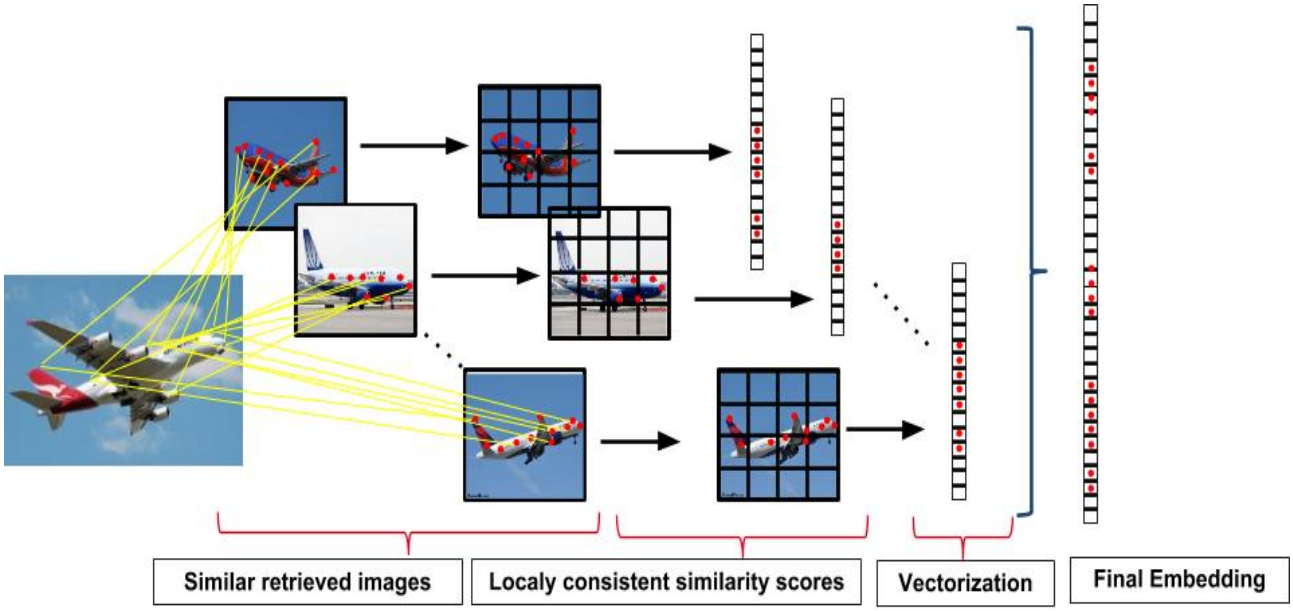


FIGURE 4.1 – Overview of the proposed image representation.

Locality-constraint Linear Coding).

4.2 The Shared Nearest Neighbor Representation

4.2.1 The Shared Nearest Neighbors Match Kernel: Basic Formulation

We consider two images I_x and I_y represented by sets of local features $\mathcal{X} = \{\mathbf{x}\}$ and $\mathcal{Y} = \{\mathbf{y}\}$ where $\mathbf{x}$ and $\mathbf{y}$ are d-dimensional feature vectors (SIFT features [Low04] in our experiments). We then build on the *normalized sum match kernel* proposed by Lyu et al. [Lyu05] to compare feature sets:

$$K(\mathcal{X}, \mathcal{Y}) = \Phi(\mathcal{X})^T \Phi(\mathcal{Y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|} \sum_{\mathbf{x},\mathbf{y}} k(\mathbf{x}, \mathbf{y}) \tag{4.1}$$

where $k()$ is itself a base Mercer kernel allowing to compare two individual local features $\mathbf{x}$ and $\mathbf{y}$. In our case, $k()$ is however not defined as a direct matching between $\mathbf{x}$ and $\mathbf{y}$ but rather as the degree of correlation of their matches in a large training set. Let denote $\mathcal{Z}$ such a training set composed of N d-dimensional feature vectors $\mathbf{z}$. We introduce the following *shared nearest neighbors (SNN) match kernel*:

$$K_{SNN}(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|} \sum_{\mathbf{x},\mathbf{y}} \sum_{\mathbf{z}} k_{\mathbf{z}}(\mathbf{x}, \mathbf{y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|} \sum_{\mathbf{x},\mathbf{y}} \sum_{\mathbf{z}} \varphi(r_{\mathbf{x}}(\mathbf{z})).\varphi(r_{\mathbf{y}}(\mathbf{z})) \tag{4.2}$$

where $r_{\mathbf{x}}(\mathbf{z}) : \mathcal{Z} \rightarrow N^+$ is a ranking function returning the rank of an item $\mathbf{z} \in \mathcal{Z}$ according to its distance to $\mathbf{x}$ and K is the number of nearest neighbors returned for each point. The distance itself could be a L_2 metric in the original feature space but, as for the method introduced in the previous chapter, we use in practice a more efficient Hamming embedding scheme. We also define $\varphi(.) : R \rightarrow R$ to be a rank-based activation function that should be decreasing to give more importance to low-ranked nearest neighbors of $\mathbf{x}$ and $\mathbf{y}$. This allows us to discard points that are far away from $\mathbf{x}$ or $\mathbf{y}$ which is likely to reduce the burstiness effect (see section 2.5.2). Moreover, the use of the rank allows us to be less sensitive to the choice of the metric or to local distributions of the input feature space. A variety of functions $\varphi(.)$ are possible, for example the negative linear function, *i.e.* $\varphi(r_{\mathbf{x}}(\mathbf{z})) = A - r_{\mathbf{x}}(\mathbf{z})$ with $A \in R$ being a constant. However, it is preferable to choose a saturating function where high-ranked neighbors do not contribute to the aggregation process as they might correspond to noisy points or irrelevant visual matches with the intermediate feature set. A monotonically decreasing function that asymptotically converges toward zero has the advantage of allowing more selectivity while comparing the respective neighborhoods of $\mathbf{x}$ and $\mathbf{y}$. A common choice is the inverse function of the rank, *i.e.* $\varphi(r_{\mathbf{x}}(\mathbf{z})) = \frac{1}{r_{\mathbf{x}}(\mathbf{z})}$. The main problem with this function is that it converges too fast towards zero. Because of this, the only neighbors that will significantly contribute to the aggregation process will be the lowest ranked ones. This makes the similarity function too sensitive to the ranking of the neighbors where such ranking function can be very unstable from a feature vector to another. This especially true in dense location of the feature space. For example, the rank of a particular feature $\mathbf{z}$ could be 2 in the neighborhood of $\mathbf{x}$ and 20 in the neighborhood of $\mathbf{y}$ leading to a rank-based matching score of $k_{\mathbf{z}}(\mathbf{x}, \mathbf{y}) = \varphi(r_{\mathbf{x}}(\mathbf{z})).\varphi(r_{\mathbf{y}}(\mathbf{z})) \approx 0.02$ which is low despite the fact that $\mathbf{z}$ was reasonably ranked in both neighborhoods. For these reasons we choose to use a saturated negative linear rank-based activation function of the form:

$$\varphi(r_{\mathbf{x}}(\mathbf{z})) = \begin{cases} \frac{K - r_{\mathbf{x}}(\mathbf{z})}{K} & r_{\mathbf{x}}(\mathbf{z}) \leq K \\ 0 & else \end{cases} \tag{4.3}$$

K is the maximum size of the neighborhoods that we want to compare which amounts to restricting our look to the K nearest neighbors of $\mathbf{x}$ and $\mathbf{y}$ to compute their neighborhoods similarities. Whatever the distance used to compute the ranks, the intuition of the *SNN match kernel* is that it counts the number of common neighbors in the neighborhood of $\mathbf{x}$ and in the one of $\mathbf{y}$. The product $k_{\mathbf{z}}(\mathbf{x}, \mathbf{y}) = \varphi(r_{\mathbf{x}}(\mathbf{z})).\varphi(r_{\mathbf{y}}(\mathbf{z}))$ is actually equal to one if $\mathbf{z}$ is the nearest neighbor of both $\mathbf{x}$ and $\mathbf{y}$ and close to zero if $\mathbf{z}$ is not in the top-K neighbors of either $\mathbf{x}$ or $\mathbf{y}$. As illustrated in Figure 4.2, if the K nearest neighbors of the red point among the reference set (blue points) are about the same as the K nearest neighbors of the green point, then the green point and the red point are likely to be similar. So if two elements have correlated similarities with respect to the elements of a reference

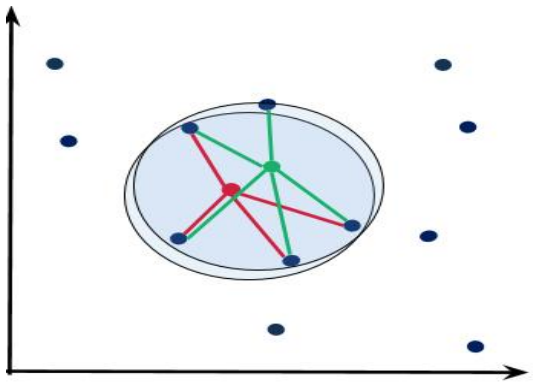


FIGURE 4.2 – Illustration of the Shared Nearest Neighbor Kernel between two local features.

dataset (*i.e.* a vocabulary), then a classifier is likely to capture these correlations.

More concretely, using this *shared nearest neighbors kernel* instead of a more classical distance in the feature space has several justifications and advantages. Shared-neighbors techniques are known to overcome several shortcomings of traditional metrics. They are notably less sensitive to the dimensionality curse, more robust to noisy data and more stable over unusual features distribution [ESK02, JP73]. Measuring the similarity between features by the degree to which their neighborhoods in the reference set are correlated is actually a form of generative metric learning. Features belonging to dense clusters are actually more likely to share neighbors than uniformly distributed and isolated features. Therefore, their contribution to the global kernel is enhanced. This is a more robust kernel than the intermediate match kernel formulation associated to the Bag-of-Visual-Words explicit representation (see equation 2.78 of section 2.5.2). Indeed, the shared nearest neighbor kernel does not consider an *abstract* vocabulary with *virtual* shared variables but rather an exhaustive vocabulary composed of *real* feature vectors extracted from a set of reference images. This allows us to considerably reduce the quantization error involved in classical intermediate match kernels such as those reviewed in section 2.5.2.

Moreover, using an indirect matching kernel rather than a direct one allows to have an explicit formulation of the embedded space $\Phi(\mathcal{X})$. By factorizing equation 4.3, it is easy to show that $K_{SNN}(\mathcal{X},\mathcal{Y}) = \Phi_{SNN}(\mathcal{X})^T \Phi_{SNN}(\mathcal{Y})$ with:

$$\Phi_{SNN}(\mathcal{X}) = \sum_{i=1}^{N} \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} \varphi(r_{\mathbf{x}}(\mathbf{z_i})).\vec{e_i} \tag{4.4}$$

such that, the explicit N-dimensional feature vector $\Phi(\mathcal{X})$ representing each image in the training set can be computed before training a classifier on top of them. The principle of this approach was already introduced in the *intermediate matching*

kernel of [BTB05b] and further re-used in many methods including the NBNN kernel of [TFSD11]. Such methods did however rely on the distance between the features of the candidate image $\mathcal{X}$ and the ones of the training set $\mathcal{Z}$ so they did not benefit from the nice properties of the SNN-kernel. Last but not least, one of the main advantages of the *SNN match kernel* is that it can be easily converted to a sparse representation as the ratio of the number of values equal to zeros is very high. Only the features $\mathbf{z}$ lying in the top neighbors of both $\mathbf{x}$ and $\mathbf{y}$ lead to consistent component values. In practice, it is therefore sufficient to consider only the top-K neighbors of each feature $\mathbf{x}$ and $\mathbf{y}$ to get a good approximation of $K(\mathcal{X}, \mathcal{Y})$. This allows using efficient nearest neighbors search techniques to construct the explicit representations $\Phi(\mathcal{X})$ and to use an efficient sparse encoding when training linear classifiers on top of them.

Approximate K-NN Search In practice, to speed up the computation of our SNN-based image representations, the ranking function $r_{\mathbf{x}}(\mathbf{z}) : \mathcal{Z} \rightarrow R^{+}$ is implemented as an approximate nearest neighbors search algorithm based on hashing and probabilistic accesses in the hash table similar to the one described in Chapter 3. More precisely, it takes as input each query feature $\mathbf{x}$ of the image I_x to be described and returns a set of K approximated neighbors in $\mathcal{Z}$ with an approximated rank $r^{K}_{\mathbf{x}}(\mathbf{z})$. The exact ranking function $r_{\mathbf{x}}(\mathbf{z})$ is simply replaced by this approximated ranking function in all of the equations above. Note that the features $\mathbf{z} \in \mathcal{Z}$ that are not returned in the top-K approximated nearest neighbors are simply *removed* from the SNN match kernels equations conducting to a considerable reduction of the computation time as they have a rank-based activation function $\varphi(r_{\mathbf{x}}(\mathbf{z}))$ equal to zero.

However, the approximate search method used in this chapter slightly differs from the one of Chapter 3. It still relies on a multi-probe strategy where we use the first bits of the binary hash codes as key entries to a hash table. These first binary components correspond to a Hamming subspace in which only the chunks (*i.e.* hash buckets) that are the most likely to contain nearest neighbors are visited. The remaining bits of the signatures are used as sub-signatures to be compared thanks to an efficient Hamming distance computation. Contrary to other binary embeddings such as Hamming embedding (with PCA-RR projections [JDS08]), ITQ [GL11] or RMMH [JB11] that try to balance the information across every binary components, what we aim to do is to get a better structure of the Hamming space by providing more information on the first binary components. While balancing the information provides robustness against the curse of dimensionality (in a raw hamming space), giving more importance to the components defining the search space (*i.e.* the buckets of the hash table) is more relevant in the context of multi-probe access in an indexing structure. Indeed, we restrict the scan of the database in most probable regions lying in a subspace that carries a great proportion of the information about the localization of the feature vectors. This is actually similar to

the clustering-based indexing methods where the coarse quantizer used to partition the space provides a great proportion of the energy of the original vectors. Thus, the hash function we used to build the hash tables relies on a Principal Component Analysis of the original space followed by a binary quantization of the b components with the higher eigenvalues. The binary quantization of each component is computed as:

$$h_j(\mathbf{z}) = sgn(\mathbf{a_j}^T\mathbf{z} - \mu_j) \tag{4.5}$$

where $\mu_j = \frac{1}{N}\sum_{\mathbf{z}} \mathbf{a_j}^T\mathbf{z}$ is the mean of all the projected values and $\mathbf{a_j}$ are the eigen vectors. The first m bits are used as key to build a single hash table. Another important theoretical justification for using such PCA-based hashing scheme is that it is consistent with the probabilistic multi-probe search model. Indeed, the model is learned on K-neighborhoods that are assumed to be generated by an axis-aligned isotropic multinomial distribution. According to our experiments, this hashing method provides better performance than the baseline of the previous chapter that uses RMMH (in terms of accuracy and computation efficiency). This is due to the high informativeness carried by the first bits produced by such PCA-like hashing scheme. Moreover, as explained by Gong et al. [GL11], PCA-hashing can be seen as a relaxed version of hashing schemes that aims to produce balanced and pairwise decorrelated binary codes. This is a desired property when we want to design efficient search algorithms in hash tables which requires to minimize the number of collisions in the buckets.

4.2.2 Spatial Consistency and Geometry

Geometrically Consistent SNN match Kernel

The problem of the classical match kernels defined in the literature is that they do not integrate the information of the relative position of the local features in the images. We therefore propose to extend our SNN match kernel Fisher Vector to include geometrical constraints on the considered matches. In this case, we consider that each local feature $\mathbf{z}$ in the training set $\mathcal{Z}$ was extracted from an image $I(z)$ at a given spatial position $\mathbf{P_z}$. A first geometrically consistent SNN kernel could then be simply constructed by adding geometrical consistency indicators in equation 4.3 such as:

$$K(\mathcal{X},\mathcal{Y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|}\sum_{\mathbf{x},\mathbf{y}}\sum_{\mathbf{z}} \delta_{\mathbf{x}}(\mathbf{z}).\delta_{\mathbf{y}}(\mathbf{z}).\varphi(r_{\mathbf{x}}(\mathbf{z})).\varphi(r_{\mathbf{y}}(\mathbf{z})) \tag{4.6}$$

where $\delta_{\mathbf{x}}(\mathbf{z})$ would be equal to one if $\mathbf{z}$ is an inlier of a RANSAC-like algorithm estimating the best affine transformation between I_x and I_z (and zero otherwise). Estimating a global affine transformation between the images might however lead to a high number of mismatches as the targeted visual patterns in fine-grained classification are typically small and the background very cluttered. It is therefore preferable

to focus on more localized geometric transformations. We therefore estimate an affine transformation for each local feature $\mathbf{x}$ (respectively $\mathbf{y}$) based on the set $\mathcal{N}_p(\mathbf{x})$ of its p spatial neighbors in I_x and their matches in I_z. The estimation of the best transformation for a given central feature $\mathbf{x}$ is performed by exhaustively probing all possible pairs of the form $(\mathbf{x}, \mathbf{x}')$ with $\mathbf{x}' = \mathcal{N}_p(\mathbf{x})$ and keeping only their best matches in the training image I_z. As our affine model is restricted to rotation and scaling transformations, each pair of matches generates a single candidate transformation parametrized by a projection matrix $\mathbf{A}_{\mathbf{x},\mathbf{x}'}(I_z)$ and a translation parameter $\mathbf{B}_{\mathbf{x},\mathbf{x}'}(I_z)$. The best transformation $(\mathbf{A}_{\mathbf{x}}(I_z), \mathbf{B}_{\mathbf{x}}(I_z))$ among the set of candidate ones is then selected as the one maximizing the number of inliers in the set $\mathcal{N}_p(\mathbf{x})$ of neighboring matches, using a threshold θ on the projected spatial positions. We finally count the number of inliers of that best transformation by weighting the contribution of each inlier match according to its rank-based activation function:

$$g_{\mathbf{x}}(I_z) = \sum_{x' \in \mathcal{N}_p(\mathbf{x})} \varphi(r_{\mathbf{x}}(\mathbf{z}')).\delta(\|(\mathbf{A}_{\mathbf{x}}\mathbf{P}_{\mathbf{x}'} + \mathbf{B}_{\mathbf{x}}) - \mathbf{P}_{\mathbf{z}'}\| < \theta) \tag{4.7}$$

where $\mathbf{z}'$ is the best match of $\mathbf{x}'$ in I_z.
We can now define the following geometrically consistent match kernel:

$$K_G(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x},\mathbf{y}} \sum_{\mathbf{z}} \delta_{\mathbf{x}}(\mathbf{z}).\delta_{\mathbf{y}}(\mathbf{z}).g_{\mathbf{x}}(I_z).g_{\mathbf{y}}(I_z).\varphi(r_{\mathbf{x}}(\mathbf{z})).\varphi(r_{\mathbf{y}}(\mathbf{z})) \tag{4.8}$$

Note that in this case, the indicator function $\delta_{\mathbf{x}}(\mathbf{z})$ is equal to one only if $\mathbf{z}$ is an inlier of the local transformation $(\mathbf{A}_{\mathbf{x}}(I_z), \mathbf{B}_{\mathbf{x}}(I_z))$ optimized on the neighborhood $\mathcal{N}_p(\mathbf{x})$. The introduction of the scores $g_{\mathbf{x}}(I_z)$ allow to give more importance to the highly consistent patches with high numbers of inliers. This embedding of the geometry of course uses an extra computation time for the construction of the explicit representation of each image (about 3 seconds).
As an improvement of $K_G(\mathcal{X}, \mathcal{Y})$, we define an alternative version based on the max pooling over the features of I_x and I_y rather than the normalized sum:

$$K_G(\mathcal{X}, \mathcal{Y}) = \max_{\mathbf{x},\mathbf{y}} . \sum_{\mathbf{z}} \delta_{\mathbf{x}}(\mathbf{z}).\delta_{\mathbf{y}}(\mathbf{z}).g_{\mathbf{x}}(I_z).g_{\mathbf{y}}(I_z).\varphi(r_{\mathbf{x}}(\mathbf{z})).\varphi(r_{\mathbf{y}}(\mathbf{z})) \tag{4.9}$$

We showed in our experiments that it can lead to better performance than $K_G(\mathcal{X}, \mathcal{Y})$. Note that this new formulation can still be factorized as $K'_G(\mathcal{X}, \mathcal{Y}) = \Phi'_G(\mathcal{X})^T \Phi'_G(\mathcal{Y})$ with the following explicit image representation:

$$\Phi'_G(\mathcal{X}) = \sum_{i=1}^{N} \max_{\mathbf{x}} \delta_{\mathbf{x}}(\mathbf{z_i}).g_{\mathbf{x}}(I_{z_i}).\varphi(r_{\mathbf{x}}(\mathbf{z_i})).\vec{e_i} \tag{4.10}$$

Spatial Pooling of the raw SNN-based image representations

As elegant as the explicit image representations $\Phi'_S(X)$ and $\Phi'_G(X)$ are, they do not conduct to good classification results in practice. Their very high dimensionality, is equal to the number of local features in the training set (often tens of millions), actually leads to strong overfitting even when using the $L2$ regularizes with high values of the regularization control parameter λ. As an illustration, the classification mAP obtained by training a logistic regression classifier from the raw feature vectors $\Phi'_G(X)$ on the Vehicle dataset introduced in [KPFJ14] ranges from 0.283 (for $\lambda = 1$) to 0.286 (for $\lambda = 100$). It is therefore required to group the individual matches of the SNN kernels before deriving an effective explicit image representation. In this work, we focus on the *spatial* pooling of the raw matches rather than aggregating them in the feature space as done in many kernels. In particular in the popular image representations such as BoW, Fisher Vectors or VLAD. We consequently loose some generalization capacity in the visual space compared to these methods but on the other side we strongly boost the locality, the interpretability and the discrimination of the trained visual patterns (particularly when using the geometrically consistent SNN kernel). Our spatial pooling strategy is very close to the one proposed in [JBJG12]. We subdivide the images I_z of the training set using spatial grids of different sizes and we aggregate the matches falling in the same cell. The main difference of our method over the one of [JBJG12] relies on the computation of the raw matching scores. Whereas they use a simple Hamming distance on the hash codes derived from the original features, our method relies on the rank-based activation function of the approximate nearest neighbors (with a quality control) combined with our geometrically consistency filtering and weighting. As discussed in [JBJG12], this spatial pooling strategy is also much more powerful than the popular Spatial Pyramid Matching of Lazebnik et al. [LSP06] although it is somehow similar. Spatially pooling the matches in the training set rather than pooling the features in the image provides much more invariance to translation and rotation while providing similar benefits in terms of partial geometry. More formally, we can reformulate our raw SNN match kernel as:

$$K_S^R(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|} \sum_{\mathbf{x},\mathbf{y}} \sum_{j=1}^{N_R} \sum_{\mathbf{z}\in Z_j} \varphi(r_{\mathbf{x}}(\mathbf{z})).\varphi(r_{\mathbf{y}}(\mathbf{z})) \tag{4.11}$$

where Z_j is the j-th image region of the whole training set containing in total N_R regions (e.g. for a regular grid subdividing each image of the training set in 4 cells, N_R is equal to the number of training images multiplied by 4). This reformulation allows deriving an alternative explicit image representation at the region level rather than at the feature level:

$$\Phi_S^R(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{j}=1}^{N_R} \frac{1}{|\mathcal{X}|} \sum_{\mathbf{z}\in Z_j} \sum_{\mathbf{x}} \varphi(r_{\mathbf{x}}(\mathbf{z_i})).\vec{e_j} \tag{4.12}$$

Kernel type	without geometry	with geometry
Accuracy	81.4	90.5

TABLE 4.1 – Cross-validation performance on the *FlickersLogos32* dataset.

Training a linear classifier on such representation will affect weights to the regions in the training set rather than to the local features themselves.

A similar image representation can be derived from the geometrically consistent kernels but we prefer to use a max pooling strategy to aggregate the raw matches falling in the same cell as we discovered that this was more beneficial in that case. The explicit image representation derived from the SNN kernel $K'_G(\mathcal{X}, \mathcal{Y})$ mapped onto an image grid R is then:

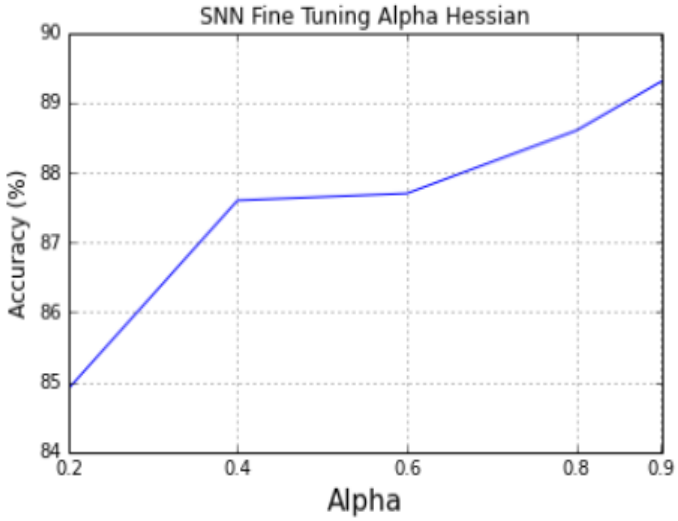
$$\Phi^R_G(\mathcal{X}) = \sum_{j=1}^{N_R} \max_{\mathbf{z} \in Z_j} \max_{\mathbf{x}} \delta_{\mathbf{x}}(\mathbf{z_i}).g_{\mathbf{x}}(I_{z_i}).\varphi(r_{\mathbf{x}}(\mathbf{z_i})).\vec{e_j} \tag{4.13}$$

In our experiments we used a simple pyramidal partition of the images recursively subdividing the image in 4 regular cells, up to a depth d_p. The image representations of each level of the pyramid are simply concatenated before being passed to the linear classifier. We denote the resulting representation as $\Phi_G^{0,1,2,\ldots,d_p}(\mathcal{X})$. Note that $\Phi^0_G(\mathcal{X})$ is the representation at the image level, without any subdivision of the image. In this case the dimensionality of the representation is equal to the number of images in the training set and the classifier learns contributive weights of the images for each class.

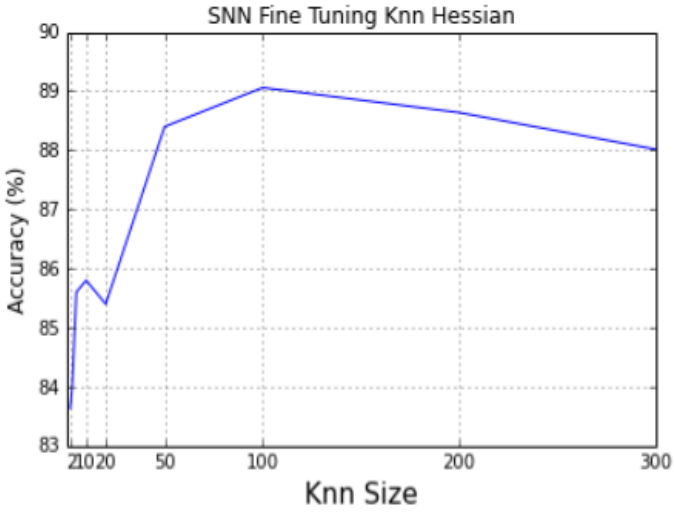
4.3 Parameters study

In this section we present the impact of the different parameters of our approach through a cross-validation procedure on the FlickrsLogos32 [RPLvZ11] dataset. We study the impact of using geometry or not in our SNN match kernels, the impact of the parameter k (the per descriptor k-nn size), the impact of the quality search parameter of the approximate nearest neighbors search procedure (α), and, last but not least, the impact of the pyramid depth d_p. Default values of the parameters when studying the others were $\gamma = 1$, $d_p = 0$, $kerneltype = geom$. Table 4.2 and Figure 4.3 display the results obtained on the *FlickerLogos32* dataset.

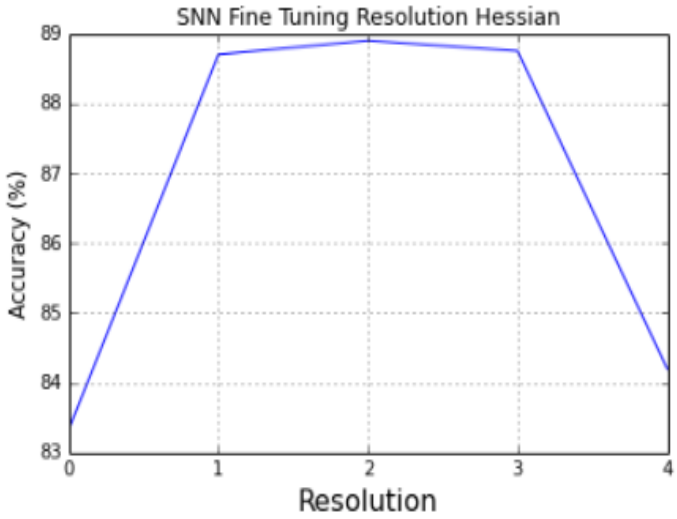
Impact of the k-nn size K (Figure 4.3b) This parameter reflects how far in the feature space we will look for neighbors for each local descriptor of the image to be represented. This can be seen as a kind of window of a particular size (a receptive field) that we put around each local descriptor in the feature space where every vectors that are beyond this window will not participate to the representation. An important thing that we need to keep in mind while analyzing this plot is that



(a) Alpha



(b) Knn



(c) Resolution

FIGURE 4.3 – Parameters study.

this similarity radius is based on a topological information, the rank (cf Equation 4.3), rather than a more traditional Euclidian distance. This approach provides us a more robust similarity evaluation to fancy distributions as explained in section 4.2.1. Here we can see that the choice of this parameter has a non negligible influence in the classification performance. If the value of K is set to a low value, then we do not bring as much information as we would need. This results in a poorly informative representation that do not encode enough variation in the data. On the opposite, if we set K to a too high value, then we start bringing irrelevant and noisy information in the representation and the classification accuracy starts dropping.

Impact of the approximate nearest neighbor search quality control α (Figure 4.3a) While the K-nn size reflects the size of the topological window used to tile the feature space, the quality parameter α reflects how accurate we are when retrieving the K-nearest neighbors, *i.e.* how noisy is the retrieved information. The results here are not very surprising as they suggest that the more accurate we are when retrieving the K-nn of each point, the more the representation is discriminant and allows good classification result. On the other hand, the computation time gets higher when considering high values for α which makes it very hard to use values of $\alpha > 0.9$.

Impact of the geometry (Table 4.2) The results of Table 4.2 on the choice of the SNN kernel clearly demonstrates the interest of taking into account the local geometry in the final representation space. As we will also see it later in section 5.1.1, this kind of information is particularly suited for fine-grained classification tasks that involve rigid objects.

Impact of the spatial pooling resolution r (Figure 4.3c) Another important parameter of our system corresponds to the size of the sub-regions of the training image where the localized geometrically consistent information are pooled. As said in section 4.2.2, this spatial pooling is the key point to avoid overfitting that would be caused by a too high dimensional SNN representation. Here, we see that a value of 0 provides significant loss of classification accuracy as it amounts to pooling the geometric information globally in each training image and thus the model loses its capacity to finely encode spatially localized information. On the other hand, if we set the resolution to a too high value, we tend to come back to the original problem of the raw SNN formulation and it starts overfitting when we increase the dimensionality of the representation. This experiment clearly demonstrates the interest of such aggregation process in the raw similarity-based feature space. It somehow *forces* some structure in the final representation space where the particularity here is the spatial nature of the pooling cells. We are convinced that this spatial components pooling strategy and the integration of geometric information is a key point to capture good invariant in the data.

4.4 Comparison to the Matching-based Spatially Consistent KNN classifier of Chapter 3

In this section we compare our method with the classification results reported in the previous chapter using our baseline methods on *FlickrLogos32* and *Vehicles29* datasets. We used the same features extraction procedure as the one used in the previous chapter for those datasets. SIFT descriptors [Low04] are extracted from points detected by the rotation-invariant Harris-Hessian-Laplace interest point detector [MS05]. Following prior works [AZ12], the descriptors are L_2-normalized to the unit ball and square-rooted. For the SNN representation, we took the best evaluated parameters in the cross validation step. For the ANN search scheme, we used a size of $K = 100$ and we set the quality control parameter $\alpha = 0.8$ as it provides sufficiently close results to the best cross validated value while keeping satisfying computation time. We only used the geometrically consistent SNN kernel and the depth of the spatial pyramid is set to $d_p = 2$. Finally, we used the logistic regression classifier of LibLinear with a L_1 regularization on top of our SNN representations ($\Phi_G^{0,1,2}(\mathcal{X})$). The classifier is parametrized by the default contribution factor value $\lambda = 1$. The used evaluation metric is the IC-mAP as in the previous chapter.

Method	FlickrL32	Vehicles29
Fisher Vector 128x4,096 [KPFJ14]	0.866	0.497
Prototype voting [KPFJ14]	0.914	0.557
SC KNN classification	0.928	0.597
Our method	**0.981**	**0.79**

TABLE 4.2 – Classification performance on the *FlickrLogos32* and *Vehicles29* datasets measured by IC-mAP.

Results are reported in table 4.2. They show that our method definitely outperforms our previous baseline as well as the Fisher Vector representation evaluated in [KPFJ14] or the prototype voting scheme they introduced. We achieve respective classification scores of 0.981 and 0.79 on the *FlickrLogos32* and *Vehicles29* datasets. Note that for some test images, the background is highly cluttered and the objects of interest rather small and slightly deformed. The impressive performance gap of increase of 0.20 points on the *Vehicles29* dataset clearly demonstrates the validity of our hypothesis in the previous chapter, *i.e.* it allows us to learn how to combine geometrically consistent visual patterns to discriminate different classes.

4.5 Discussions

4.5.1 Discussion on the complexity and memory

The complexity for the knn search procedure is $O(N^{1+\rho})$ (with $\rho < 1$ according to [JB08]) and the complexity of the construction of the explicit representation computation is $O(N * k)$ because we iterate over the k nearest neighbors of each points. The total complexity of the algorithm for the training phase is then $O(N*k+N^{1+\rho})$ and $O(N_{test}*k+N^{1+\rho})$ for the testing phase (with N and N_{test} the respective number of local features in the training and testing dataset) leading to a practical computation time of about **5** seconds per image (on a Bi-Xeon(R) E5-2650 CPU 2.00GHz) to compute its SNN explicit representation without the geometry embedding and **9** seconds with it. Although this method can be considered as scalable when we look at the number of descriptors to be searched as nearest neighbors for each point of each image, it still implies too high prediction time and high memory cost as we have to store all of the compressed vectors of the training set in RAM (about 10^8 descriptors for usual classification tasks). In Chapter 6, we address this memory and computation time issue by using a supervised dimensionality reduction approach on top the SNN representations. We will show how this allows us to highly compress the size of the initial vocabulary while keeping very satisfying classification performance and while learning well interpretable patterns.

4.5.2 Links with coding schemes and aggregation methods

A very interesting aspect of the raw SNN representation (cf. section 4.2.1) relies on the fact that it shares some common properties with soft assignment coding and aggregation schemes. Our approximate K-nn search model based on probabilistic multi-probe accesses in hash tables allows us to select the most probable regions of the feature space for each local feature. This can actually be considered as a fast soft assignment over a huge vocabulary composed of every feature in the dataset (between 100,000,000 and 1,000,000,000 in our experiments). This is analogous to the optimization procedure of coding based schemes when inferring codes. Then, as shown by equation 4.3, all of the encoded local features are spatially aggregated through a sum pooling operation over the whole image to build a global representation such as in BoVW or Fisher Vectors and VLAD. Actually, Locality-constrained Linear Coding (LLC) [WYY+10] (cf. section 2.3.2) is the coding scheme most closely related to our method as it consists in representing a feature vector with nearby words belonging to a visual vocabulary. The major difference is that our inference procedure is not based on an optimization procedure that tries to decompose a vector into a linear combination of visual words so as to minimize a reconstruction error. Instead, it is based on representing the data onto a large spatially localized vocabulary where each visual word corresponds to a particular image's region. This is what makes our representation more related to Similarity-Based Embedding. However, as we have

seen in section 2.5.2, such kind of methods often consider image level similarities rather than finer representation based on feature-level similarities as we do for the raw SNN representation. In this work, we rather choose to force some spatial structure in the raw representation space with the use of group component aggregation. Indeed, as explained in section 4.2.2, as we do not learn local *abstractions* (*i.e.* maxima or modes of the distribution of the data), we loose some generalization capacity in the visual space compared to traditional aggregation-based methods (such as BOVW or Fisher Vectors). However, considering such optimization-free exhaustive vocabulary can be seen as the use of the whole empirical data manifold as a support to build image representations. The main advantage of our method is that it offers a high quality control for the approximate NN search procedure which allows us to control how fine we want to model this manifold.

4.6 Conclusion

In this chapter, we introduced a new match kernel based on shared-nearest neighbors and localized geometric information. We derived an image representation corresponding to the explicit embedding of this newly introduced kernel. Each image is densely described by local handcrafted descriptors and each descriptor is encoded by a vector containing its similarity with the descriptors of the training database. Similarly to traditional coding and aggregation-based schemes, we proposed a first approach to build a global image representation by aggregating all the previously computed similarity-based encoded version of the local descriptors of the image to be represented. Their very high dimensionality, equal to the number of local features in the training set (often tens of millions), actually leads to strong overfitting and a lack of spatial invariance.

To overcome this overfitting problem and to gain more spatial invariance in the final embedding, the components corresponding to similarities of local descriptors belonging to the same local image region of the training set are aggregated into a single component. This leads to a representation where each component corresponds to the similarity computed between the image and the set of training image regions obtained by regularly subdividing them at a particular resolution. This finally leads to a representation whose dimension is equal to the number of such training image patches (comparable to a visual vocabulary where each word now corresponds to a spatially localized image region). Even if they seem very similar because they both consider spatial aggregation, this spatial pooling approach provides much more spatial invariance than the popular SPM because, in our case, the spatial aggregation is performed on the training images rather than on the image to represent. Thus, different images showing two objects at different locations will still activate the same spatially consistent components of the representation. This local aggregation is then followed by a geometrical consistency check that allows localized robustness

to viewpoint changes. We show that our approach achieves high performance gains on previously considered classification tasks.

In the following chapter, we are going to evaluate the performance of our new representation by comparing it with state-of-the-art approaches on several fine-grained classification tasks. We also investigate the behavior of our approach when using several underlying feature schemes such as handcrafted color descriptors and CNN features. Finally, we also introduce a temporal extension of this shared nearest neighbor representation and we show that it provides significant results on a bird song identification task.

Chapitre 5

Shared Nearest Neighbors Representation Experiments

Contents

In this chapter, we present the experimental results of the Shared Nearest Neighbors representation by comparing it to other fine-grained classification methods. To this aim, we evaluate our approach on several fine-grained classification tasks involving different kinds of objects such as: aircrafts, bird species that appear in highly cluttered background and look less like rigid objects, or flower species whose discriminant attributes are mostly based on color information. Then, we investigate the

behavior of our method by using several underlying feature schemes (such as hand-crafted color descriptors or off-the-shelf Convolutional Neural Network features) and we compare our results to state-of-the-art CNN architectures. Finally, we demonstrate the relevance of such representations on fine-grained audio classification tasks with a temporal extension of the original SNN representation by notably integrating audio features in our pipeline.

5.1 Evaluation of the SNN representation on fine-grained classification tasks

In this section we evaluate the performance of our approach on several fine-grained classification tasks and compare it to state-of-the-art methods. We first present the different settings that we use to evaluate our approach. Then, we compare our method to the fine-grained classification results reported in Gosselin et al. [GMJP14] on the *OFGVC-Aircraft* [MKR+13] dataset. We also evaluate the SNN representation on other fine-grained classification tasks (the *CUB-Birds-200* dataset [WBW+11] and the *OxfordFlower102* dataset [NZ08]) where color information play an important role to discriminate between the classes. The evaluation metric for all experiments is the top-1 classification accuracy.

5.1.1 Settings

Pre-processings and Local Features For all the experiments presented in this section, we used the features extraction procedure used in [GMJP14] so as to conveniently compare our SNN representation with the Fisher Vector that was reported as the state-of-the-art method on the different fine-grained classification tasks. The feature extraction pipeline consists in the following steps:

- All the images were first down-sampled to a fixed size of S pixels while keeping their original aspect ratio and the images whose number of pixels was smaller than S were not modified. The training datasets are also augmented by flipping each image along the vertical axis resulting in twice more images.
- SIFT descriptors was then densely extracted in a multi-resolutional regular grid where the maximum resolution and step size on the x-axis and y-axis are respectively set to 5, 3 and 3.
- A low energy filtering strategy was applied in order to eliminate uninformative descriptors from the database. More concretely, the L_2 norm was computed and the patches whose norm is below a predetermined threshold τ were removed. For gradient orientation histogram descriptors such as SIFT, this allows us to remove uniform patches that are not likely to contain useful information for the final representation.
- Each local descriptor was then post-processed by using a component-wise non-linear change of variable so-called *RootSIFT* [AZ12] which simply consists

in replacing each component by its square-root.
— Finally the descriptors were L_2-normalized.

Contrary to ours, the feature extraction process applied in [GMJP14] also integrates a dimensionality reduction based on PCA before the *root-sifting* step. The main reason for that difference is that our approximate nearest neighbor procedure already performs a PCA embedding to compress the features. The subtle difference is that we perform the PCA embedding after the *Root-Sift* procedure rather than before.

Shared Nearest Neighbor Embedding The following steps of our classification pipeline simply consists in computing, for each image, the SNN representation as explained in the previous chapter. We fix all the parameters according to our previous cross-validation experiments of section 4.3. The number of neighbors K returned by the approximate knn search method is fixed to $K = 100$, the quality control parameter of the approximate search is fixed to $\alpha = 0.8$, the depth of the pyramid is set to $d_p = 2$. As suggested in [KPFJ14], we also study in our cross-validation experiments the impact of a power-law normalization of our representations, which is a popular post-processing stage applied after coding schemes such as BOVW or Fisher Vectors [PD07]. This normalization is parametrized by a single parameter γ applied to all components such that:

$$x_i^{j\prime} = sign(x_i^j)|x_i^j|^\gamma \tag{5.1}$$

It usually ensures that infrequent (yet potentially highly informative) patches are not overpowered by frequent (yet not necessarily informative) patches such as uniform background patches. The power-law normalization factor is set to $\gamma = 0.6$.

SNN kernel combination with an early fusion scheme In many classification problems, it is difficult to predict whether the rigid local geometry will help or not. We therefore combine the geometry-free and the geometrically consistent match kernels by simply concatenating their representations Φ^R_{SNN} and Φ^R_G. Thus, each image region of the training set is represented by two complementary scores (with or without geometry). We actually make use of the sparse logistic regression model of the LibLinear [FCH+08] library because it somehow forces the classifier to choose which kind of information contributes the most to the recognition by using a reduced subset of the parameters to be different from zero. For the following experiment, we evaluate our methods on both the non-geometric and geometric SNN kernel as well as the combination of the two kernels.

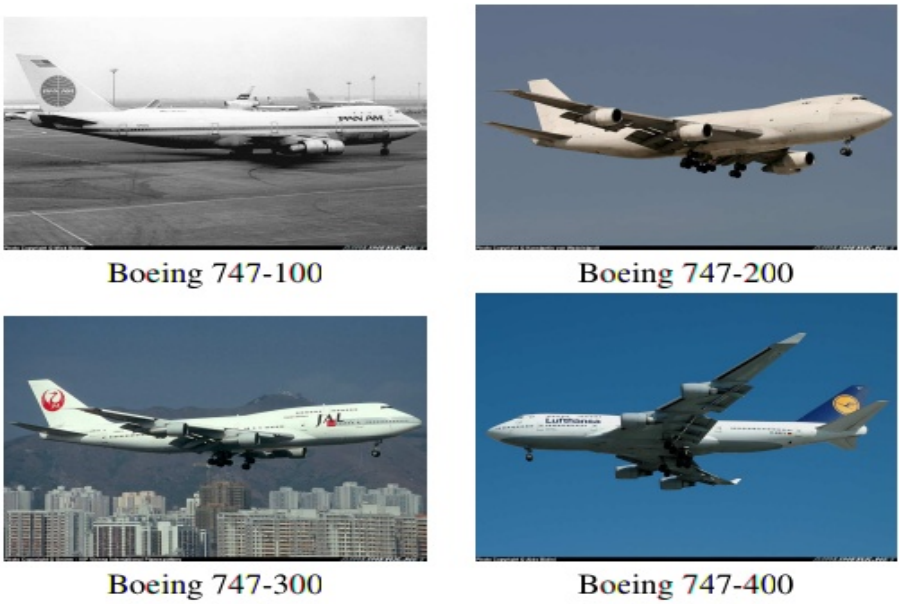


FIGURE 5.1 – Image samples of the *FGVC-Aircraft* dataset. Figure taken from [GMJP14].

5.1.2 Fine-grained classification results

We first compare our method to the fine-grained classification results reported in [GMJP14] on the *OFGVC-Aircraft* dataset [MKR+13] which is illustrated in Figure 5.1. This dataset is composed of 10,000 photographs of aircrafts labeled with 100 models that are very specific such as $A340-200$, $A340-600$ or $Boeing-737-400$. The dataset is split into $3,333$ images for the test set and $6,667$ images for the training set.

Method	*Top-1 Accuracy*
CafeNet	78.85
Fisher Vectors S_A	75.88
Fisher Vectors S_B	80.5
Fisher Vectors S_A + S_B	81.46
Our method with non-geometric Kernel	76.5
Our method with geometric Kernel	79.7
Our method with early fusion	**82.8**

TABLE 5.1 – Classification performance on the *FGVC-Aircraft* dataset measured by Mean Accuracy.

In table 5.1, we report our final results compared to the best three runs reported in [GMJP14] on the *FGVC-Aircraft* dataset. It shows that our method outperforms

the heavily tuned Fisher Vector configurations of [GMJP14] as well as the deep Convolutional Neural Network of [KSH12] used in the CafeNet run. From table 5.1, we see that, the use of the geometric kernel gives better results than the use of the non-geometric kernel. This is mainly because aircrafts correspond to rigid objects involving rigid patterns which visually describe them. However, it is interesting to look at the result obtained by the non-geometric kernel that is not so bad compared to the geometric one while being less heavy in terms of computation time. This means that raw visual matches and localized spatial pooling bring enough information to solve fine-grained classification problems. The result obtained by the early fusion scheme between non-geometric and geometric kernel clearly shows that geometric and non-geometric information are complementary. Then, the combination of these two kinds of information leads to better classification performance. One can also notice that only the Fisher Vector configuration S_B uses the same visual features as ours whereas the configuration S_A (and consequently $S_A + S_B$) uses some color features additionally. The real comparison of our method to the fisher vector encoding scheme should therefore rather be restricted to S_B. It is also important to note that the CafeNet run extensively made use of external training data (*i.e.* ImageNet data) to pre-train the deep convolutional neural network model described in [KSH12].

Considering the results on the *FGVC-Aircraft* dataset on Table 5.1, we see that color information does not play an important role to discriminate between the different classes for this dataset. Indeed, the Fisher Vectors using color descriptors do not provide classification performance as high as the one obtained with the Fisher Vectors using SIFT descriptors. Thus, we also investigate the behavior of our method on other fine-grained classification tasks whose discriminant attributes are rather based on color information such as the *CUB-Birds-200* dataset and the *OxfordFlower102* dataset:

- ***CUB-Birds-200*** [WBW+11] (Fig 5.2a): It contains 11,788 images of 200 bird species. 5994 images are used for training and 5794 for evaluation. Many of the species in the dataset exhibit highly subtle differences which are sometimes even hard for humans to distinguish. Multiple levels of annotation are available for this dataset: bounding boxes, part landmarks, binary attributes and boundary segmentation.
- ***OxfordFlowers102*** [NZ08] (Fig 5.2b): This dataset contains 102 categories each consisting of 40 to 258 images. The flowers appear at different scales, pose and lighting conditions. Furthermore, the dataset provides segmentation for all of the images.

Settings In the following experiments, we only use the image label information of these two datasets during training and testing. To compute the SNN representations, we use the same protocol than the one used for the experiments on the *FGVC-Aircraft* dataset. For the Fisher Vectors experiments, we take the fine-grained

classification results on these two datasets reported in Murray et al. [MP14]. Particularly, we take the results they obtained with their *Generalized Max Pooling* Fisher Vector applied on top of SIFT descriptors and X-Color descriptors (the same color descriptors as those used for the *SA* system whose results are reported in Table 5.1). For the CNN experiments, we use our own run performed with the C++ Caffe library [JSD+14] using the *AlexNet* architecture described in [KSH12] with the same parameters. As for the results reported on Table 5.1, we also make use of ImageNet data to pre-train the ConvNet.

Method	*CUB-Birds-200* [WBW+11]	*OxfordFlowers102* [NZ08]
AlexNet [KSH12]	**53.5**	**85.3**
Fisher Vectors (SIFT) [MP14]	17.0	73.3
Fisher Vectors (Color) [MP14]	29.3	75.1
Fisher Vectors (Fusion) [MP14]	**33.3**	**84.6**
SNN with non-geometric Kernel	21	72.2
SNN with geometric Kernel	**33.3**	**74.6**

TABLE 5.2 – Classification performance on the *CUB-Birds-200* dataset and the *OxfordFlower102* dataset measured by Mean Accuracy.

In Table 5.2, we report our results on the *CUB-Birds-200* dataset and the *OxfordFlower102* dataset. Once again, while our results should only be compared to the Fisher Vectors runs that do not make use of color descriptors, we show that our method provides competitive results with the Fisher Vectors even by only considering SIFT descriptors. We notably show the relevance of using geometric information on the *CUB-Birds-200* dataset where we increase the top-1 classification accuracy of 12 points. We show that the CNN model significantly outperforms the others methods on these datasets. This is mainly due to the good generalization ability of such methods when learning from models that were previously pre-trained on a huge set of external data. It is important to note that our runs, as well as the Fisher Vectors runs, do not make use of such fine-tuning procedure which provides significant advantages for methods based on CNN. This is particularly true in this experiment where the ImageNet training dataset highly intersects in terms of semantic contents with both the *CUB-Birds-200* and the *OxfordFlower102* datasets. This results in a CNN architecture with visual features that are adapted to the task. As we can see on Figure 5.3, the low-level features learned by the CNN well encode both shape and color information. We see that the use of color descriptors for the Fisher Vectors representation provides a significant increase of 12 points of top-1 accuracy for the *CUB-Birds-200* dataset. Not surprisingly, combining two feature schemes allows increasing the classification accuracy of about 9 points more. This demonstrates some complementarity between these two kinds of information. These results suggest that considering other feature schemes than purely appearance-based descriptors is likely to provide significant improvements on the performance we could

(a) Birds [WBW+11]

(b) OxfordFlower102 [NZ08]

FIGURE 5.2 – Fine-grained classification datasets.

obtain on such fine-grained classification tasks.

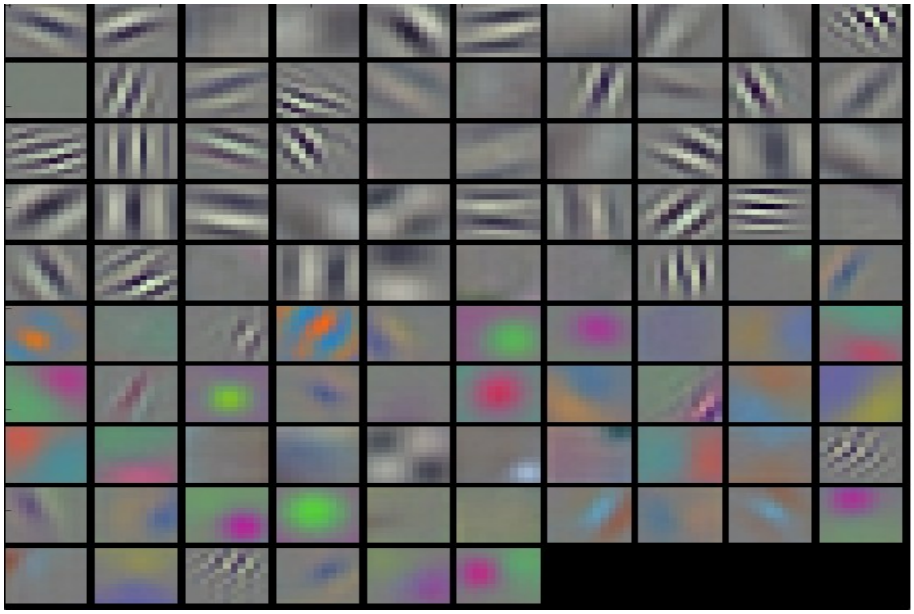


FIGURE 5.3 – Low level features learned by the AlexNet CNN architectures.

5.2 Study of the impact of different underlying feature schemes

In this section, we investigate the behavior of the SNN representation by using it on top of two different feature schemes. We first consider using handcrafted color descriptors as it should allow improving our results on some fine-grained classification tasks (such as those studied in the previous section). Then, we consider CNN features that are learned in a supervised manner on the task. The following experiments are led on *CUB-Birds-200* and *OxfordFlower102* because they correspond to the fine-grained datasets whose we expect the performance are most likely to be improved by integrating such feature schemes.

5.2.1 Handcrafted Color Features

In the following experiments, we consider the same protocol than the one used for the previous section for the extraction of the SNN representations. The major difference relies on the fact that we use handcrafted color descriptors rather than SIFT descriptors.

Color descriptors We chose to use off-the-shelf *RGB-SIFT descriptors* [VDSGS10] extracted with the software of Koen Van de Sande [1]. This feature scheme consists in computing traditional SIFT descriptors for each *RGB* channel independently and by subsequently concatenating them to form 384-dimensional local features. These features are then L_2 normalized. Considering the results reported on Table 5.2 and Table 5.1, we only evaluate the impact of the color descriptors by using on top of them the geometric SNN kernel as it provides the best classification performance.

Late fusion scheme As suggested by the results of the Fisher Vector run reported on Table 5.2, we also investigate the combination of both *appearance-based* and *color-based* SNN representation respectively denoted by $\Phi_{SNN}^{SIFT}(.)$ and $\Phi_{SNN}^{Color}(.)$. To this aim, we use a similar late fusion scheme as the one used in [MP14, GMJP14] by simply computing a linear combination of the two representations:

$$\Phi_{SNN}^{SIFT+Color}(\mathcal{X}) = (1-p)\Phi_{SNN}^{SIFT}(\mathcal{X}) + (p)\Phi_{SNN}^{Color}(\mathcal{X}) \tag{5.2}$$

where $0 \leq p \leq 1 \in R$ is a weighting term giving more importance to the $\Phi_{SNN}^{SIFT}(.)$ representation if it is close to 0 or more importance to the $\Phi_{SNN}^{Color}(.)$ representation if it is close to 1. To evaluate the performance of the $\Phi_{SNN}^{SIFT+Color}(\mathcal{X})$ representation, we take 25% of the images of the *CUB-Birds-200* training set to form a validation set and the remaining images are used for learning. Figure 5.5 shows the cross-validation performance obtained for different values of p.

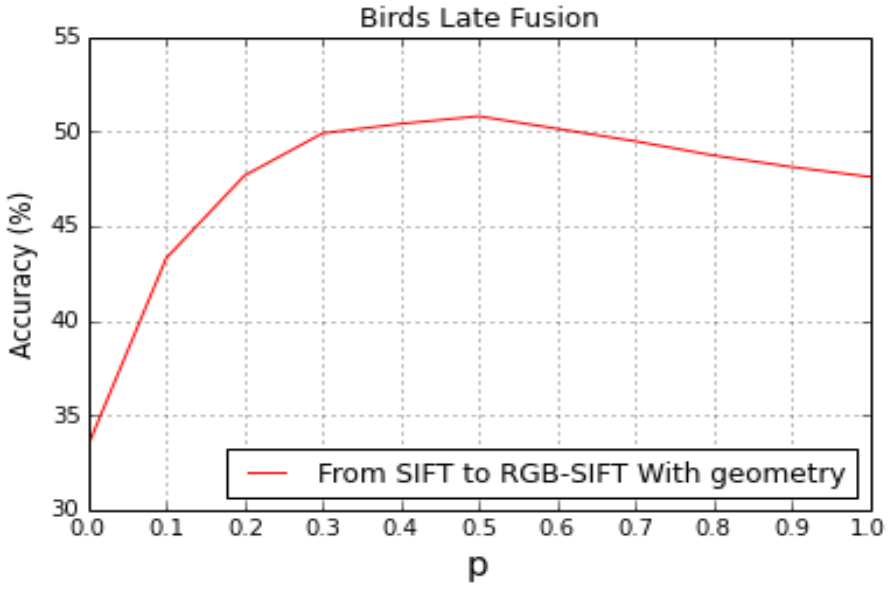


FIGURE 5.4 – Late fusion between two SNN representation respectively applied on top of RGB-SIFT and SIFT features on the *CUB-Birds-200* dataset.

1. http://koen.me/research/colordescriptors/readme

Method	*CUB-Birds-200* [WBW+11]
SNN (SIFT)	33.3
SNN (Color)	47.6
SNN (SIFT+Color)	**50.3**

TABLE 5.3 – Evaluation of the SNN representation using handcrafted color descriptors on the *CUB-Birds-200* dataset. The SNN representation have been computed using the geometrically consistent kernel.

Results The obtained results on the *CUB-Birds-200* dataset are reported on Table 5.3. We see that using color information provides a significant gain of 14 points on top-1 classification accuracy. Figure 5.4 shows that combining both information provides better cross-validation performance than using either only one feature scheme. We see that combining both information provides an additional improvement of about 3 points of top-1 classification accuracy when considering a weighting factor $p = \frac{1}{2}$.

5.2.2 Supervised Deep Learning Features

As we have seen in the previous section, using color descriptors in the SNN representation provides better fine-grained classification results on datasets where color is a discriminant information between the classes. In this section, we are interested in using data-dependent local features rather than handcrafted ones, *i.e.* feature vectors that have been learned on the task to extract relevant attributes to discriminate the classes. We expect that such kind of features would allow to automatically capture which kind of information (*i.e.* opponent colors, oriented gradient, etc.) is relevant for a particular classification task. For instance, for the *CUB-Birds-200* dataset, the supervised model should learn that features carrying color information should be more adapted than appearance-based information. In the following experiments, we consider supervised CNN architectures to learn such kind of features. As we have seen in section 2.4.2, such deep learning models are able to learn progressive embedding from the raw pixel space to more and more abstract representation space until the label space.

CNN features settings We perform the same image pre-processing as those used in the previous sections (image down-sampling and mirroring). We use the Caffe C++ Convolutional Neural Network implementation with the GoogLeNet architecture [SIV16] (illustrated on Figure 2.8b). We fine-tune a model pre- trained on the ImageNet dataset [DDS+09] with a learning rate value set to 0.001 and local multipliers set to 10 on the last fully connected layer. The weight decay and momentum parameters are respectively set to 0.0005 and 0.9. Once the model is learned, we use it as a dense feature extractor by forwarding the images to different layers of the model according to which level of abstraction we want the local features to have.

More concretely, for each location of the image, we take the activation values across all the feature maps to get a feature vector whose dimension is equal to the number of filters of the layer. Table 5.4 summarizes the different output configurations of the GoogLeNet architectures. As explained in section 2.4.2, the pooling layers reduce by a factor 4 the resolution of the feature maps from a layer to the next one. Then, the higher the layer's level, the lower the number of extracted features. For example, if we choose to extract CNN features corresponding to the *inception_3a* layer, we obtained 784 densely sampled 256-dimensional local features for each image.

Impact of the layer's level and the geometry Figure 5.5 shows the results of the cross-validation experiments we led on the *CUB-Birds-200* dataset. We evaluate the top-1 cross-validation accuracy of the SNN representations (with and without the geometrically consistent kernel) build on top of CNN-based features extracted from different layers. An interesting property we can note is that the performance first increases when considering higher level features and it starts decreasing when the number of features to be pooled in the SNN vector becomes too low. The best performance is reached when considering a good trade-off between the number and the level of information of the local features to be aggregated. This demonstrates a common property that share most aggregation-based methods: they need a lot of local features to have good encoding properties. Moreover, we see that for low-level CNN features (until layer 3), the geometry provides significantly better results than when we do not make use of it. However, we observe that the geometry becomes less important as we increase the level of the layer from which we extract the local features. This let suggest that higher-level features provide good combinations of localized lower-level features.

Layer id	Layer name	output spatial size	#features	feature space dimension
1	*conv1/7x7_s2*	112x112	12,544	64
2	*conv2/3x3*	56x56	3136	192
3	*inception_3a/output*	28x28	784	256
4	*inception_4a/output*	14x14	196	512
5	*inception_5a/output*	7x7	49	1,024

TABLE 5.4 – Output configurations of the different layer of the GoogLeNet CNN architectures that we used for feature extraction.

Method	*CUB-Birds-200* [WBW+11]	*OxfordFlower102* [NZ08]
SNN (SIFT)	33.3	74.6
SNN (CNN)	**48.5**	**90.2**

TABLE 5.5 – Evaluation of the SNN representation using handcrafted CNN descriptors on the *CUB-Birds-200* dataset and the *OxfordFlower102* dataset. The SNN representation have been computed using the geometrically consistent kernel.

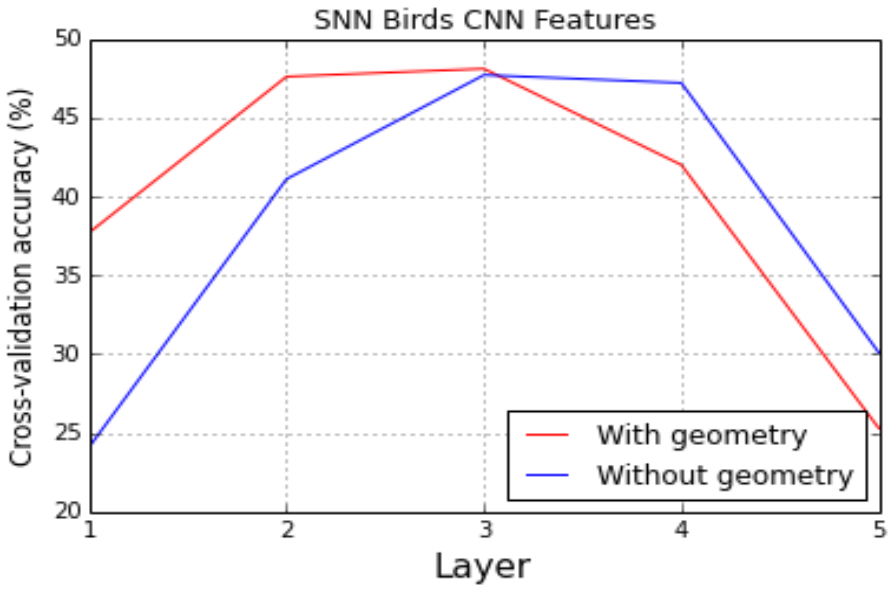


FIGURE 5.5 – CNN features from different layers of the GoogLeNet architecture pretrained on ImageNet.

Classification results For the following experiment, we follow the results of the cross-validation experiments described above to choose the CNN features and the SNN settings. We use CNN features extracted from the *inception_3a* layer, leading to 784 densely sampled 256-dimensional local features for each image. We then build our SNN representation using the geometrically consistent kernel. Then, we evaluate the classification performance on the *CUB-Birds-200* dataset and report our results on Table 5.5. We see that using CNN features improve our previous baseline for the *CUB-Birds-200* dataset. We now exhibit fine-grained classification results that significantly outperform the best runs of Fisher Vectors reported in [MP14] and in Table 5.2 on the *CUB-Birds-200* and the *OxfordFlower102* datasets.

5.3 Comparison to state-of-the-art ConvNet architectures

In this section, we compare our approach with more recent Convolutional Neural Network architectures that are now recognized as state-of-the-art methods on many classification tasks. As explained in the state-of-the-art section of this thesis, the good performance of ConvNet models on such a number of image classification tasks mainly arise from the availability of huge labelled datasets such as ImageNet [DDS+09] from which we first can learn the model before fine-tuning the parameters on the targeted classification task. As we do not make use of such fine-tuning process in our classification scheme, we are here particularly interested in comparing our

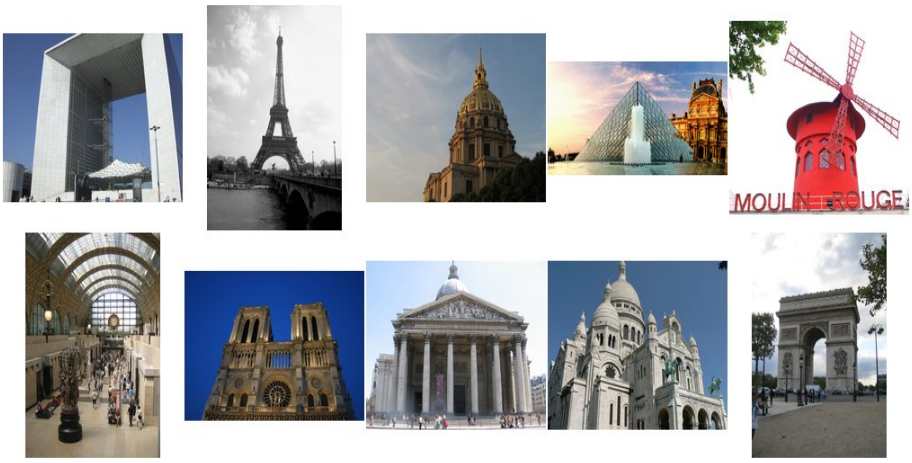


FIGURE 5.6 – Image samples of the ParisBuilding6k dataset [PCI+08].

results with CNN models learned without fine-tuning. In this section, we focus our comparison with respect to the popular GoogLeNet [SIV16] architecture.

5.3.1 Datasets

Table 5.6 lists the dataset we study in this section and summarizes for each of them the total number of images, the number of training images, testing images, and classes. Contrary to the experiments of Chapter 3, the *FlickrsLogos32* dataset is used without the distractor images. This leads to a dataset composed of 2,240 images labeled with 32 logo classes (split into 1,280 training images and 960 test images). We also evaluate our approach on the *ParisBuildings6k*[PCI+08] dataset that consists in 6,392 photographs of labeled with 12 Parisian buildings. This dataset has been originally designed to evaluate image retrieval performance and is, thus, not explicitly composed of a training and testing dataset. Then, we split it into 3,199 training images and 3,193 test images. Figure 5.6 illustrates some examples of this dataset. As we can see in Table 5.6, we compare our approach to Convolutional Neural Networks models on datasets with varying training set sizes involving different kind of objects (*e.g.* small objects, rigid objects, object classes with high or low intra-class variability, etc.).

Dataset	#images	#test images	#training images	#classes
FlickrsLogos32 [RPLvZ11]	2,240	960	1,280	32
Vehicles29 [KPFJ14]	10,622	5,356	5,266	29
FGVC-Aircrafts [MKR+13]	3,333	6,667	10,000	100
CUB-Birds200 [WBW+11]	5,794	5,994	11,788	200
OxfordFlowers102 [NZ08]	6,129	2,040	8,169	102
ParisBuildings6k [PCI+08]	3,193	3,199	6,392	12

TABLE 5.6 – Statistics of the datasets.

5.3.2 Settings

SNN representation settings We use the same protocol than the one used in the previous sections except that we only consider the geometrically consistent SNN kernel. We use different underlying feature schemes according to the best results obtained through cross-validation experiments that we led on the different datasets. For the *FlickrLogos32*, *Vehicles29* and *ParisBuildings* datasets, SIFT features were extracted around Harris-Hessian-Laplace interest points. For the *OFGVC-Aircraft* dataset, we use SIFT descriptors densely extracted as explained in section 5.1.1. For the *CUB-Birds-200* dataset and the *OxfordFlower102* dataset we extract CNN features corresponding to the *inception_3a* layer, we obtained 784 densely sampled 256-dimensional spatially localized features for each image.

Convolutional Neural Networks settings We perform the same image pre-processing as those used in the previous sections (image down-sampling and mirroring). We use the Caffe C++ Convolutional Neural Network implementation with the GoogLeNet architecture [SIV16]. We evaluate the classification accuracy on both fine-tuned and non-fine-tuned CNN models with respective learning rate of 0.001 with local multipliers of 10 on the last fully connected layers and a learning rate of 0.01 without adding additional local multiplier. The weight decay and momentum parameters are respectively set to 0.0005 and 0.9.

5.3.3 Classification results

From Table 5.7, we can conclude that, although state-of-art-CNN architectures give better results than our approach on several tasks, our method outperforms CNN models on datasets where the number of training samples is low or when the objects are small and / or rigid (as the *FlickrsLogos32* and the *ParisBuildings* datasets). This suggests that our geometrically consistent SNN match kernel is particularly relevant for such tasks as it provides us with good invariance properties with respect to rigid 3D transforms. We also observe slightly better results for the *OxfordFlower102* dataset. As explained in section 5.1.2 when commenting results of Table 5.2, this is probably due to the fact that the ImageNet dataset highly intersects with the *Ox-*

fordFlower102 dataset in terms of semantic contents. As a consequence, the CNN model already learned highly informative patterns to discriminate between flower species. Using such informative patterns to build our SNN representation and learning from such *small* datasets allows us to exhibit better classification results over CNN methods that needs a lot of training samples to provide good generalization properties. This suggests that our approach can be a good alternative to CNN when considering learning from real world visual domain knowledge where some classes may be poorly populated or strongly imbalanced.

Method	*FlickrsLogos32*	*Vehicles29*	*Paris*	*Aircraft*	*CUB-Birds200*	*OxFlowers102*
GoogLeNet [SIV16]	87.5	**79.9**	71.3	**88.1**	**72.4**	89.5
GoogLeNet (no FT)	67.7	59.3	55.3	72.7	24.4	59.5
SNN	**92.5**	75.5	**76.5**	80.2	48.5	**90.2**

TABLE 5.7 – Classification performance of the SNN representation compared to state-of-the-art Convolutional Neural Networks on several datasets. Results are expressed in terms of top-1 classification accuracy. The results obtained from the *GoogLeNet* experiments have been respectively obtained with and without fine-tuning (FT).

5.4 Temporal Extension of the SNN representation

In this section, we introduce a temporal extension to the previously introduced SNN match kernel so as to address the problem of fine-grained audio classification. Similarly, to the image SNN representation, this extension is based on shared nearest neighbors match kernel of the low level audio features extracted at the frame level. Again, to make such strategy scalable to the tens of millions of MFCC features extracted from the tens of thousands audio recordings of the training set, we used high-dimensional hashing techniques coupled with an efficient approximate nearest neighbors search algorithm with controlled quality. Then, as explained in the previous chapter, we use a sliding window for the temporal pooling of the raw matches so as to boost locality and interpretability of the detected patterns. The main difference between this audio extension and the original representation is the fact that we perform some weighting on each low level feature according to the semantic coherence of its nearest neighbors which showed significant gains in terms of audio classification performance. Results show the effectiveness of the proposed technique which ranked 2nd among the 7 research groups participating to the LifeCLEF 2015 audio-based bird species identification task [2].

Section 5.4.1 describes the preliminary audio processing and features extraction steps. Section 5.4.2 then presents this audio representation to be further classified

2. http://www.imageclef.org/lifeclef/2015/bird

thanks to a linear supervised classifier (section 5.4.4). Section 5.4.5 and 5.4.6 finally reports and discuss the results we obtained within the LifeCLEF 2015 challenge.

5.4.1 Pre-processing and features extraction

The dataset used for this challenge is composed of 33,203 audio recordings belonging to 999 bird species from Brazil. As various recording devices are used, and because it is difficult to capture these sounds as birds are often far away from the recording devices, many recordings contain a lot of noise. To overcome this problem, we used SoX, the "Swiss Army knife of sound processing programs" [3]. As a first step, we used the *noisered* specialised filter, to filter out noise from the audio, and then we reduce the length of large (i.e. $> 0.1s$) silent passages from audio files to $0.1s$. In order to obtain audio files with ideally no more noise but still enough signal, we tried removing as much noise as possible (using the noisered amount parameter) while guaranteeing that the resulting audio file was at least 20% the size of the initial audio record. After this pre-processing step, we used an open source software framework, marsyas [4], to extract MFCC features with parameters based on the provided audio features in the BirdCLEF task: MFCC were computed on windows of 11.6 ms, each 3.9 ms, and we additionally derived their speed resulting in 26-dimensional feature vectors (13+13) for each frame.

5.4.2 Temporal Shared Nearest Neighbors Match Kernel

We consider recordings represented by sets of 26-dimensional MFCC features and the temporal extension of the pooling algorithm described in section 4.2.2 now consists in aggregating the raw matches within a sliding window (centered around each frame) and then keeping the max score over the whole record. More formally, we can reformulate our explicit formulation of Equation 4.4 as:

$$\Phi_S^w(X) = \sum_{m=1}^{M} \left(\max_{t_i \in [1,T_m]} \sum_{t=t_i-(w/2)}^{t_i+(w/2)} \sum_{\mathbf{x} \in X} \varphi_{\mathbf{x}}(\mathbf{z_t^m}) \right) . \mathbf{e_m} \tag{5.3}$$

where M is the number of audio recordings in the training set, T_m the number of frames of the m-th recording and $\mathbf{z}_t^m$ the MFCC feature of the t-th frame of the m-th recording. The size w of the sliding window was trained by cross-validation and then fixed to $w = 1000$ frames (resulting in a sliding window of 3.9 seconds).

3. http ://sox.sourceforge.net/
4. http ://marsyas.info/

5.4.3 Weak semantic weighting

As we are in the case of weakly annotated audio recordings with multiple classes (primary and secondary species) and highly cluttered contexts, we suggest improving our SNN match kernel by weighting the query features according to the semantic coherence of their k nearest neighbours. We therefore compute a discrimination score $f(\mathbf{x})$ for all MFCC features $\mathbf{x} \in X$ of a given audio recording I_X. A weak label $l(\mathbf{x})$ is first estimated for each $\mathbf{x}$ as the most represented label within the k-nearest neighbors of $\mathbf{x}$ in the training set (actually the ones computed by the hash-based k-nn search method described in section 4.2.1). The semantic weight $f(\mathbf{x})$ is then computed as the percentage of the k-nearest neighbors having the same weak label than the feature itself (i.e. the percentage of k-nearest neighbors whose label is equal to $l(\mathbf{x})$). Finally, our representation of a given audio recording I_X becomes:

$$\Phi_S^{w\prime}(X) = \sum_{m=1}^{M} \left(\max_{t_i \in [1,T_m]} \sum_{t=t_i-(w/2)}^{t_i+(w/2)} \sum_{\mathbf{x}\in X} f(x).\varphi_{\mathbf{x}}(\mathbf{z_t^m}) \right) .\mathbf{e_m} \tag{5.4}$$

5.4.4 Training and classification

To achieve an effective supervised classification task, we trained a linear discriminant model on top of our proposed SNN matching-based representations (cf. Equation 5.3). This requires first building the representations of all audio recordings in the training set and then in learning as many linear classifiers as the number of species in the training set. The resulting linear classifiers are of the form:

$$h(\Phi_S^{w'}(X)) = \mathbf{W}^T.\Phi_S^{w'}(X) + b \tag{5.5}$$

This way they interestingly affect weights ω_j to each audio recording in the training set according to its relevance for the targeted class (rather than affecting weights to the individual MFCC features as in the raw representation of Equation 4.4). In our experiments, we used a linear support vector machine for training these discriminant linear models. We more precisely used the LibLinear implementation of the scikit-learn library with a squared hinge loss function and a L_2 penalty. The C parameter of the SVM was fixed to $C = 100.0 * weight(class)$ where $weight(class)$ is a class-dependent weight that is automatically adjusted to be inversely proportional to the class frequency. Finally, the scores returned by the SVM are converted into probabilities using the following p-value test:

$$P(\text{class}) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{1}{\sqrt{2}}\frac{s(class) - \mu(\text{class})}{\sigma(\text{class})}\right)\right) \tag{5.6}$$

where erf is the Gauss error function and $\mu(class)$ and $\sigma(class)$ are respectively the mean and the standard deviation of the SVM score across the considered class. We will see in the experiments that this conversion provides a noticeable accuracy improvement.

5.4.5 Experiments and results

Dataset and task

The LifeCLEF 2015 bird dataset [JGG+15] is built from the Xeno-canto collaborative database [5] involving at the time of writing more than 240k audio records covering 9,350 bird species observed all around the world thanks to the active work of more than 2,510 contributors. The dataset contains minimally 14 recordings per species and minimally 10 different recordists per species. Audio records are associated to various metadata such as the type of sound (call, song, alarm, flight, etc.), the date and locality of the observations (from which rich statistics on species distribution can be derived), some textual comments of the authors, multilingual common names and collaborative quality ratings (more details can be found in [JGG+15]). The task was evaluated as a bird species retrieval task. A part of the collection was delivered as a training set available a couple of months before the remaining data is delivered. The goal was to retrieve the singing species among the top-k returned for each of the undetermined observation of the test set. Participants were allowed to use any of the provided metadata complementary to the audio content but we did not in our own submissions.

Run Name	MAP 2(without Background Species)	MAP 2 (with Background Species)
MNB TSA Run 4	0.454	0.414
MNB TSA Run 3	0.442	0.411
MNB TSA Run 2	0.442	0.405
MNB TSA Run 1	0.424	0.388
INRIA ZENITH Run 2	**0.334**	**0.291**
QMUL Run 1	0.302	0.262
INRIA ZENITH Run 3	**0.292**	**0.259**
INRIA ZENITH Run 1	**0.265**	**0.240**
GOLEM Run 2	0.171	0.149
GOLEM Run 1	0.161	0.139
CHIN. AC. SC. Run 1	0.01	0.009
CHIN. AC. SC. Run 3	0.009	0.01
CHIN. AC. SC. Run 2	0.007	0.008
MARF Run 1	0.006	0.005
MARF Run 2	0.003	0.002
MARF Run 3	0.005	0.005
MARF Run 4	0.000	0.000

TABLE 5.8 – Official results of LifeCLEF 2015 Bird Task - Our runs are referred as **INRIA Zenith Run 1**, **INRIA Zenith Run 2** and **INRIA Zenith Run 3**.

5. http://www.xeno-canto.org/

Submitted runs and results

We submitted three *runs* to be evaluated within the LifeCLEF 2015 challenge:

INRIA Zenith Run 1: This run was not based on the method described in this chapter, but on a direct instance-based classification method we evaluated beforehand [JCB14]. This allows us to measure improvements provided by the SNN representation-based learning. It basically relied on a very similar matching process than the one described in 4.2.1 but it did not train any supervised classifier on top of the resulting matching score. It actually only computed the top-30 most similar training records of each query and then used a simple vote on the labels of the retrieve records as classifier. It however included a pre-filtering of the training set that removed the less discriminant MFCC features from the training set.

INRIA Zenith Run 2: The approach described in this section.

INRIA Zenith Run 3: The same approach than Run 2, but without the conversion of the SVM scores into probabilities (see section 5.4.4).

The results of the whole challenge, including our own results as well as the results of the other participating research groups, are reported in Figure 5.7 and Table 5.8.

5.4.6 Discussion and perspectives

Our system globally achieved very good performance and ranked as the second best one among the 7 participating research groups. Our best run, i.e. the one based on the method proposed in this section, achieved a mAP of $0,334$ when considering only the primary species of each test recording. This is 3 points better than the state-of-the-art approach of the QMUL research group which makes use of unsupervised feature learning as described in [SP14] whereas we used classical MFCC features. Also, compared to the mAP of our first run (equals to 0.265), it shows that training discriminant models using our SNN match kernel is much more effective than using the former instance-based classification approach. The weights learned by the SVM on the pooled matches actually compensate most of the bias involved by the heterogeneity of the noise level in the recordings and the heterogeneity of the recordings length. The intermediate performance of INRIA Zenith Run 3 shows, however, that the conversion of the SVM scores into probabilities plays an important role in the performance of Run2. Our interpretation of this phenomenon is related to the fact that the number of training records per species follows an heavily tailed distribution (as in most biodiversity data). The SVM scores are consequently boosted for the most populated species to the detriment of the less populated ones. Our p-value normalization allows compensating this bias by normalizing the distribution across all classes.

Now, the performance of our approach is still much lower than the best performing system of MNB TSA which has a mAP equal to 0.453. Note that their approach is in essence not so far from ours as they also represent the audio recordings thanks to their matching score in a reference set of audio segments. A major difference however is that they pre-compute a clean set of relevant audio segments whereas we use all the recordings of the training set as vocabulary. They notably consider only the audio recordings with the highest user ratings in the metadata, and, then extract only the segments that are likely to contain a bird song (thanks to bandwidth considerations). A second difference is that their matching is computed at the signal level whereas we are using MFCC features that might loose some important information. We believe that integrating these two additional paradigms within our framework could make it competitive with their approach. Investigating more in depth the semantic pruning strategy that we introduced in [JCB14] but in the context of our new SNN match kernel might for instance be an effective way of further improving the quality of the reference set.

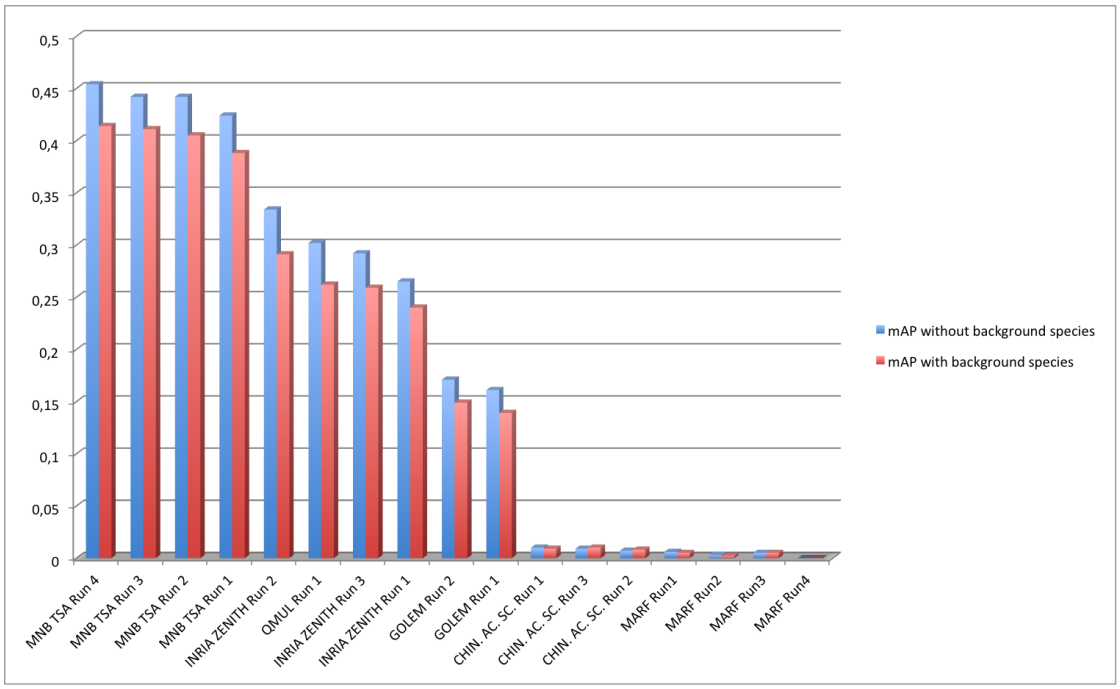


FIGURE 5.7 – Official results of LifeCLEF 2015 Bird Task - Our runs are referred as **INRIA Zenith Run 1**, **INRIA Zenith Run 2** and **INRIA Zenith Run 3**.

5.5 Conclusion

In this chapter, we show that the SNN representation demonstrates very competitive performance on fine-grained classification tasks such as the *FGVC-Aircraft* dataset, where we demonstrate the benefit of integrating geometric information in the global image representation. We observed that CNN, as well as the competitive Fisher Vector representation (when used with handcrafted color features), demonstrate better performance on visual classification tasks where other attributes such as color or higher level visual information, are useful. Thus, we also consider the use of other local feature schemes, such as handcrafted color descriptors and off-the-shelf CNN features. We show that it provides significant improvement over the SIFT-based SNN representations on the *CUB-Birds-200* and the *OxfordFlower102* datasets. Although state-of-art-CNN architectures gives better results than our approach on several tasks, we demonstrate that the SNN representation exhibits better classification performance than CNN when considering small rigid objects (such as logos) or when the amount of labelled training data is small. This is due to the explicit use of the localized geometric information in our representation scheme and the to poor generalization ability of CNN when learning with small training datasets. Finally, we also introduce a temporal extension of this Shared Nearest Neighbor representation. We show that it provides significant results on a bird song identification task.

In the next chapter, we address the memory and computation time issues of the SNN representation that we discussed earlier in section 4.5.1. We propose a supervised dimensionality reduction approach on top the SNN representations. We show how this allows us to highly compress the size of the initial vocabulary while keeping very satisfying classification performance on well interpretable patterns.

Chapitre 6

Spatially Consistent Visual Dictionary Learning through Supervised Feature Elimination

Contents

As we have seen in the previous chapters, an interesting property of Shared Nearest Neighbor embedding is that it explicitly maps the visual content of a given image onto a very large set of visual patches. So that the image can be represented through a very high-dimensional feature vector encoding the explanatory power of each visual patch in the training set. In this chapter, we propose an approach to drastically reduce the dimensionality of such brute-force and over-complete representation thanks to a recursive feature elimination method. We show that the number of *spatial atoms* of the representation can be reduced by up to two orders of magnitude without much degrading the encoded information. The method that we propose in this chapter can actually be seen as a supervised method for learning

a compact vocabulary of *discriminant* and *spatially localized* visual patches to be used as atoms of an interpretable image representation.

6.1 Definition and Overview

We define a *spatially localized vocabulary* as a set $\mathcal{Z}$ of *spatial atoms* Z_j, $j \in [1, N]$, each uniquely corresponding to a spatial region R_j of an image in the training set. We define each *spatial atom* Z_j as being itself composed of a set of spatially localized d-dimensional feature vectors $\mathbf{z_j^i}$, $i \in [1, |Z_j|]$, extracted from R_j and representing its local visual content. Our aim is to automatically learn a *spatially localized vocabulary* $\mathcal{Z}$ that is as much compact as possible while still containing the most explanatory visual patterns of the labeled classes in the training set. We therefore introduce a new Recursive Visual Patch Selection algorithm (RVPS) that is summarized in **Algorithm 1**. Its principle is to progressively compress the *spatially localized vocabulary* $\mathcal{Z}$ by recursively eliminating the less discriminant atoms. Each recursion includes 3 main steps: (i) the computation of the SNN representations of the images in the training set $\mathcal{X}$ (based on the current *spatially localized vocabulary* $\mathcal{Z}^{(t)}$), (ii) the learning of a multi-class support vector machines on top of the computed SNN representations and (iii), the elimination of the less discriminant spatial atoms $Z_j^{(t)}$ in $\mathcal{Z}^{(t)}$. These 3 steps are repeated T times. The main parameter of the algorithm is the filtering ratio s that fixes the percentage of non-eliminated atoms at each iteration (*e.g.* $s = 0.9$ means that 90% of the atoms are kept within the *SpatialAtomsFiltering* function). The initialization of the algorithm as well as the description of the different steps of each recursion are illustrated in Figure 6.1 and will be detailed in the following subsections.

Algorithm 1: *RecursivePatchSelection.*

input : Vocabulary $\mathcal{Z}$, filtering ratio s, training set X, image labels $\mathcal{Y}$, Number of iterations T

output: Filtered Vocabulary $\mathcal{Z}^{(T)}$

if $(T > 1)$
  $\mathcal{Z}^{(T-1)}$=*RecursivePatchSelection*($\mathcal{Z}$,s,X, $\mathcal{Y}$, $T-1$);
else
  $\mathcal{Z}^{(T-1)}$=$\mathcal{Z}$;
Φ=*ComputeSNN*(X,$\mathcal{Z}^{(T-1)}$);
$\mathbf{W}$=*LearnSVM*(Φ,$\mathcal{Y}$);
$\mathcal{Z}^{(T)}$=*SpatialAtomsFiltering*($\mathcal{Z}^{(T-1)}$,$\mathbf{W}$,s);
return $\mathcal{Z}^{(T)}$

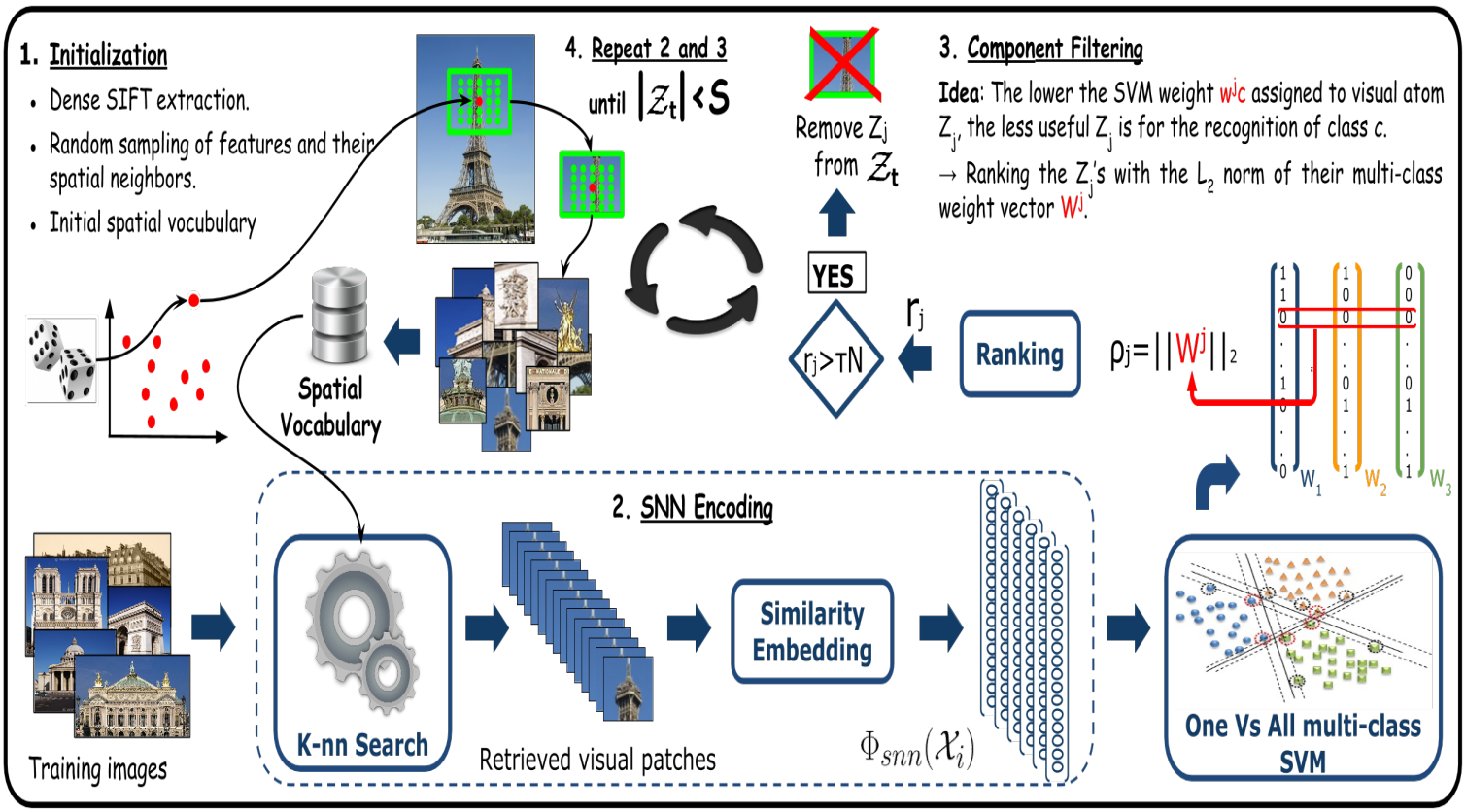


FIGURE 6.1 – Overview of the spatially localized visual dictionary learning algorithm.

6.2 Supervised Spatially Localized Visual Dictionary Learning

6.2.1 Initialization

The initial vocabulary $\mathcal{Z}^{(0)}$ to be used as input of the *RecursivePatchSelection* algorithm is created by randomly picking N_0 spatial atoms within the images of the training set $\mathcal{X}$. When N_0 is very large (*e.g.* 1 million of potentially overlapping regions), this allows starting the vocabulary learning with an over-complete representation to be progressively reduced afterwards. More practically, we uniformly draw N_0 local features $\mathbf{z_j}$ from the raw set of all spatially localized local features extracted from the images (be they *hand crafted* such as SIFT features or *off-the-shelf* low level features learned through a convolutional neural network). The j-th spatial atom Z_j is then formed by $\mathbf{z_j}$ itself and by the set of its top-m spatially neighboring local features $\mathbf{z_j^i}$, $i \in [1, m]$.

6.2.2 SNN representation

The goal of this step is to compute intermediate representations of the images $\mathcal{X} \in X$ based on a *spatially localized vocabulary* $\mathcal{Z}$. Each image in $\mathcal{X}$ is supposed to be described by a set of d-dimensional spatially localized local features $\mathcal{X} = \{\mathbf{x}\}$. To map these local features onto the N spatial atoms Z_j of the vocabulary $\mathcal{Z}$, we use the Shared Nearest Neighbor representation introduced in chapter 4 from which we can derive the following explicit embedding:

$$\mathbf{\Phi}(\mathcal{X}) = \sum_{\mathbf{j}=1}^{N} \Phi_j(\mathcal{X}).\vec{e_j} = \sum_{\mathbf{j}=1}^{N} \frac{1}{|\mathcal{X}|} \sum_{\mathbf{z} \in Z_j} \sum_{\mathbf{x} \in \mathcal{X}} \varphi(r_{\mathbf{x}}(\mathbf{z})).\vec{e_j} \tag{6.1}$$

where $r_{\mathbf{x}}(\mathbf{z}) : R^d \rightarrow N^+$ is the same approximate ranking function than the one depicted in section 4.2.2 as well as the rank-based activation function $\varphi(r)$. Intuitively, each component $\Phi_j(\mathcal{X})$ of the high-dimensional representation $\mathbf{\Phi}(\mathcal{X})$ quantifies how likely it is that the image $\mathcal{X}$ contains the same visual pattern than the one depicted by the spatial atom Z_j. In our experiments, we used the geometrically consistent SNN kernel variant (see section 4.2.2) for the datasets that involve rigid objects, *i.e.* buildings and logos.

6.2.3 Multi Class Feature Selection scheme

To select the most discriminant atoms for a given classification task, we adopt a SVM-based multi class feature selection strategy first proposed in Guyon et al. [GWBV02] and Chapelle et al. [CK08]. It consists in defining a filtering criterion ρ_j for the j-th component Φ_j of the representation space by analyzing the weights w_{jk} with $k \in [1, C]$ across the C one-versus-all L_2 regularized Support Vector Machine (SVM [Vap99]) classifiers learnt on the task. As explained in [CK08], a very simple and theoretically elegant filtering criterion is the l_2 norm of the vector $\mathbf{w_j} = \sum_{k=0}^{C} w_{jk}\vec{e_i}$. This can be simply shown by introducing in L_2 regularized objective function scale factors $\sigma_j \in [0, 1]$ to each component weight w_{jk} for $k \in [1, C]$ subject to $\sum \sigma_j = m$ as we want to keep a restricted number of components, say m. By introducing these changes and modeling the constraints on the σ_j with Lagragian multipliers we get:

$$L'(w, \sigma) = \frac{1}{2} \sum_{k=1}^{c} (\sum_{j=1}^{d} w_{jk}^2 + \lambda \sum \sigma_j + \frac{C}{2} \sum_{i=1}^{n} l(y_{ik} \sum_{j=1}^{d} \sqrt{\sigma_j} w_{jk} x_{ji})) \tag{6.2}$$

Applying the change of variable $w'_{jk} \leftarrow \sqrt{\sigma_j} w_{jk}$ and relaxing the constraints to $\sigma_j \geq 0$ one can obtain:

$$L'(w', \sigma) = \frac{1}{2} \sum_{k=1}^{c} (\sum_{j=1}^{d} \frac{w'^2_{jk}}{\sigma_j} + \lambda \sum \sigma_j + \frac{C}{2} \sum_{i=1}^{n} l(y_{ik} \sum_{j=1}^{d} w'_{jk} x_{ji})) \tag{6.3}$$

Then if we note $V(\sigma) = \min_w(L'(w', \sigma))$ and we evaluates its first order local approximation at $\sigma_j = 1$, this gives the tangent corresponding to direction to go to minimize $V(\sigma)$ with respect to σ_j. Intuitively, the more the value of the gradient is positive, the more going toward $\sigma_j = 0$ optimizes the objective function. A natural valued criterion for removing the component j can then be this gradient such that the more optimal component to remove is given by:

$$\tilde{j} = argmax_j \frac{\partial V(\sigma)}{\partial \sigma_j}|_{\sigma_j=1} \tag{6.4}$$

with:

$$\frac{\partial V(\sigma)}{\partial \sigma_j} = \frac{\partial L'(w,\sigma)}{\partial w}\frac{\partial w}{\partial \sigma_j} + \frac{\partial L'(w,\sigma)}{\partial \sigma_j} = 0 + \frac{1}{2}(-\sum_{k=1}^{c}\frac{w_k^2}{\sigma_j^2} + \lambda) \tag{6.5}$$

and thus:

$$\tilde{j} = argmin_j \sum_{k=1}^{c} w_{jk}^2 = argmin_j |\mathbf{w_j}|_2^2 \tag{6.6}$$

The filtering score of an atom Z_j can, then, be computed as $\rho_j = |\mathbf{w_j}|_2^2$ and the filtering consists in ranking all the components thanks to ρ_j and keep only the top sN atoms (where N is the total number of atoms in $\mathcal{Z}$ and s the filtering ratio). Note that when an atom Z_j is pruned, all the local features $\mathbf{z_j^i}$ belonging to it are definitely removed from the vocabulary.

6.2.4 Discussion

We highlight the fact that our Recursive Visual Patch Selection algorithm (RVPS) is actually different from a classical Recursive Feature Elimination (RFE) [CK08]. The RFE method actually relies on a fixed representation space and attempt to find the optimal subspace by eliminating the less informative components. On the contrary, the representation space induced by our manifold learning method is evolving at each iteration. Not only some components atoms are removed from the vocabulary but the contribution of the remaining ones do evolve as well. This is mainly due to the rank-based activation function of the SNN embedding. When removing some atoms, the rank $r_{\mathbf{x}}(\mathbf{z})$ of the kept features can only decrease and, as a consequence, the contribution $\Phi_j(\mathcal{X})$ of the remaining atoms can only increase. So that the selected atoms do progressively increase their contribution to the representation of more and more pictures. In other words, we do progressively improve the encoding of the manifold structure of the data thanks to the selection of more and more contributive data items. If we did not recompute the SNN representations after each atom elimination step, we would select some discriminant atoms but we would not select the most contributive ones.

6.3 Experiments

In this section, we will present the experimental protocol that has been used to show the contribution of our Recursive Visual Patch Selection algorithm as a supervised spatially consistent visual dictionary learning scheme. We will first introduce the different datasets and settings that have been used in the different experiments. We will perform a series of experiments to analyze the performance of our approach. We will first analyze the impact of the recursive filtering on the classification performance and will show that we can highly compress the SNN exhaustive visual vocabulary without much degrading the classification accuracy. We will then evaluate our approach on several fine-grained visual classification tasks and compare the classification performance with state-of-the-art ConvNet architectures with and without the use of external training data and fine-tuning procedure. Finally, we will qualitatively demonstrate the ability of our method to learn highly discriminant and interpretable visual models.

6.3.1 Experimental Settings

Datasets To evaluate our method, we used the **FlickrLogos32** [RPLvZ11], **Paris Buildings** [PCI+08] and **Oxford Flower** [NZ08] datasets.

Data Augmentation As described in details in [GMJP14], we performed some data augmentation by resizing the images so that their resolution is not higher than 300k pixels and we then mirror the training images.

Local Features For the *FlickrLogos32* and *ParisBuildings* datasets, SIFT features were extracted around Harris-Hessian-Laplace interest points. For the Oxford-Flower dataset, we rather used off-the-shelf CNN-based features learnt with the GoogLeNet CNN architecture pretrained on the ImageNet dataset [KSH12]. Images were forwarded to the inception_3a layer output leading to 784 densely sampled 256-dimensional spatially localized features for each image. All descriptors were L_2-normalized to the unit ball and square rooted.

SNN Settings The number of neighbors m returned by the approximate knn search method is fixed to 100, the quality control parameter of the approximate search is fixed to $\alpha = 0.4$. The PCA-like hashing scheme to index the local features is chosen and length b of the hash codes is fixed to the dimension of the initial feature space (128 for SIFT and 256 for CNN-based features). The length t of the hash code prefix used to construct the hash table is set to $t = log_2 N + 2$. We used the spatially consistent variant of the SNN embedding only for the two datasets involving rigid objects, *i.e. FlickrLogos32* and *ParisBuildings* and the non-geometric SNN kernel $\Phi_S(.)$ was used for the flower dataset. The number N_0 of random spatial atoms in

the initial vocabulary $\mathcal{Z}^{(0)}$ was fixed to $N_0 = 2^{20}$ the spatial neighborhoods size of each atom was fixed to $m = 256$ local features.

6.3.2 Recursive filtering impact

To study the impact of the filtering ratio s of our RVPS algorithm, we ran it with four different values, *i.e.* $s = 0.5$, $s = 0.3$, $s = 0.1$ and $s = 0.01$. The recursively computed image representations were then used as input of a L_1 regularized logistic regression (with default regularization constant $C = 1$). Figure 6.2 displays the resulting classification accuracy on the *FlickrLogos32* dataset as a function of the number of spatial atoms in the learnt vocabulary. It shows that if the filtering ratio is too strong (*e.g.* $s = 0.1$ or $s = 0.01$), the classification performance quickly degrades. On the other side, with a reasonable filtering ratio of $s = 0.5$ or $s = 0.3$, the classification performance remains rather stable with up to two orders of magnitude less atoms in the vocabulary. When the vocabulary contains only 256 spatial atoms, the accuracy is still very acceptable meaning that they are highly informative for the classification task.

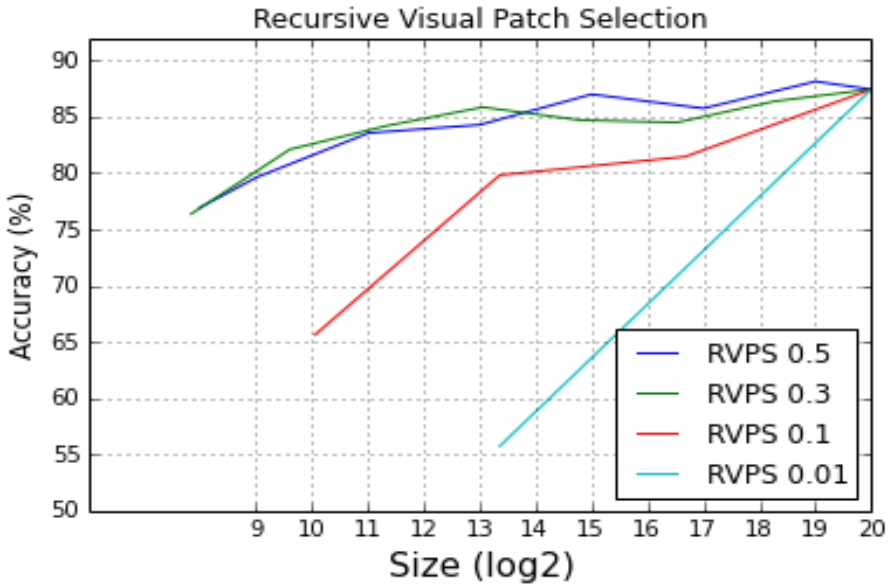


FIGURE 6.2 – RVPS accuracy vs. number of spatial atoms.

6.3.3 RVPS vs. RFE

To further study the effectiveness of our Recursive Visual Patch Selection algorithm (RVPS), we compared it with a classical Recursive Feature Elimination (RFE) computed on top of our initial SNN representations (*i.e.* the ones based on the initial vocabulary $\mathcal{Z}^{(l)}$). The results are provided in Table 6.1. They show that at constant dimensionality, the representations learned by RVPS are much more effective for the classification task than the ones learned by RFE. This is not sufficient to conclude that the selected spatial atoms are better in terms of interpretability (the generative aspect is probably even more important). But this proves that they do provide a better generalization ability which is already an interesting criterion.

# atoms	**256**	**1,024**	**4,096**	**16,384**	**65,536**
RFE	12.19	15.94	39.2	75.6	81.32
RVPS	78.4	83.6	84.4	86.9	86.15

TABLE 6.1 – RVPS vs. RFE classification accuracy.

6.3.4 Classification performance

In this section, we compare the classification accuracy obtained from the RVPS representations to the one of the GoogLeNet convolutional neural network [SIV16]. For the GoogLeNet runs, we considered both pre-training on ImageNet (and fine-tuning on the targeted task) and without any pre-training phase (*i.e.* without fine-tuning on the targeted task). We respectively used a learning rate of 0.001 with local multipliers of 10 on the last fully connected layers and a learning rate of 0.01 without adding additional local multiplier. For both fine-tuning and no fine-tuning, we used a weight decay parameter of 0.0005 and a momentum of 0.9. The results are provided in Table 6.2 for the three datasets. They show that the RVPS-based representations are quite competitive with the CNN ones with slightly lower performance than the network fine-tuned on ImageNet but much better performance than the one trained on the same data than our RVPS method. Now, the main advantage of RVPS is to allow interpreting very easily which visual patterns of the training set were learnt. Indeed, each spatial atom of the spatially localized visual vocabulary has a uniquely defined visual representation.

6.3.5 Interpretability of the learnt vocabulary

In this section, we qualitatively demonstrate the ability of our method to identify which image parts were mostly used by the learning algorithm to model the different visual concepts of a specific domain knowledge. To identify the most class-specific spatial atoms that allow the learning algorithm to identify a given object, we simply take the vocabulary Z^t obtained after several iterations of our filtering step and

Method	*FlickrLogos*	*Paris*	*Flower*
GoogLeNet FT	87.5	70.5	89.56
GoogLeNet no FT	66.8	54.4	61.7
RVPS - 16,384 atoms	86.9	73.2	86.43
RVPS - 4,096 atoms	84.4	70.9	86.36
RVPS - 1,024 atoms	83.6	67.6	84.31

TABLE 6.2 – RVPS vs. CNN classification Accuracy.

Notre Dame | Guiness | Google | Arc de Triomphe | Daffodil | Hard-leaved pocket orchid

FIGURE 6.3 – Learned Spatial Atoms for 6 classes.

recompute on top of it C One-vs-All L_2 regularized SVM, one for each class. For each SVM model $\mathbf{w_k}$, we rank every component j with the class-wise weight w_{jk}. Figure 6.3 shows bounding boxes of top-3 visual atoms from different visual concepts in their original images. We see that for some domain such as logos, the chosen patches often correspond to variants of the same visual pattern or to the same visual pattern but under different view conditions and different contexts. This is notably the case of the *Google* logos in Figure 6.3 where discriminant spatial atoms well correspond to image regions where the logo appear in different contexts (here newspaper and commercial adds). For more complex visual entities involving richer visual diversity such as buildings or plants, the chosen patches rather correspond to different parts of the whole entity such as the *Notre Dame* cathedral whose discriminant spatial atoms have been as well identified as outside part than inside part. This experiments let us suggest that this kind of spatially consistent vocabulary learning algorithm might be very helpful for domain experts to identify very discriminant patterns for different classes of objects.

6.4 Discussion and Conclusion

The work of this chapter can be seen as a first step toward the very interesting and promising challenge of transferring the knowledge from learning algorithms to humans so that we can gain insights into which part of the data is used by the learning algorithm to discriminate between different visual concepts. This kind of algorithms would allow humans such as domain experts to (i) understand which visual patterns are discriminant or ambiguous from a concept to another, (ii) detect some errors or limitations in the machine learning process, and (iii) improve their knowledge of the objects of interest by taking advantage of the machine to detect fine relevant details that could not be discovered otherwise. Most visual representations used in classical visual models are too abstract to fulfill these interpretability objectives. The learned *atoms* (*e.g.* latent variables in probabilistic models or *visual words* in codebook learning methods) do actually not have a uniquely defined and easily interpretable visual appearance. So that it makes it difficult to understand precisely which parts of the training data has been used by the system to learn the task. This is mainly due to the fact that most supervised or unsupervised models are based on either local generalization (BoW, Fisher Vectors, etc.) or discriminative abstracting models (such as Convolutional Neural Networks). Thus, the resulting visual representations do not rely on directly *visualizable* atoms, *i.e.* atoms that can be mapped without any ambiguity onto localized visual contents in the training set.

The challenge that we addressed in this work was thus to devise a new image representation learning algorithm that overcomes the lack of interpretability of state-of-the-art models. We did show that the proposed Recursive Visual Patch Selection algorithm allows to provide competitive image classification performance with the state-of-the-art while enabling to learn highly discriminant and interpretable visual models that maps the visual content of the images onto a vocabulary of uniquely defined and spatially localized visual atoms.

Chapitre 7

Conclusion

Contents

In this chapter, we review the different contributions we proposed in this work and discuss about their main limitations. Then, we propose some promising research perspectives to investigate with respect to the results presented in this thesis.

7.1 Contributions

In this work, we focus on the issue of *fine-grained classification* which is a particular classification task where classes may be visually distinguishable only from subtle localized details. Although state-of-the-art aggregation-based methods (such as Fisher Vectors or VLAD) are based on local descriptors, the lack of spatially localized information implies that a lot of details are lost in the final image representation. However, it would be of great interest to keep this information especially in the context of fine-grained classification. This also implies that these representations are not able to recognize small objects that can appear in possibly highly cluttered backgrounds. This work is motivated by the need of devising fine image representations to address such fine-grained classification challenges by encoding enough localized discriminant information. The main research line we investigate relies on spatially localized similarities between images computed thanks to efficient approximate nearest neighbor search technique and a localized parametric geometry.

The first contribution we propose is a spatially consistent matching-based K-nn

classifier based on the pooling of localized similarity scores of the training images with respect to the image to be represented. The images are first described with local features. For each local descriptor, we search its K nearest neighbors among the descriptors extracted from the training images thanks to an approximate nearest neighbor search technique. Then, we integrate finely localized strong geometry to refine the similarity scores of the retrieved patterns and we aggregate these scores for each class. This allows us to drastically reduce the number of false alarms in the retrieved patterns and to focus the attention on regions of interest. We finally vote for the class with the highest score. We demonstrate that our approach exhibits promising classification performance on datasets involving rigid and planar objects such as images of logos (FlickersLogos, Belgalogos) and also on car models (Vehicles29). The main drawback of this approach is that the score of each class is an aggregation of a lot of task-irrelevant matching scores that are only based on geometrically consistent similarities. As a consequence, the classifier gives to each retrieved localized pattern as much importance as for the other ones whereas only some of them correspond to discriminant features. This suggests that better performance should be obtained by taking into account the relative discriminating importance of the different local patterns of the visual dictionary (*i.e.* the training images in our case).

To this aim, we propose a second contribution where we introduce a new match kernel based on shared-nearest neighbors and localized geometric information. We derive an image representation corresponding to the explicit embedding of this newly introduced match kernel. We propose a first version of the SNN representation where each image is densely described by a set of local handcrafted descriptors. Each image descriptor is encoded by a vector containing its similarity with the descriptors of the training database. We use an efficient indexing and approximate k-nearest neighbors search technique in order to scale up the computation of these similarity scores. Each image is then represented through a very high-dimensional feature vector encoding its similarity to each local descriptor in the training set. As the number of extracted local features from common training datasets is often very large (about hundreds of millions or billions), learning from these representations is likely to cause serious overfitting. To avoid overfitting and to reduce the processing costs, the dimensionality of the resulting over-complete representation is further reduced by aggregating into a single component the raw matches belonging to the same region (patch) of the training images. Such image patches are obtained by regularly subdividing the training images to a particular resolution. In order to allow the system to learn spatially consistent visual patterns, we perform local geometry consistency analysis in a RANSAC fashion to refine the similarity scores of these sub-regions. The resulting geometrically consistent SNN embedding explicitly maps the visual content of a given image onto a large set of visual patterns. These patterns are, somehow, comparable to a visual vocabulary where each word corresponds to a particular training image's region. Hence, although it shares common properties with other match kernels or vocabulary-based and aggregation-based methods (such as BoVW, Fisher

Vectors or VLAD), the main originality of our approach is to embed the spatial arrangement of the fine-grained visual patterns into a high-dimensional global image representation. Moreover, as we apply spatial pooling of local matches in training images, our representation scheme provides much more spatial invariance than the popular SPM method where the spatial pooling is rather performed on the image to be represented. On the opposite, each component of the SNN representation corresponds to a uniquely defined spatially localized visual word. Hence, different images showing two objects at different locations will still *activate* the same components of the representation. We show that our approach provides high performance gains compared to our previous baseline of Chapter 3, especially for the Vehicle29 dataset. This validates our hypothesis concerning the need to learn to combine finely localized geometrically consistent patterns.

To evaluate the performance of the SNN representation, we compare it with state-of-the-art approaches on several fine-grained classification tasks. We notably evaluate it on the *FGVC-Aircraft* dataset and show that integrating geometric information in such global image representation allows us to obtain fine-grained classification performance outperforming state-of-the-art methods. However, we observe that our representation obtains lower performance on other fine-grained tasks where discriminant attributes are based on color or higher-level visual information rather than appearance-based descriptors (such as SIFT). Thus, we also investigate the behavior of our method by using several underlying feature schemes to improve the genericity of the SNN representation. We notably consider using handcrafted color descriptors and off-the-shelf CNN features. We show that using CNN features provides significant improvements over the SIFT-based SNN representations on the *CUB-Birds-200* and the *OxfordFlower102* datasets. We also demonstrate the relevance of such representation on fine-grained audio classification tasks with a temporal extension of the original SNN representation by integrating audio features in our pipeline. Our method exhibits competitive results on fine-grained audio classification tasks such as the *LifeCLEF* 2015 bird species identification challenge [JGG+15]. Although we show that the potential of our approach is subject to strong competition with state-of-the-art Convolutional Neural Network architectures, we observe that our method exhibits better classification performance than CNN when considering small rigid objects (such as logos) or when the amount of labelled training data is small. The poor generalization ability of Convolutional Neural Networks when learning with small training datasets suggests that our approach can be a good alternative when learning from specific domain visual knowledge involving poorly populated and strongly imbalanced classes.

The Shared Nearest Neighbor embedding explicitly maps the visual content of an image onto a very high-dimensional feature vector encoding the explanatory power of each visual patch in the training set. Although the approximate nearest neighbors search scheme that we use provides us sub-linear search time, our method

still exhibits high computation time while predicting an image. This is especially true when we make use of the geometrically consistent SNN kernel. Another noticeable drawback of this method is that it requires high memory costs to store all the local features in memory despite the fact that our indexing techniques allows us to compress them. Then, as a last contribution, we propose an approach to reduce both memory cost and computation time while predicting an image. To this aim, we investigate a way to drastically reduce the dimensionality of such over-complete representation thanks to a recursive feature elimination method. We show that the number of *spatial atoms* of the representation can be reduced by up to two orders of magnitude without much degrading the encoded information. We also qualitatively demonstrate the ability of our method to identify which image regions were mostly used by the learning algorithm to model the different visual concepts of a specific domain knowledge. This method can be seen as a supervised method for learning a compact and highly interpretable visual vocabulary composed of *discriminant* and *spatially localized* visual patches. The main originality of this work relies on this interpretability objective that most visual representations used in classical visual models are too abstract to fulfill. Indeed, the resulting visual representations do not rely on directly *visualizable* atoms, *i.e.* atoms that can be mapped without any ambiguity onto localized visual contents in the training set.

7.2 Discussion and Future Works

7.2.1 Limitations of our approaches

One of the main drawbacks of our approach compared to CNNs is the lack of generalization and genericity to different visual contents. Indeed, as explained in the previous section, we exhibit better classification performance than CNN on a restricted number of classification tasks. This is mainly due to the good generalization ability of CNNs when learning from a lot of training data. Their main strength relies on the fact that the layers correspond to trainable modules that are jointly learned in a supervised manner so as to progressively embed the image from the pixel space to more abstract representations. Thus, contrary to most aggregation-based methods proposed in the computer vision literature (that rely on handcrafted local feature schemes), such approaches allow learning discriminant visual patterns that are adapted to the classification task. As other aggregation-based methods, our approach requires cross-validating the choice of the underlying local feature scheme or to use a complex combination procedure between several local feature schemes so as to adapt to different classification tasks. In the next section, we suggest interesting future research directions to improve the genericity of our approach.

Furthermore, although we propose a method to compress the size of our visual vocabulary, our approach still requires memory costs and computation time greater than those involved in CNNs while predicting an image. Another noticeable draw-

back of this approach is that the first iterations of our visual vocabulary learning procedure (introduced in Chapter 6) requires to represent the images onto a large set of potentially redundant spatial atoms. This implies a high memory cost for the first iterations and that the supervised model learned on top of these very high-dimensional image representations is likely to overfit during the first iterations. This suggests that we should find a way to select the discriminant visual atoms in an online way rather than *a posteriori* from an initial large visual vocabulary.

7.2.2 Toward more generic image representations

A first way to allow our system to learn discriminant patterns in a tractable way would be to consider the SNN representation as a two layered architecture that can be trained with backpropagation. The input layer would be an input tensor of dimension wxhxd corresponding to the input image from which d-dimensional local descriptors are densely extracted. The first processing layer would correspond to a bunch of k d-dimensional filters constrained to be equal to k local features of the training dataset. These filters would correspond to our visual vocabulary. Then, the next layer would be a sum pooling layer producing a global image representation on top of what a fully connected layer (*e.g* a logistic regression classifier) could be stacked. The learning algorithm would consist in preforming standard backpropagation to compute the update values for each filter. The model would be constrained to update each filter to a position in the parameter space corresponding to an existing local descriptor of the training set. This could be done by finding the nearest neighbor descriptor of the updated filter. This can be seen as a regularization technique that constrains the model to learn discriminant visual patterns corresponding to spatially localized descriptors of the training set. In this way, we would keep the interpretability property of our previously proposed approach while learning visual patterns adapted to the classification task. Note that this version of the SNN representation would not exactly correspond to one proposed in Chapter 4. Indeed, as we need to compute derivatives to perform backpropagation, we have to replace the rank-based activation function by another differentiable non-linearity. Moreover, it would be of great interest to integrate strong geometry in this supervised SNN representation.

This approach is similar to the one depicted in [AGT+15] where the authors propose to directly integrate dictionary learning objective term into the supervised cost of a deep architecture. This allows combining the good properties of both deep learning methods and aggregation-based methods. This suggests that we could also investigate on the extension of the supervised SNN representation to a deep architecture where different layers could be jointly learned. We hope that this would allow us to improve the generalization and genericity of our approach while still conserving the interpretability of the learned visual patterns.

7.2.3 Toward interactive interpretable visual dictionary learning methods

Despite recent progress in computer vision and machine learning, automatically recognizing any object of the real world remains difficult because of the lack of training data in sufficient quantity and quality. This is particularly true for fine-grained visual knowledge that requires high expertise. A good alternative would be to consider alleviating this lack of annotated data with unsupervised learning algorithms. These methods have been widely considered in the 2000's to improve the generalization ability of deep learning methods on supervised tasks with a small number of training examples. Unsupervised learning is an attractive research area that is mainly motivated by the impressive generalization ability of human beings that probably comes from the fact that we continuously update our model of the world with unsupervised learning mechanisms. For such learning algorithms, we are not provided any labels at training time, and the goal is to learn how to represent data by keeping interesting information and throwing away useless ones. Nowadays, unsupervised learning is still a very challenging issue because it is hard to define what kind of information can be considered as relevant or not. Moreover, as for supervised learning, such methods are prone to be strongly affected by the fact that real world domain specific knowledge usually involve highly imbalanced training datasets. This implies that it is difficult for such algorithms to capture enough diverse and discriminant patterns for such unequally and poorly populated domain specific classes which only contain few instances in the training set.

Then, a very interesting challenge would be to *i)* integrate domain expertise of users into such learning procedures and to *ii)* transfer the knowledge of these learning algorithms to humans so that we can gain insights into which part of the data is used by the learning algorithm to discriminate between different visual concepts. The spatially localized vocabulary learning technique that we introduced in Chapter 6 is actually a first attempt toward this challenging objective. A promising extension of this baseline would be to consider a semi-supervised learning framework by proposing collaborative and interactive learning procedures allowing domain experts to directly interact with the visual vocabulary. Active learning and *Relevance Feed-Back* [RHOM98, FCB04] strategies have been successfully applied in machine learning and image retrieval context [OCB10, FBC08, CEOT08, GCB05]. Such methods would allow domain experts to understand which visual patterns the learning algorithm detected as discriminant or ambiguous from a concept to another. Then, the experts could improve their knowledge of the objects of interest by taking advantage of the learning algorithms to discover fine relevant details. Secondly, this would also allow the experts to detect some errors or limitations in the machine learning process and to use their own domain expertize to interactively correct the model. Iterating over these two phases would allow creating a virtuous circle between users and machines so as to increasingly enrich the visual knowledge of the targeted objects by produ-

cing realistic and verified observations such as training data or interpretable visual models.

Bibliographie

[AGMM15] Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. *arXiv preprint arXiv :1503.00778*, 2015.

[AGT+15] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad : Cnn architecture for weakly supervised place recognition. *arXiv preprint arXiv :1511.07247*, 2015.

[AHK01] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434. Springer, 2001.

[Ama87] Shun-ichi Amari. Differential geometrical theory of statistics. *IMS Monograph vol. 10, Differential Geometry in Statistical Inference*, pages 20–94, 1987.

[AZ12] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918. IEEE, 2012.

[BBDBS10] Lamberto Ballan, Marco Bertini, Alberto Del Bimbo, and Giuseppe Serra. Video event classification using string kernels. *Multimedia Tools and Applications*, 48(1) :69–87, 2010.

[BBLP10] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2559–2566. IEEE, 2010.

[BBM05] Alexander C Berg, Tamara L Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 26–33. IEEE, 2005.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828, 2013.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :509–517, 1975.

[Ben93] Yoshua Bengio. A connectionist approach to speech recognition. *International Journal on Pattern Recognition and Artificial Intelligence*, 7(4) :647–668, 1993.

[Ben09] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1) :1–127, 2009.

[Bez81] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[BH07] Rachid Benmokhtar and Benoit Huet. Neural network combining classifier based on dempster-shafer theory for semantic indexing in video content. In *International Conference on Multimedia Modeling*, pages 196–205. Springer, 2007.

[Bis06] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[BK88] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5) :291–294, 1988.

[BK00] Issam Bazzi and Dina Katabi. Using support vector machines for spoken digit recognition. In *INTERSPEECH*, pages 433–436, 2000.

[BKL06] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM, 2006.

[BLP+07] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19 :153, 2007.

[BMW+11] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential Deep Learning for Human Action Recognition. In B. Lepri A.A. Salah, editor, *2nd International Workshop on Human Behavior Understanding (HBU)*, Lecture Notes in Computer Science, pages 29–39. Springer, November 2011.

[BMW+12] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sparse Shift-Invariant Representation of Local 2D Patterns and Sequence Learning for Human Action Recognition. In IEEE, editor, *21st International Conference on Pattern Recognition (ICPR)*, November 2012.

[Bor07] Karsten Michael Borgwardt. *Graph kernels*. PhD thesis, lmu, 2007.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Bot12] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks : Tricks of the Trade*, pages 421–436. Springer, 2012.

[Bou06] David Bouchain. Character recognition using convolutional neural networks. *Institute for Neural Information Processing*, 2007, 2006.

[BRF10] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *Advances in neural information processing systems*, pages 244–252, 2010.

[BS09] Liefeng Bo and Cristian Sminchisescu. Efficient match kernel between sets of features for visual recognition. In *Advances in neural information processing systems*, pages 135–143, 2009.

[BSI08] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[BT09] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1) :183–202, 2009.

[BTB05a] Sabri Boughorbel, J-P Tarel, and Nozha Boujemaa. Generalized histogram intersection kernel for image recognition. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–161. IEEE, 2005.

[BTB05b] Sabri Boughorbel, Jean Philippe Tarel, and Nozha Boujemaa. The intermediate matching kernel for image local features. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 889–894. IEEE, 2005.

[BTF04] Sabri Boughorbel, Jean-Philippe Tarel, and Francois Fleuret. Non-mercer kernels for svm object recognition. In *BMVC*, pages 1–10, 2004.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[Bül] Arbeitsgruppe Bülthoff. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*.

[CEOT08] Michel Crucianu, Daniel Estevez, Vincent Oria, and Jean-Philippe Tarel. Speeding up active relevance feedback with approximate knn retrieval for hyperplane queries. *International Journal of Imaging Systems and Technology*, 18(2-3) :150–159, 2008.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[CJ10] Barbara Caputo and Luo Jie. A performance evaluation of exact and approximate match kernels for object recognition. *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, 8(3), 2010.

[CK08] Olivier Chapelle and S Sathiya Keerthi. Multi-class feature selection with support vector machines. In *Proceedings of the American statistical association*, 2008.

[CLVZ11] Ken Chatfield, Victor S Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details : an evaluation of recent feature encoding methods. In *BMVC*, volume 2, page 8, 2011.

[CLZ13] Y. Chai, V. Lempitsky, and A. Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *IEEE International Conference on Computer Vision*, 2013.

[CM05] Ondrej Chum and Jiri Matas. Matching with prosac-progressive sample consensus. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 220–226. IEEE, 2005.

[CMK03] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[Cov65] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3) :326–334, 1965.

[CPZ97] Paolo Ciaccia, Marco Patella, and Pavel Zezula. Deis-csite-cnr. In *Proceedings of the... International Conference on Very Large Data Bases*, volume 23, page 426. Morgan Kaufmann Pub, 1997.

[CVM04] Antoni B Chan, Nuno Vasconcelos, and Pedro J Moreno. A family of probabilistic kernels based on information divergence. *Univ. California, San Diego, CA, Tech. Rep. SVCL-TR-2004-1*, 2004.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314, 1989.

[CZ97] Yair Censor and Stavros Andrea Zenios. *Parallel optimization : Theory, algorithms, and applications*. Oxford University Press on Demand, 1997.

[DDDM04] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11) :1413–1457, 2004.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[Del13] Jonathan Delhumeau. Jonathan delhumeau, philippe-henri gosselin, hervé jégou, patrick pérez. 2013.

[DG08] Stefan Duffner and Christophe Garcia. Robust Face Alignment Using Convolutional Neural Networks. In *International Conference on Computer Vision Theory and Applications (VISAPP 2008)*, pages 30–37, January 2008.

[DJP11] Olivier Duchenne, Armand Joulin, and Jean Ponce. A graph-matching kernel for object categorization. In *2011 International Conference on Computer Vision*, pages 1792–1799. IEEE, 2011.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[DRS11] Matthijs Douze, Arnau Ramisa, and Cordelia Schmid. Combining attributes and fisher vectors for efficient image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 745–752. IEEE, 2011.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[Duf08] Stefan Duffner. Face image analysis with convolutional neural networks. 2008.

[EGMS14] Khaoula Elagouni, Christophe Garcia, Franck Mamalet, and Pascale Sébillot. Text recognition in multimedia documents : a study of two neural-based ocrs using and avoiding character segmentation. *International Journal on Document Analysis and Recognition (IJDAR)*, 17(1) :19–31, 2014.

[EHJ+04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2) :407–499, 2004.

[ESK02] Levent Ertoz, Michael Steinbach, and Vipin Kumar. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*, pages 105–115, 2002.

[EVGW+a] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http ://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[EVGW+b] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http ://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395, 1981.

[FBC08] Marin Ferecatu, Nozha Boujemaa, and Michel Crucianu. Semantic interactive image retrieval combining visual and conceptual content description. *Multimedia systems*, 13(5-6) :309–322, 2008.

[FBF77] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3) :209–226, 1977.

[FCB04] Marin Ferecatu, Michel Crucianu, and Nozha Boujemaa. Tuning svm-based relevance feedback for the interactive classification of images. In *EWIMT*, 2004.

[FCH+08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear : A library for large linear classification. *Journal of machine learning research*, 9(Aug) :1871–1874, 2008.

[FCNL12] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. *arXiv preprint arXiv :1202.2160*, 2012.

[FCNL13] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1915–1929, 2013.

[FFFP06] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4) :594–611, 2006.

[FH05] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1) :55–79, 2005.

[Fuk80] Kunihiko Fukushima. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4) :193–202, 1980.

[Gar09] Christophe Garcia. *Apprentissage automatique en analyse de visages, pour l'indexation d'images et les interfaces avancées*. Habilitation à diriger des recherches, INSA de Lyon, November 2009.

[GCB05] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Active semi-supervised fuzzy clustering for image database categorization. In *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 9–16. ACM, 2005.

[GD05] Kristen Grauman and Trevor Darrell. The pyramid match kernel : Discriminative classification with sets of image features. In *Tenth IEEE*

International Conference on Computer Vision (ICCV'05) Volume 1, volume 2, pages 1458–1465. IEEE, 2005.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[GHP00] Aravind Ganapathiraju, Jonathan Hamaker, and Joseph Picone. Hybrid svm/hmm architectures for speech recognition. In *INTERSPEECH*, pages 504–507. Citeseer, 2000.

[GHP07] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[GHRS04] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2004.

[GIM+99] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[GL10] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 399–406, 2010.

[GL11] Yunchao Gong and Svetlana Lazebnik. Iterative quantization : A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. IEEE, 2011.

[GLS06] Thomas Gärtner, Germany Quoc V Le, and Alex J Smola. 1 a short tour of kernel methods for graphs. 2006.

[GMJP14] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49 :92–98, 2014.

[GS00] Theo Gevers and Arnold WM Smeulders. Pictoseek : Combining color and shape invariant features for image retrieval. *IEEE transactions on Image Processing*, 9(1) :102–119, 2000.

[GTC10] Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia. Kernel sparse representation for image classification and face recognition. In *European Conference on Computer Vision*, pages 1–14. Springer, 2010.

[Gut84] Antonin Guttman. *R-trees : a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[GWBV02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3) :389–422, 2002.

[Hau99] David Haussler. Convolution kernels on discrete structures. Technical report, Citeseer, 1999.

[HBL+07] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[HH99] Benoit Huet and Edwin R Hancock. Shape recognition from large image libraries by inexact graph matching. *Pattern Recognition Letters*, 20(11) :1259–1269, 1999.

[Hin02] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8) :1771–1800, 2002.

[HKK+10] Michael E Houle, Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Can shared-neighbor distances defeat the curse of dimensionality ? In *International Conference on Scientific and Statistical Database Management*, pages 482–500. Springer, 2010.

[HKM+99] Jing Huang, S Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Spatial color indexing and applications. *International Journal of Computer Vision*, 35(3) :245–268, 1999.

[HKQ10] Jian Hou, Jianxin Kang, and Naiming Qi. On vocabulary size in bag-of-visual-words representation. In *Pacific-Rim Conference on Multimedia*, pages 414–424. Springer, 2010.

[HLH+12] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012.

[HN04] Simon Haykin and Neural Network. A comprehensive foundation. *Neural Networks*, 2(2004), 2004.

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2) :251–257, 1991.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7) :1527–1554, 2006.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv :1512.03385*, 2015.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors : towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv :1502.03167*, 2015.

[ITGJ15] Ahmet Iscen, Giorgos Tolias, Philippe-Henri Gosselin, and Hervé Jégou. A comparison of dense region detectors for image search and fine-grained classification. *IEEE Transactions on Image Processing*, 24(8) :2369–2381, 2015.

[JB08] Alexis Joly and Olivier Buisson. A posteriori multi-probe locality sensitive hashing. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 209–218. ACM, 2008.

[JB09] Alexis Joly and Olivier Buisson. Logo retrieval with a contrario visual query expansion. In *MM '09 : Proceedings of the seventeen ACM international conference on Multimedia*, pages 581–584, 2009.

[JB11] Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 873–880. IEEE, 2011.

[JBJG12] Mihir Jain, Rachid Benmokhtar, Hervé Jégou, and Patrick Gros. Hamming embedding similarity-based image classification. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 19. ACM, 2012.

[JCB14] Alexis Joly, Julien Champ, and Olivier Buisson. Instance-based bird species identication with undiscriminant features pruning-lifeclef 2014. In *CLEF2014*, 2014.

[JDS08] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008.

[JDS09] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1169–1176. IEEE, 2009.

[JDS10] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3) :316–336, 2010.

[JDS11] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1) :117–128, 2011.

[JGG+15] Alexis Joly, Hervé Goëau, Hervé Glotin, Concetto Spampinato, Pierre Bonnet, Willem-Pier Vellinga, Robert Planqué, Andreas Rauber, Simone Palazzo, Bob Fisher, et al. Lifeclef 2015 : multimedia life species

identification challenges. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 462–483. Springer, 2015.

[JGP14] Ziheng Jiang, Ping Guo, and Lihong Peng. Locality-constrained low-rank coding for image classification. In *AAAI*, pages 2780–2786, 2014.

[JH+99] Tommi S Jaakkola, David Haussler, et al. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

[JP73] Raymond A Jarvis and Edward A Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 100(11) :1025–1034, 1973.

[JPD+12] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9) :1704–1716, 2012.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe : Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[JXYY13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1) :221–231, 2013.

[KB05] Tron Krosshaug and Roald Bahr. A model-based image-matching technique for three-dimensional reconstruction of human motion from uncalibrated video sequences. *Journal of biomechanics*, 38(4) :919–929, 2005.

[KFS05] Kwang In Kim, Matthias O Franz, and Bernhard Scholkopf. Iterative kernel principal component analysis for image modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9) :1351–1366, 2005.

[KH09] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[KLL01] Ja Seong Ku, Kyoung Mu Lee, and Sang Uk Lee. Multi-image matching for a general motion stereo camera model. *Pattern Recognition*, 34(9) :1701–1712, 2001.

[KM11] Piotr Koniusz and Krystian Mikolajczyk. Spatial coordinate coding to reduce histogram representations, dominant angle and colour pyramid match. In *2011 18th IEEE International Conference on Image Processing*, pages 661–664. IEEE, 2011.

[KO16] Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *arXiv preprint arXiv :1601.08169*, 2016.

[Koc15] Gregory Koch. *Siamese neural networks for one-shot image recognition.* PhD thesis, University of Toronto, 2015.

[KPFJ14] Josip Krapac, Florent Perronnin, Teddy Furon, and Hervé Jégou. Instance classification with prototype selection. In *Proceedings of International Conference on Multimedia Retrieval*, page 431. ACM, 2014.

[KRL10] Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. Fast inference in sparse coding algorithms with applications to object recognition. *arXiv preprint arXiv :1010.3467*, 2010.

[KS04] Yan Ke and Rahul Sukthankar. Pca-sift : A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE, 2004.

[KSDFF13] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LB95] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 255–257. MIT Press, 1995.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[LBOM12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks : Tricks of the trade*, pages 9–48. Springer, 2012.

[LBRN06] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

[LCJB+89] Yann Le Cun, LD Jackel, B Boser, JS Denker, HP Graf, I Guyon, D Henderson, RE Howard, and W Hubbard. Handwritten digit recognition : Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11) :41–46, 1989.

[LEC+04] Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4) :467–476, 2004.

[LGTB97] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition : A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1) :98–113, 1997.

[LH05] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1482–1489. IEEE, 2005.

[LJW+07] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh : efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment, 2007.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110, 2004.

[LS03] Bastian Leibe and Bernt Schiele. Analyzing appearance and contour based methods for object categorization. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–409. IEEE, 2003.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[LSST+02] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb) :419–444, 2002.

[LV09] John A Lee and Michel Verleysen. Simbed : Similarity-based embedding. In *International Conference on Artificial Neural Networks*, pages 95–104. Springer, 2009.

[LWL11] Lingqiao Liu, Lei Wang, and Xinwang Liu. In defense of soft-assignment coding. In *2011 International Conference on Computer Vision*, pages 2486–2493. IEEE, 2011.

[Lyu05] Siwei Lyu. Mercer kernels for object recognition with local features. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 223–229. IEEE, 2005.

[MBPS09] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[MCO16] Gaétan Marceau-Caron and Yann Ollivier. Practical riemannian neural networks. *arXiv preprint arXiv :1602.08007*, 2016.

[MCUP04] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10) :761–767, 2004.

[MHV03] Pedro J Moreno, Purdy P Ho, and Nuno Vasconcelos. A kullback-leibler divergence based kernel for svm classification in multimedia applications. In *Advances in neural information processing systems*, page None, 2003.

[MKR+13] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[ML09] Marius Muja and David Lowe. Flann-fast library for approximate nearest neighbors user manual. *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, 2009.

[ML12] Sancho McCann and David G Lowe. Local naive bayes nearest neighbor for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3650–3656. IEEE, 2012.

[ML14] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11) :2227–2240, 2014.

[MP14] Naila Murray and Florent Perronnin. Generalized max pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2473–2480, 2014.

[MPCM10] Andrej Mikulík, Michal Perdoch, Ondřej Chum, and Jiří Matas. Learning a fine vocabulary. In *European Conference on Computer Vision*, pages 1–14. Springer, 2010.

[MPHG91] David Marr, Tomaso Poggio, Ellen C Hildreth, and W Eric L Grimson. A computational theory of human stereo vision. In *From the Retina to the Neocortex*, pages 263–295. Springer, 1991.

[MPS+09] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040, 2009.

[MS02] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *European conference on computer vision*, pages 128–142. Springer, 2002.

[MS04a] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1) :63–86, 2004.

[MS04b] Lionel Moisan and Bérenger Stival. A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix. *International Journal of Computer Vision*, 57(3) :201–218, 2004.

[MS05] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10) :1615–1630, 2005.

[MY09] Jean-Michel Morel and Guoshen Yu. Asift : A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2) :438–469, 2009.

[NS06] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. IEEE, 2006.

[NZ08] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[NZSF96] Yoshihiko Nomura, Dill Zhang, Yuko Sakaida, and Seizo Fujii. 3-d object pose estimation based on iterative image matching : Shading and edge data fusion. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 513–517. IEEE, 1996.

[OCB10] Wajih Ouertani, Michel Crucianu, and Nozha Boujemaa. Interactive learning of heterogeneous visual concepts with local features. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 995–998. ACM, 2010.

[OF97] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set : A strategy employed by v1 ? *Vision research*, 37(23) :3311–3325, 1997.

[Oll13] Yann Ollivier. Riemannian metrics for neural networks i : feedforward networks. *arXiv preprint arXiv :1303.0818*, 2013.

[Oll15] Yann Ollivier. Riemannian metrics for neural networks i : feedforward networks. *Information and Inference*, 4(2) :108–153, 2015.

[OT01] Aude Oliva and Antonio Torralba. Modeling the shape of the scene : A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3) :145–175, 2001.

[Par15] Sylvain Paris. Methods for 3d reconstruction from multiple images. *Saatavissa : http ://people. csail. mit. edu/sparis/talks/Paris_ 06_ 3D_ Reconstruction. pdf. Hakupäivä*, 23, 2015.

[PCI+07] 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.

[PCI+08] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization : Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[PCM09] Michal Perd'och, Ondrej Chum, and Jiri Matas. Efficient representation of local geometry for large scale object retrieval. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 9–16. IEEE, 2009.

[PCS09] Sébastien Poullot, Michel Crucianu, and Shin'Ichi Satoh. Indexing local configurations of features for scalable content-based video copy detection. In *Proceedings of the First ACM workshop on Large-scale Multimedia Retrieval and Mining*, pages 43–50. ACM, 2009.

[PD07] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11) :559–572, 1901.

[PKF07] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72(2) :179–193, 2007.

[PKVVG00] Marc Pollefeys, Reinhard Koch, Maarten Vergauwen, and Luc Van Gool. Automated reconstruction of 3d scenes from sequences of images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 55(4) :251–267, 2000.

[PL15] Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks : A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3743–3752, 2015.

[PLSP10] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384–3391. IEEE, 2010.

[PNF+08] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3) :143–167, 2008.

[PSM10] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision*, pages 143–156. Springer, 2010.

[PSZ11] James Philbin, Josef Sivic, and Andrew Zisserman. Geometric latent dirichlet allocation on a matching graph for large-scale image datasets. *International journal of computer vision*, 95(2) :138–153, 2011.

[PT03] Simon Perkins and James Theiler. Online feature selection using grafting. In *ICML*, pages 592–599, 2003.

[PWQP14] Xiaojiang Peng, Limin Wang, Yu Qiao, and Qiang Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In *European Conference on Computer Vision*, pages 660–674. Springer, 2014.

[PZQP14] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *European Conference on Computer Vision*, pages 581–595. Springer, 2014.

[QSG13] Zhiwei Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2) :143–169, 2013.

[RBH+15] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

[RC] Arushi Raghuvanshi and Vivek Choksi. Facial expression recognition with convolutional neural networks.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[RDGM10] Julien Rabin, Julie Delon, Yann Gousseau, and Lionel Moisan. Mac-ransac : a robust algorithm for the recognition of multiple objects. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPTV 2010)*, page 051, 2010.

[RHOM98] Yong Rui, Thomas S Huang, Michael Ortega, and Sharad Mehrotra. Relevance feedback : a power tool for interactive content-based image retrieval. *IEEE Transactions on circuits and systems for video technology*, 8(5) :644–655, 1998.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[RHW88] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3) :1, 1988.

[Ros58] Frank Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386, 1958.

[RPLvZ11] Stefan Romberg, Lluis Garcia Pueyo, Rainer Lienhart, and Roelof van Zwol. Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 25 :1–25 :8, New York, NY, USA, 2011. ACM.

[RT01] Yossi Rubner and Carlo Tomasi. The earth mover's distance. In *Perceptual Metrics for Image Database Navigation*, pages 13–28. Springer, 2001.

[RVM+11] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders : Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

[SAH08] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[SB91] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1) :11–32, 1991.

[SG07] Zohra Saidane and Christophe Garcia. Automatic scene text recognition using a convolutional neural network. In *Workshop on Camera-Based Document Analysis and Recognition*, volume 1, 2007.

[SH07a] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *RBM*, 500(3) :500, 2007.

[SH07b] Ruslan Salakhutdinov and Geoffrey E Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, pages 412–419, 2007.

[Sha09] Amnon Shashua. Introduction to machine learning : Class notes 67577. *arXiv preprint arXiv :0904.3664*, 2009.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1) :1929–1958, 2014.

[SIV16] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv :1602.07261*, 2016.

[SJ72] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1) :11–21, 1972.

[SKH08] Alexander Shekhovtsov, Ivan Kovtun, and Václav Hlaváč. Efficient mrf deformation model for non-rigid image matching. *Computer Vision and Image Understanding*, 112(1) :91–99, 2008.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[SLK09] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Scramsac : Improving ransac's efficiency with a spatial consistency filter. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2090–2097. IEEE, 2009.

[SMH05] Fabrice Souvannavong, Bernard Merialdo, and Benoit Huet. Region-based video content indexing and retrieval. In *Proc. of 4th International Workshop on Content-Based Multimedia Indexing, Riga, Latvia.* Citeseer, 2005.

[Smo86] Paul Smolensky. Information processing in dynamical systems : Foundations of harmony theory. Technical report, DTIC Document, 1986.

[SP14] Dan Stowell and Mark D Plumbley. Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ*, 2 :e488, 2014.

[SPMV13] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector : Theory and practice. *International journal of computer vision*, 105(3) :222–245, 2013.

[SSL14] Vladyslav Sydorov, Mayu Sakurada, and Christoph H. Lampert. Deep fisher kernels - end to end learning of the fisher kernel gmm parameters. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep fisher networks for large-scale image classification. In *Advances in neural information processing systems*, pages 163–171, 2013.

[SZ03] Josef Sivic and Andrew Zisserman. Video google : A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

[SZ14a] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[SZ14b] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

[TAJ16] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. Image search with selective match kernels : aggregation across single and multiple images. *International Journal of Computer Vision*, 116(3) :247–261, 2016.

[TBFJ15] Giorgos Tolias, Andrei Bursuc, Teddy Furon, and Hervé Jégou. Rotation and translation covariant match kernels for image retrieval. *Computer Vision and Image Understanding*, 140 :9–20, 2015.

[TFJ14] Giorgos Tolias, Teddy Furon, and Hervé Jégou. Orientation covariant aggregation of local descriptors with embeddings. In *European Conference on Computer Vision*, pages 382–397. Springer, 2014.

[TFSD11] Tinne Tuytelaars, Mario Fritz, Kate Saenko, and Trevor Darrell. The nbnn kernel. In *2011 International Conference on Computer Vision*, pages 1824–1831. IEEE, 2011.

[Tha06] Arasanathan Thayananthan. *Template-based pose estimation and tracking of 3D hand motion*. PhD thesis, University of Cambridge, 2006.

[TLSJ12] Feng Tang, Huan Lu, Tanfeng Sun, and Xinghao Jiang. Efficient image classification using sparse coding and random forest. In *Image and Signal Processing (CISP), 2012 5th International Congress on*, pages 781–785. IEEE, 2012.

[Tse01] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3) :475–494, 2001.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface : Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[Vap99] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5) :988–999, 1999.

[VdEPV93] Petra A Van den Elsen, E-JD Pol, and Max A Viergever. Medical image matching-a review with classification. *IEEE Engineering in Medicine and Biology Magazine*, 12(1) :26–39, 1993.

[VDSGS10] Koen Van De Sande, Theo Gevers, and Cees Snoek. Evaluating color descriptors for object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 32(9) :1582–1596, 2010.

[VF05] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International Work-Conference on Artificial Neural Networks*, pages 758–770. Springer, 2005.

[VGGVS08] Jan C Van Gemert, Jan-Mark Geusebroek, Cor J Veenman, and Arnold WM Smeulders. Kernel codebooks for scene categorization. In *European conference on computer vision*, pages 696–709. Springer, 2008.

[VGVSG10] Jan C Van Gemert, Cor J Veenman, Arnold WM Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *IEEE transactions on pattern analysis and machine intelligence*, 32(7) :1271–1283, 2010.

[VLL+10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec) :3371–3408, 2010.

[VSKB10] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr) :1201–1242, 2010.

[VZ12] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 34(3) :480–492, 2012.

[Wan01] James Z Wang. Image classification by image matching. In *Integrated Region-Based Image Retrieval*, pages 105–122. Springer, 2001.

[WB97] Roger Weber and Stephen Blott. An approximation based data structure for similarity search. *Report TR1997b, ETH Zentrum, Zurich, Switzerland*, 1997.

[WBW+11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[WCG03] Christian Wallraven, Barbara Caputo, and Arnulf Graf. Recognition with local features : the kernel recipe. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 257–264. IEEE, 2003.

[WDJ11] Christian Wengert, Matthijs Douze, and Hervé Jégou. Bag-of-colors for improved image search. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1437–1440. ACM, 2011.

[WL08] Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244, 2008.

[WLG+15] Pichao Wang, Wanqing Li, Zhimin Gao, Jing Zhang, Chang Tang, and Philip Ogunbona. Deep convolutional neural networks for action recognition using depth map sequences. *arXiv preprint arXiv :1501.04686*, 2015.

[WLKC16] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data—a survey. *Proceedings of the IEEE*, 104(1) :34–57, 2016.

[WP14] Liping Wang and Juncheng Pu. Image classification algorithm based on sparse coding. *Journal of Multimedia*, 9(1) :114–122, 2014.

[WSSJ14] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search : A survey. *arXiv preprint arXiv :1408.2927*, 2014.

[WTF09] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[WWCN12] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[WWFW98] James Ze Wang, Gio Wiederhold, Oscar Firschein, and Sha Xin Wei. Content-based image indexing and searching using daubechies' wavelets. *International Journal on Digital Libraries*, 1(4) :311–328, 1998.

[WWH+14] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval : A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166. ACM, 2014.

[WYY+10] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE, 2010.

[WZNM14] Feng Wang, Wan-Lei Zhao, Chong-Wah Ngo, and Bernard Merialdo. A hamming embedding kernel with informative bag-of-visual words for video semantic indexing. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 10(3) :26, 2014.

[YBV13] Shaoyi Yin, Mehdi Badr, and Dan Vodislav. Dynamic multi-probe lsh : An i/o efficient index structure for approximate nearest neighbor search. In *International Conference on Database and Expert Systems Applications*, pages 48–62. Springer, 2013.

[YJHN07] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 197–206. ACM, 2007.

[YYGH09] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.

[YYH10] Jianchao Yang, Kai Yu, and Thomas Huang. Supervised translation-invariant sparse coding. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3517–3524. IEEE, 2010.

[YZG09] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *Advances in neural information processing systems*, pages 2223–2231, 2009.

[ZBMM06] Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. Svm-knn : Discriminative nearest neighbor classification for visual category recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2126–2136. IEEE, 2006.

[ZDI+15] Lilei Zheng, Stefan Duffner, Khalid Idrissi, Christophe Garcia, and Atilla Baskurt. Siamese Multi-layer Perceptrons for Dimensionality Reduction and Face Identification. *Multimedia Tools and Applications*, pages „ 2015.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[ZGL+13] Tianzhu Zhang, Bernard Ghanem, Si Liu, Changsheng Xu, and Narendra Ahuja. Low-rank sparse coding for image classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 281–288, 2013.

[ZJG13] Wan-Lei Zhao, Hervé Jégou, and Guillaume Gravier. Oriented pooling for dense and non-dense rotation-invariant features. In *BMVC-24th British Machine Vision Conference*, 2013.

[ZKTF10] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[ZMGL15] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv :1506.02351*, 2015.

[ZWH+13] Chunjie Zhang, Shuhui Wang, Qingming Huang, Jing Liu, Chao Liang, and Qi Tian. Image classification using spatial pyramid robust sparse coding. *Pattern Recognition Letters*, 34(9) :1046–1052, 2013.