Feature selection, sparse coding and normalization for material image Classification

Sixiang Xu

N°d'ordre :2021LYSES046

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de :

Université Jean Monnet Saint-Étienne

Ecole Doctorale ED SIS N°488

Sciences, Ingénierie, Santé

Spécialité/discipline de doctorat :

Informatique, Image Vision

Soutenue publiquement le 14/12/2021,par :

Sixiang Xu

Feature Selection, Sparse Coding and Normalization for Material Image Classification

---

Devant le jury composé de :

Maria Vanrell Martorell	PR, Universitat Autònoma de Barcelona, CVC	Rapporteure
Jochen Lang	PR, Université d'Ottawa, school of EECS	Rapporteur
Nicolas Thome	PR, CNAM, Laboratoire Cédric	Examinateur
Gabriela Csurka Khedari	Chargé de recherche, Naver Labs Europe	Examinatrice
Christophe Ducottet	PR, Université Jean-Monnet, Hubert Curien	Examinateur
Alain Trémeau	PR, Université Jean-Monnet, Hubert Curien	Directeur
Damien Muselet	MCF, Université Jean-Monnet, Hubert Curien	Encadrant

A time will come to ride the wind and cleave the waves, I'll set my cloudlike sail to cross the sea which raves

— Li Bai *'Hard is the way of the world'*

Abstract

Image classification, which consists of predicting a single class for each input image, is a core subject in the computer vision community. And as one of its tasks, material classification from an image is challenging for humans but also for computer systems because materials may have various appearances depending on their surface properties, lighting geometry, viewing geometry, camera settings, etc.

In the beginning, first material image datasets are created with these dependencies well controlled and known. In addition to images and their categories, dependencies information is also provided as complementary features. Many works have been proposed, which achieve high performance on the classification task. However, their generality to real-world application is limited because only a few of material instances represent a material category. Furthermore, some methods use dependencies as features. They must be measured before the classification and that leads inevitably less efficiency.

In the recent years, new material datasets tend to be in large scale and to be without any dependencies. All the images are taken in real-world environment, instead of the laboratory. These datasets, no doubt, are more

challenging than ever and a closer fit to real-world application. Moreover, a revolutionary image classification architecture, Convolutional Neural Network(CNN), has emerged and has shown high performance in large-scale image datasets, like ImageNet, which is dedicated to object classification in the real word. This architecture makes real-world material classification of high accuracy possible. This thesis investigates how to implement appropriately the CNN, which is pretrained by ImageNet, into material classification task. Generally, we call this process transfer learning because we transfer the knowledge learned in ImageNet to our task.

To this end, our two approaches are reported. They all work on aggregating features extracted by the CNN into a more powerful representation for classification, however both are in totally different ways. The first one consists in selecting more discriminative features from all the candidates with a criterion, called confidence score, showing how confident the classifier is to its prediction. We assume that features with high confidence score are more discriminative. Fisher vector is a state-of-the art feature aggregation approach. The second approach ameliorates the fisher vector representation when applied to CNN's features. With some modification, we embed it as a module into the CNN and allow it and other components of the CNN to be trained together under the classification supervision. To validate our solutions, we test them on the several widely-used datasets and compare them with recent state-of-the art approaches, showing their competitive performance. We also conduct ablation studies in order to study how our solutions achieve good performance.

key words: Image classification, material classification, orderless pooling, transfer learning, confidence score, fisher score, sparse coding

Acknowledgement

First of all, I would like to thank my supervisors, accompanying me with their endless patience during my three-years adventure. Thank you Alain. You were always helping me, especially when Damien worked in Canada. And you encouraged me to participate in many academic forums, like the summer school ICVSS. And you too, Damien. Countless emails and meetings between us are enough to prove that you play an important role in my Ph.D. candidate career. I am still keeping grateful when remembering that you and your family shared your personal time with me and invited me to visit Canada Museum of History. What warm welcoming you gave! And my Canadian supervisor, Robert, without you, my trip at the University of Ottawa would never have happened. Thanks equally for your valuable advice to my work published in the IVCNZ conference.

Then, I want to give a big hug to my parents. The call at 3.PM every Saturday is the most precious connection between us for 6 years. Your encouragement and positive attitude towards life help me regain confidence when I was in difficulties. You are my the most solid rock!

Fortunately , I always have the most friendly homemates. Jian and Chen,

we take care of the other two for almost 3 years. I feel really relaxed when coming back to our apartment and then sharing my feelings about the day. No matter my feelings were good or bad, you were always there and listened. Please remember our collective achievement: we succeeded in cooking many hometown specialties in France! Ramesh, as my roommate in Canada, thank you for helping me to get used to the life there. You are so kind! Good luck for your new study in Britain and do not stop pursuing your dreams!

Lastly, I would like to express my appreciates to all my friends and my colleagues. You helped me a lot in different aspects and there were always nice memories with you all. It's you who let me love France and Canada and encourage me to become more open to the world.

Contents

List of Figures

List of Tables

Notions & Symbols

General conventions

x and X	are two scalars
$\mathbf{x}$	is a column vector
$\mathbf{X}$	is a matrix or a multi-dimensional array

Mathematical sets

$\mathbb{R}$	set of real elements
$\mathbb{R}^D$	set of vectors with D real elements
$\mathbb{R}^{D\times M}$	set of matrices with D rows and M columns of real elements
$\mathbb{R}^{D\times M\times N}$	set of arrays with D channels, M rows and N columns of real elements
$\mathbb{R}^{B\times D\times M\times N}$	set of arrays with B batches, D channels, M rows and N columns of real elements

Vectors, Matrices and Arrays

x_d and $(\mathbf{x})_n$	n-th element of the vector $\mathbf{x}$
$\mathbf{x}_n$ or $\mathbf{X}_{:,n}$	n-th column of the matrix $\mathbf{X}$
$(\mathbf{x}_i)_j$ or $(\mathbf{X})_{i,j}$	element at the ith row and jth column of the matrix $\mathbf{X}$
$\mathbf{X}^i$	a matrix/array with the index i
$\mathbf{X}_{b,:,:,:}$	3-D array at the bth batch of the array $\mathbf{X}$
$\mathbf{X}^T$	transpose of the matrix $\mathbf{X}$

Chapter 1

Introduction

Material classification is a visual recognition task closely related to texture classification and dedicated to classify input texture/material images into categories such as fabrics, water, steel, foliage, ... (see Fig. 1.1).

As one of basic visual perceptions, studying how human vision system behaves to process real-world material has a long history since the 1960's [7]. Having an obvious difference with object recognition, material recognition's input concerns visual information coming from surfaces, instead of objects [8].

Figure 1.1 – Examples of material images and their categories. Images from the Flickr Material Database [1].

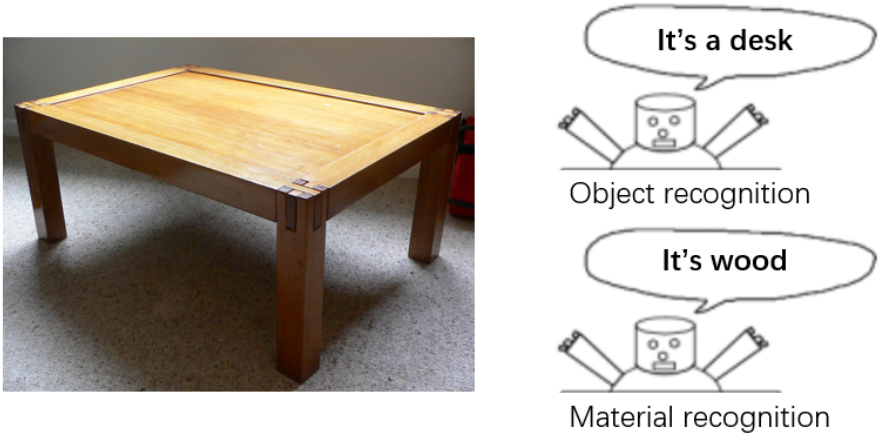


Figure 1.2 – Object and material recognition systems explain an image from different perspectives

The illustration from Fig 1.2 shows the different outputs provided by an object and a material recognition systems. We can see that, object recognition system considers the target as a 'desk' and material system detects some piece of 'wood' appearing in the image. Obviously, these two tasks are different but enrich each other since knowing the material of one surface could help to recognize the object and vice versa. As explained below, the main difference between these two tasks is that global shape and spatial organization of local features are interesting elements for object recognition while material recognition requires accurate local texture features.

Being able to recognize materials in an image is challenging but very useful for many computer vision tasks. Reading a language dictionary and a supermarket's promotion brochure, you may find many stuff words, like 'meat', 'tea', 'sky', 'soil', 'skin tissue', ... and each word may relate to one material recognition applications, like food texture classification [9], satellite or aerial imagery [10, 11], ground terrain recognition and detection [12, 13]

and medical image analysis [14, 15, 16]. Furthermore, material recognition algorithms can be implemented into robotic visual systems which allow product search, object manipulations or autonomous navigation on the surface made of specific material [13]. Also material classification is a key step of automatic waste sorting.

Material classification study starts in the 1960's and focuses on describing material with expert-defined features. Julesz introduced his pioneering work about texton theory [17, 18] where, texture or material's descriptors, called textons, are defined as elementary local conspicuous features, such as edges or corners. After that, researchers began to work on how to design efficient filter banks to extract texture features [19, 20, 21, 22, 23, 24]. Besides the study of local feature extraction, a number of approaches, like bags-of-textons [25], were proposed to aggregate local features into a global representation which can more effectively depicts material images. At the physical level, material appearances were collected under controlled conditions which means that the parameters such as lighting color, or direction and viewing direction were strictly set and recorded. With these controlled input conditions and appearances, some models, such as BRTF (Bi-directional Reflection Transmittance Function) and BTF (Bidirectional Texture Function) can be built to characterize the appearance of material instances. These models provide instance-level features which are more useful to identify material instances rather than material categories. Material appearance images, condition parameters and instance models are collected as databases [26, 3, 27]. A key characteristic for material images is that target material occupied the whole region of an image and no clutter background was involved.

In 2012, AlexNet a Convolutional Neural Network [28] (CNN) broke the image classification accuracy record in ImageNet ILSVRC [29], a very large

object recognition dataset. For material classification research, CNNs pre-trained on ImageNet replace expert-designed filters, relying on their produced high discriminative features, which were originally designed for object recognition [30, 31]. At the same time, some new material datasets were created in which images were acquired under uncontrolled conditions and target material did not necessarily fill them. Some background information is also included [1, 32, 33, 34, 12, 35]. Solving material classification on these new datasets became very challenging but new learning-based approaches already found smart and original solutions. Since these solutions are all based on deep neural networks, we have built our contributions on such architectures. The first section of this chapter introduces the general workflow of the deep neural networks and the second one presents the motivations and main ideas of our contributions.

1.1 Image classification with CNN

Image classification is one of the most fundamental research fields in the computer vision community, and its spur progress always influences greatly not only itself but also other visual recognition tasks, like video classification, image segmentation, medical image analysis. And it even has impact on other domains, such as natural language processing or brain-computer interface. Convolutional Neural Networks (CNN) represent a breakthrough in computer vision, since AlexNet [28] clearly outperformed the state-of-the-art in ImageNet ILSVRC competition [29]. This achievement is considered as one of the milestones both for deep learning and computer vision.

This thesis mainly deals with research fields based on CNN networks for material recognition, so a brief presentation about image classification based

on a CNN network seems necessary. We propose to present below, the basic feedforward propagation, the main modules of a classical architectures and the backward propagation.

1.1.1 Feedforward propagation

For a multi-class image classification task over K classes, the goal is to correctly classify an image $\mathbf{I}$ into its ground-truth category $y \in \mathcal{Y} = \{1, ..., K\}$. Feedforward propagation of a CNN can be abstracted as a function f that projects an image into a prediction vector $\hat{\mathbf{z}} \in \mathbb{R}^K$:

$$\hat{\mathbf{z}} = f(\mathbf{I}). \tag{1.1}$$

Then, this prediction $\hat{\mathbf{z}}$ is transformed into a probability vector $\hat{\mathbf{p}}$ with a $softmax$ function:

$$\hat{\mathbf{p}} = softmax(\hat{\mathbf{z}}). \tag{1.2}$$

Precisely, each element $\hat{p}_k$ of $\hat{\mathbf{p}}$ is evaluated as:

$$\hat{p}_k = \frac{\exp(\hat{z}_k)}{\sum_i^K \exp(\hat{z}_i)}, \tag{1.3}$$

where $\hat{p}_k$ represents the probability that $\mathbf{I}$ belongs to the kth class. We notice that each value in $\hat{\mathbf{p}}$ can not exceed 1 and that all the values sum to 1 ($\sum_{k=1}^K \hat{p}_k = 1$), thus assimilating this vector to a probability distribution.

And the index of the element with the highest probability is picked as the predicted category $\hat{y}$:

$$\hat{y} = \underset{k \in \mathcal{Y}}{\operatorname{argmax}}\, \hat{p}_k. \tag{1.4}$$

In the case of a correct prediction, the predicted category is equal to the groud-truth class:

$$\hat{y} = y. \tag{1.5}$$

1.1.2 Deep learning network structure

In the previous section, a deep CNN is defined as a function f. Now we look into its structure in detail. Note that deep CNN architecture engineering is still under study and there exist many different architectures. Here, we introduce a general structure which refers to AlexNet [28] or VGG [36] networks. These networks were also adopted in our experiments, as detailed in the following chapters.

If we look into the function f of a CNN, its structure is a stack of layers. According to their properties, we regrouped these layers into three sequential components: convolution, pooling and classification, as shown in Fig 1.3, and respectively viewed as functions: $f_{conv}(.)$, $f_{pool}(.)$ and $f_{fc}(.)$.

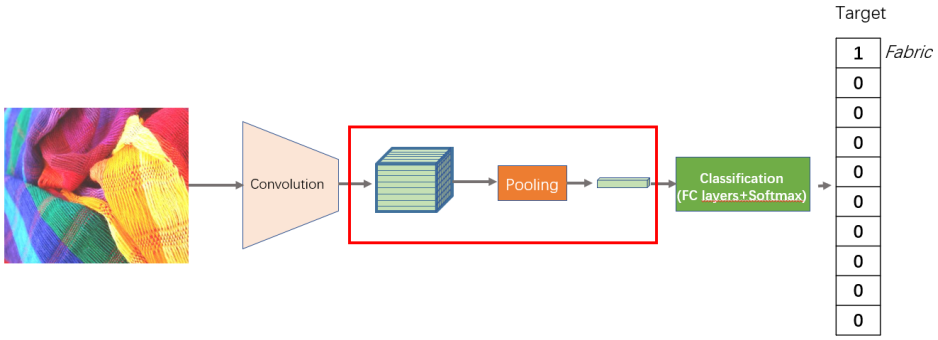


Figure 1.3 – A classical architecture of a CNN.

Suppose that the input of a deep CNN is an image $\mathbf{I}$. The convolution component's job is to extract local features from $\mathbf{I}$:

$$\mathbf{X} = f_{conv}(\mathbf{I}) \tag{1.6}$$

The output $\mathbf{X} \in \mathbb{R}^{C \times W \times H}$ is a 3D tensor containing a set of local feature vectors $\mathbf{X}_{:,w,h}$ at 2-D spatial position: $(w \in \{1, ..., W\}, h \in \{1, ..., H\})$, where $\mathbf{X}_{:,w,h} \in \mathbb{R}^C$ and C, W, H are respectively the number of channels, the width

and the height. In the convolution component, the extraction is realised with repeated convolution layers plus local pooling layers in a hierarchical way. This means that the first layers extract low-level primitive features, such as edges or colors while the last layers combine these low-level features into high-level semantic features, such as hands, wheels or trees.

Next, located inside the red bounding box of Fig 1.3, the global pooling component aggregates the local features from X into one global and compact feature vector:

$$\mathbf{a} = f_{pool}(\mathbf{X}) \tag{1.7}$$

where $\mathbf{a} \in \mathbb{R}^C$. For example, after extracting features of a nose, a mouth and eyes with convolution component, features after pooling component is able to represent a face.

Lastly, fully connected layers and a $softmax$ function (see Eq. 1.2) constitute the classification component which provides the predicted probabilities for the considered categories:

$$\hat{\mathbf{p}} = softmax(f_{fc}(\mathbf{a})) \tag{1.8}$$

For more details about these three components, please see Appendix A.

1.1.3 Backward propagation

To consistently make correct predictions, a training process, described in Algorithm 1, is needed to learn CNN's ensemble of every lth layer's parameters $\left\{\boldsymbol{\theta}^l | l = 1, ..., L\right\}$ with a training dataset $\mathcal{D}$, consisting of N samples defined by a pair of values corresponding to the image and its ground-truth category $\mathcal{D} = \left\{(\mathbf{I}^1, y^1), ..., (\mathbf{I}^N, y^N)\right\}$.

Concretely, at each iteration of the training process, a mini batch $\mathcal{D}_B$ with B samples is randomly drawn from $\mathcal{D}$ and we optimize an objective

Algorithm 1: Network training

Input : Training dataset $\mathcal{D}$, CNN function f with parameters $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^L\}$

Output : CNN function f with parameters $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^L\}$

for $\mathcal{D}_B \in \mathcal{D}$ **do**

$\mathcal{D}_B = \{(\mathbf{I}^1, y^1), ..., (\mathbf{I}^B, y^B)\}$

% Step 1: Feedforward Propagation

Eq. 1.1 and Eq. 1.2–>$\{\hat{\mathbf{p}}^1, ..., \hat{\mathbf{p}}^B\}$

% Step 2: Backward Propagation

Eq. 1.11–>$\{\frac{\partial \mathrm{L_{ce}}}{\partial \boldsymbol{\theta}^1}, ..., \frac{\partial \mathrm{L_{ce}}}{\partial \boldsymbol{\theta}^L}\}$

% Step 3: Parameter update

Eq. 1.10–>$\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^L\}$

end

function, namely the cross entropy loss:

$$\mathrm{L_{ce}} = -\frac{1}{B}\sum_{n=1}^{B}\log(\hat{p}^n_{k=y^n}), \tag{1.9}$$

where $\hat{p}_k$ is obtained with Eq. 1.3, with performing Stochastic Gradient Descent (SGD) or its variants to update every ith layer's parameters $\boldsymbol{\theta}^l$:

$$\boldsymbol{\theta}^l = \boldsymbol{\theta}^l - \frac{\partial \mathrm{L_{ce}}}{\partial \boldsymbol{\theta}^l} lr \tag{1.10}$$

where the scalar variable lr represents the learning rate, determining how 'aggressively' we update the parameter values with their gradients $\frac{\partial \mathrm{L_{ce}}}{\partial \boldsymbol{\theta}^l}$.

Practically, the chain rule is used to calculate the partial derivative $\frac{\partial \mathrm{L_{ce}}}{\partial \boldsymbol{\theta}^l}$:

$$\frac{\partial \mathrm{L_{ce}}}{\partial \boldsymbol{\theta}^l} = \frac{\partial \mathrm{L_{ce}}}{\partial \hat{\mathbf{p}}}\frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{X}^{L+1}}\frac{\partial \mathbf{X}^{L+1}}{\partial \mathbf{X}^{L}}...\frac{\partial \mathbf{X}^{l+2}}{\partial \mathbf{X}^{l+1}}\frac{\partial \mathbf{X}^{l+1}}{\partial \boldsymbol{\theta}^l} \tag{1.11}$$

where $\mathbf{X}^{l+1}$ is the output of lth layer. The gradients in the chain rule, i.e. $\frac{\partial \mathrm{L_{ce}}}{\partial \hat{\mathbf{p}}}$,

$\frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{X}^{L+1}}$, $\frac{\partial \mathbf{X}^{l+1}}{\partial \mathbf{X}^{l}}$ and $\frac{\partial \mathbf{X}^{l+1}}{\partial \boldsymbol{\theta}^{l}}$, are accessible because the corresponding functions in each layer, such as Eq. 1.9, Eq. 1.3, are almost differentiable[1].

As observed in Eq. 1.11, the order (from output to input) of the gradient calculation is in the inverse direction of Feedforward Propagation (FP), thus gradient calculation process is called Backward Propagation (BP).

In the following chapters 3 and 4, we mainly use three popular CNNs: ResNet-50 [37], VGG-16 [36] and AlexNet [28]. Although there exist some differences among them, e.g., skip connection of ResNet-50 (see Fig. A.1), their respective structure and training process basically conforms to the description in this subsection.

We have already underlined that the main goal of material classification systems is to extract a global feature vector that accurately represents the most relevant local features without paying attention to their spatial distribution. From the previous general presentation, it is clear that the main step on which we should concentrate is the pooling component: $f_{pool}(.)$, whose aim is to evaluate a global feature vector from a set of local features (see Eq. 1.7). In this thesis, we are presenting two main contributions around the global pooling layer, as introduced in the next section and detailed in the chapters 3 and 4.

1.2 Contributions

The goal of this thesis is to **improve material classification performance based on CNNs and real-world datasets**. To meet this goal, we concentrate on two key steps of the classification framework that are parts of the

[1]In practice, some exceptions, like ReLU function, do not affect network training

global pooling module, namely local feature selection and orderless pooling. The motivations and main ideas of the proposed solutions are introduced below and detailed in Chapters 3 and 4, respectively.

1.2.1 Feature selection

Today, as mentioned in the previous section, in most of the CNN architectures, after stacked convolutional layers extracting local features from the input image, a Global Average Pooling (GAP) layer is considered as $f_{pooling}(.)$ in Eq.1.7 and it merges all the local features into a single global feature vector [37]. Then, a fully connected layer predicts the image class based on this global feature vector. With this classical approach, each local feature vector equally contributes to the final decision through the averaging operation. However, when large areas of the images are ambiguous or when useful information is mainly provided by fine image details in some tiny areas, averaging all the local features could be sub-optimal. And we will show that this is all the more true in the images of materials. An illustrative example is shown in Fig.1.4: On the left column, some small but informative parts of the images are masked and it makes the class prediction very difficult with the remaining large and ambiguous areas. Once one has access to these details (right column), class prediction becomes much easier.

However, how to choose key discriminative areas and eliminate ambiguous features is not trivial and this is the question we address in Chapter 3.

At the first glance, we have run a naive experiments in order to measure the impact of the context of the considered objects on the classification performances. To do so, we used a benchmark material dataset, called Flickr Material Database (FMD) [1] which provides binary masks covering material region, as shown in Fig. 1.5. This dataset has 10 different categories.

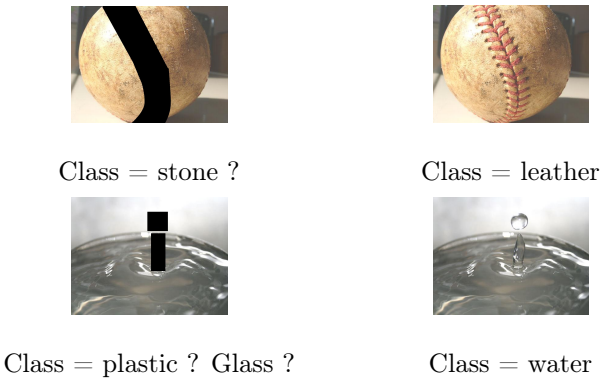


Figure 1.4 – Images from the Flickr Material Dataset [1], showing that, sometimes, some details are essential to predict the correct class while large areas are ambiguous.

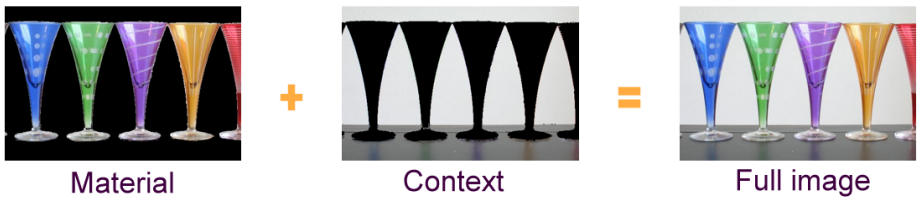


Figure 1.5 – With the provided masks, we are able to create three images from each sample image: the material image (without backgound), the context image (only background) and the full image.

Thus, with these data, we were able to conduct different classification experiments by changing the inputs of the networks:

- Training and testing only on the context images,
- Training and testing only on the material images,
- Training and testing on the full images.

The resulted accuracies averaged over 5 runs are provided in Table 1.1. There, we can see that i) the context provides few interesting features to clas-

Table 1.1 – Background context's impact on material classification accuracy.

	Average accuracy over 5 runs
Train/Test on the context only	27.1
Train/Test on the material only	68.0
Train/Test on the full images	71.4

sify the images (27.1% accuracy for 10 categories), ii) the material provides much better features than the context (68.0% accuracy) and iii) the context and the material sometimes provides complimentary information since working with the full images (71.4% accuracy) improves over the tests with the material only (68.0%). So it can be concluded that the impact of context to material classification is positive and it is not appropriate to simply select features only from material areas by eliminating image's context.

In order to automatically select the most informative local feature vectors, we will propose in Chapter 3 to use confidence scores that represents the usefulness of local feature vectors on each image area. By exploiting a very recent and successful approach, designed for global failure prediction [6], we propose to predict the local feature confidence with an additional branch in the network. Only the local feature vectors with higher confidence predictions are preserved and averaged into a global feature vector. In Chapter 3 we will provide both quantitative and qualitative results on three material datasets and will demonstrate that our method not only augments classification accuracy but also improves the calibration of the output probabilities.

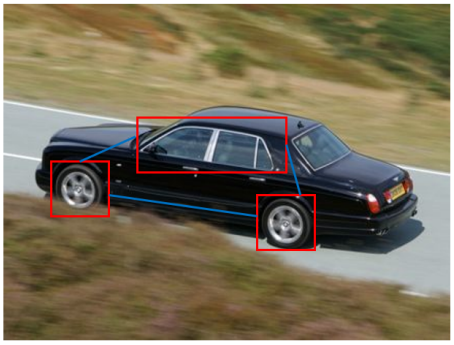



Figure 1.6 – Left: an object image from Stanford Cars Dataset [2]. The car's parts, marked by red bounding boxes, have stable spatial relationship (marked by blue lines). Right: a material image from Flickr Material Dataset [1]. No specific spatial arrangement is presented among different instances.

1.2.2 Orderless pooling

As discussed at the beginning of this chapter, compared to object recognition tasks, material recognition has its own properties. One of these properties, is its spatial orderless arrangement. As illustrated in Fig 1.6, as an object, the discriminative parts of a car, like wheels, windows, have strong spatial and topological relationships. For example, wheels are arranged on the same horizontal level and windows are beyond the wheels. For material images, such predefined relationships between surface areas do not exist and should not be accounted for in the final global feature vector.

Orderless pooling emerges recently and becomes attractive for material classification tasks because it aggregates material features without taking into account their spatial locations in the image. For example, one of the simplest orderless pooling methods is the global average pooling which averages over all the local features. It is widely used in the recent deep neural network

architectures such as the ResNets [37].

Actually, before CNNs, orderless pooling was already well developed in the bags of visual words (BOW) [38], the VLAD [39] or the Fisher Vectors [40], and they have provided good results when applied to CNN features for texture or image classification [31, 41]. Among them, the Fisher Vectors (FV) are the most promising because it generalizes the VLAD and BOW and uniquely appends second order statistics. Later, few works embedded aforementioned orderless poolings as a learnable component into a CNN [42, 43, 44]. This implementation allowed an end-to-end training in which all the parameters in the CNN, including those of the orderless pooling component, can be learned under the supervision of the target task.

Nevertheless, embedding orderless pooling in a deep architecture has two main disadvantages from our point of view. First, since the deep features lie in high dimensional spaces, specific tools are required to accurately model their distributions. For example, Liu et al. show that a classical Gaussian Mixture Model requires too many gaussian centers to accurately model high-dimensional deep features [45]. Hence, they propose a sparse coding solution that is not embedded in a deep architecture. Taking inspiration from this approach, we have proposed to insert a trainable module in a deep neural network that is able to sample gaussian centers from a subspace in order to accurately model the deep features.

Second, dealing with second order statistics such as those provided by the Fisher Vectors is not easy and requires successive normalization steps [46]. Nevertheless, the solutions suggested in [46] require Symmetric Positive Definite (SPD) matrices as input, which is not the case of our Fisher representation. Consequently, we have proposed a new and original approach to normalize any matrix (not squared or symmetric) that represents second or-

der statistics. To the best of our knowledge, it is the first time that matrix normalization applied to Fisher-based representation.

These two main contributions are embedded in a deep architecture so that the final network is trainable end-to-end with the single classification loss.

1.3 Organization of the thesis

The remainder of the thesis is organized as follows. First, we describe related works in Chapter 2, including classical material classification solutions and CNN-based approaches. We also decide to concentrate on two main points, confidence prediction and orderless pooling that are important to understand our contributions. Some parts of this chapter refer to our survey paper published in *International Conference on Big Data, Machine Learning and Applications (BIGDML) conference* in 2019.

Chapter 3 presents our first main contribution, briefly introduced in the section 1.2.1. Some parts of this chapter refer to our conference paper published in *International Conference on Image and Vision Computing, New Zealand (IVCNZ)* in 2020.

Our second main contribution, briefly introduced in the section 1.2.2, is precisely discussed in Chapter 4. Some parts of this chapter, related to sparse coding, refer to our conference paper published in *International Conference on Computer Analysis of Images and Patterns (CAIP)* in 2021. An extended version of this paper with normalization has been submitted in July 2021 for publication to *Computer Vision and Image Understanding*.

Lastly, Chapter 5 draws conclusions, reveals actual challenges and trends in the material classification, and suggests perspectives for future works.

Chapter 2

Related works and datasets

In this chapter, we explore and present the most remarkable works in the context of material classification, starting from hand-crafted features to deep-CNN solutions. This analysis reveals that orderless pooling and end-to-end learning are two essential elements for material classification. Also, it allows to discover the main limitations of the current state-of-the-art approaches. These remarks will be the starting points of our original solutions detailed in the next chapters. In this chapter, we also present, the numerous image material datasets. Some of them will be used in the experimental tests of the next chapters.

2.1 Handcrafted features

In the 60's, the earliest work about material analysis reveals that material or texture can be perceived spontaneously if proximate pixels of uniform brightness form a specific connectivity [7]. In the 80's, to further explain human perception of material, Julesz introduced texton theory [17, 18]. He argued

that textures can be perceived if elementary local conspicuous features, called textons, are present, such as crossing, corners, etc. He also stated that only first-order statistics of these textons are meaningful. In other words, spontaneous perception cannot be triggered if the probability of every texton in one material region is equal.

As for local conspicuous features extraction, expert-designed filter banks, like Gabor filters [19, 20, 21, 22], Gabor wavelets [23], Differences of Gaussians [24], serving as sliding windows, can produce local features from the input material image. Based on texton theory, a series of works [47, 48, 49, 50, 51] tried to mathematically model textons and consequently, Bags-of-words [52] and Bags-of-textons [25] were proposed to aggregate features into a histogram representation over a given texton dictionary. By the end of the last century, researchers concentrate on the extraction of invariant feature representation. Some types of features are more robust than others to certain variations, such as background illumination or object size. The most notable invariant features include Local Binary Pattern (LBP) [53], Speeded Up Robust Feature (SURF) [54] and Scale Invariant Feature Transform (SIFT) [55]. Besides material classification, they dominated visual recognition field before the deep learning era.

2.2 Deep learning features

Impressed by the outstanding results provided by the deep neural networks, many research teams began to use CNNs pretrained on ImageNet [29] for their own studies. Indeed, it appeared that the knowledge learned from a large image dataset for classification tasks, can be helpful for other datasets and tasks. This is called 'transfer learning' and has been widely used in the

context of material classification.

In a first study, Wieschollek and Lensch simply collects material features extracted by a network pretrained on ImageNet and trains a classifier with these features [30]. This simple method outperforms alternatives based on handcrafted features with an evident margin, showing that generic deep features are transferable to material classification.

2.3 Orderless pooling

Using a pretrained CNN allows to extract a group of local features from material images. These local features are aggregated to a global feature vector thanks to a pooling module, as explained in A.2. This global representation of the image can be used as input for the following classifier. As mentioned earlier in this thesis, the local features of the material images do not have specific spatial arrangement that could help for the classification task. Thus, in this context, orderless pooling is preferred for these images, because it combines local features while omitting their spatial position in the image, as shown in Fig 2.1. The representation given by an orderless pooling can be more relevant to material and thus improve the classification performance.

The second useful property of orderless pooling is its ability to handle feature vector sets of undefined size [31], which means that we can feed the network with images of more flexible sizes. For classical networks such as AlexNet [28] or VGG-16 [36], the extracted feature map is directly sent to a Fully Connected (FC) layer with a predefined neuron number, which means that the input image should have a predefined and fixed size. This is particularly inappropriate when the network works with features over many different image areas [56, 57].

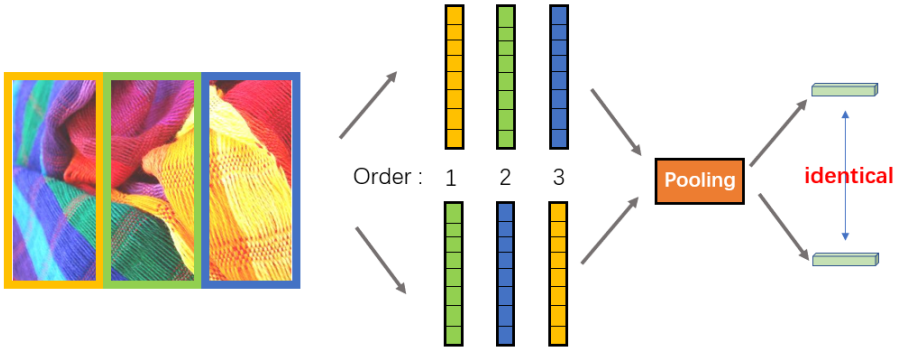


Figure 2.1 – Illustration of the orderless pooling. Best viewed in color. Suppose there are three feature vectors (shown here with blue, green and yellow false colors) extracted from three image areas (marked by bounding box of the same color). If these feature vectors are input into orderless pooling in different orders, then pooled representations are identical.

The third advantage of this orderless pooling is that it allows to decrease the number of trainable parameters by a large margin. Indeed, since the neurons of the first FC layer are connected to all the input cells of the feature map, reducing the dimensions (by skipping the spatial dimensions) of this feature map, reduces the number of inputs a lot, and thus, the number of weights to learn. This is a good way to reduce overfitting [37].

2.3.1 Basic deep modules

The simplest orderless pooling methods are global average pooling and global max pooling, where local features $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N | \mathbf{x}_i \in \mathbb{R}^D\}$ per feature channel j are pooled with the average operation:

$$a_j = \frac{1}{N}\left(\sum_{i=1}^{N} x_{i,j}\right), \tag{2.1}$$

or by the max operation:

$$a_j = \max_{i=1,\dots,N} x_{i,j}, \tag{2.2}$$

where a_j is the jth element of the pooled representation: $\mathbf{a} \in \mathbb{R}^D$

In particular, the series of ResNet [37] use a Global Average Pooling (GAP) layer that merges all the local features into a single global feature vector before the FC classifier.

2.3.2 Non-embedded methods

Let first introduce a group of orderless pooling methods, which are not embedded in a neural network, which means that they are trained separately from the used classifier. In other words, these methods can not be used in an end-to-end trainable network. So, these pooling methods are trained with the deep features provided by a pretrained network. Then their outputs are fed to a classifier that is trained in a later step.

For ones who are not very familiar with orderless pooling methods, such as Bag of Words(BOW), Fisher Vector(FV) and Vector of Locally Aggregated Descriptors(VLAD), an introduction in more details can be found in section 4.3.2.

Bags-of-Textons.

After the texton theory introduced in the works [17, 18], Leung and Malik define texton in a more practical way [25] as a cluster center in the filter response space. One of the consequences of this definition is that it allows to learn a universal texton dictionary for a set of images. Then, the pooled representation of an image is a histogram over this texton dictionary. This

pooling technique is referred as Bag-of-Textons (BoT) [25, 24]. It is orderless because the histogram counts feature vectors closed to every textons in the dictionary. BoT was later generalized to more visual recognition tasks, namely Bag-of-Features (BoF) or Bag-of-words (BoW) [52].

Fisher Vector based methods.

One of the drawbacks of BoW is that it only counts the occurrences of the visual words in one image. This 0^{th} order statistics representation ignores the distribution of the local features around their cluster center. Fisher Vector methods extend BoW and produce representation of higher orders. The classical Fisher Vector [40] method models the distribution of the training data with a Gaussian Mixture Model (GMM). Then, it characterizes each data point with the derivatives over the model parameters, i.e, means, weights and covariances of each Gaussian component in the GMM. This coding approach is referred below as Gaussian Mixture Model based Fisher Vector (GMMFVC).

As deeply discussed in this Thesis (see Chapter 4), it was found that the normalization plays an important role in fixing the 'burstiness' issue of Fisher vector, where discriminative but relatively rare visual features are overwhelmed by those frequently appearing [58]. Thus, Perronin et al. proposed element-wise signed square rooting and L2-normalization to cope with this problem [58]. Although there exist other alternatives [59, 60] to improve GMMFVC (which show comparable or even better performances in certain cases), the GMMFVC with such normalization outperforms the BoW in many applications, including material recognition [31, 61].

Cimpoi et al. used pretrained CNN to extract features on an image at multiple scales and aggregated them to GMMFVC [31]. This direct im-

plementation of GMMFVC produced state-of-art results on several material classification datasets. Song et al. further improved the work of [31] by creating a Locally-transferred Fisher Vector (LFV) module, which transforms the Fisher vector into a more discriminative representation [61].

A complete Fisher vector is composed of derivatives over every parameter of the GMM model. Its size is sometimes too large and this inconvenience was firstly found in the image search on a very large scale dataset where features have to be saved. To obtain more compact representation, Jegou et al. proposed the Vector of Locally aggregated descriptors (VLAD) [62]. It aggregates the residuals between the local features and their nearest visual words (1st order statistic). That algorithm can be seen as a simplified calculation of the derivatives over the means of GMM. Although it is simple, it's great performance is validated [41, 63].

The main disadvantage of Fisher Vector or VLAD is that they rely on a limited number of codewords or Gaussian centers, which prevents accurate modeling of the data distribution in high-dimensional deep feature spaces. Liu et al. proposed a smart solution to overcome this problem which consists in sampling the center of each Gaussian from a subspace [45]. It therefore benefits from an infinite number of Gaussians to fit the data distribution. The authors showed that this problem can be solved by a classical sparse coding method, which is a regression with L1 norm regularization (called LASSO regression). Another interesting solution to cope with this problem was proposed by Dixit et al who computed Fisher vectors from a Mixture of Factor Analyzers (MFA) instead of the classical GMM [64]. The idea of MFA is to approximate the data manifold by low dimensional linear spaces and, in this sense, is similar to the idea of sparse coding [45]. As shown in Chapter 4, these solutions are not adapted for end-to-end material classification and one

of our contribution is to deal with the high dimension of deep features with an end-to-end pooling module (presented in Chapter 4).

2.3.3 Finetunable methods

End-to-end learning.

Finetunable pooling methods enable an end-to-end learning of the network. As mentioned in section 1.1.3, during the training phase of a CNN, loss function's gradients are backpropagated from the last layer $l = L$ to the first one $l = 1$, with respect to an ensemble of every layer's parameters $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^L\}$ where L is the number of layers in the CNN. Consequently, all the parameters can be updated with their own gradients. This learning process is called 'end-to-end', since parameters from one end (output) to the other end (input) are learned jointly under the same supervision of the current classification task.

The advantages of end-to-end learning are obvious: First, instead of independently training every module one by one, the training phase is unified into only one step. Thanks to this simplification, the features collection step is thus cancelled. The large memory, which was necessary to save the features, is not needed anymore. Second, though it is proven that features made by pretrained CNNs are transferable into material recognition tasks, an end-to-end learning can further improve the performance of the classifier by finetuning the network and fitting the features for the target task. This end-to-end finetuning contributes to further reduce the loss of the objective function.

In this section, finetunable pooling methods are regrouped into three families of methods. Several state-of-the art approaches will be presented for each family.

Fisher Vector and BOW methods

Motivated by the advantages of BoW or Fisher Vectors, some researchers proposed end-to-end trainable versions [65, 42, 43, 44, 63, 66, 67]. Passalis and Tefas inserted a Bag-of-Features pooling in deep neural networks thanks to radial basis function neurons [65]. The output of the pooling module is a histogram of the visual words (0^{th} order statistic) learned from the training set. NetVLAD was the first network that transforms VLAD into a deep module which allows an end-to-end training [42]. It was later improved by Zhang et al. with Deep Ten [43]. It was shown that first order statistics are more accurate to characterize images in classification tasks and the Fisher vectors go further by using first and second order statistics. Deep Fisher-Net is an embedded implementation of the GMM Fisher vector [44]. Lin et al. introduced NetFV which extends NetVLAD by appending second order statistics [63]. Li et al. embedded the MFA fisher vector [64] in a deep network which is end-to-end trainable [66]. In a recent study, Brendel and Bethge also proposed to aggregate the class activation of each local patch in the image in a global feature vector [67].

Bilinear Pooling

Bilinear model was firstly introduced by Tenenbaum et al in [68] to separate style and contents. Lin et al. [69] extended it with a pooling layer plugged at the end of the last convolutional layer. The pooling layer aggregates feature vectors $\mathbf{X} = \left\{\mathbf{x}_1, ..., \mathbf{x}_N | \mathbf{x}_n \in \mathbb{R}^D\right\}$ of the last convolutional layer by average-pooling their outer products:

$$\mathbf{A} = \frac{1}{N}\left(\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T\right) \tag{2.3}$$

where the feature map $\mathbf{X}$ contains n feature vectors of D dimensions. As an orderless pooling method, bilinear pooling pools input feature vectors into a fixed-size output $\mathbf{A} \in \mathbb{R}^{D \times D}$. Furthermore, opposite to Fisher Vector based methods which need an extra dictionary, bilinear pooling can be non-parametric. Moreover, it extracts also second order statistics, as Fisher vectors do. This second aggregation method, though simpler than Fisher Vector based methods, shown its efficiency for multiple processing tasks, like texture synthesis [70], style transfer [71], segmentation [72] or visual question answering [73, 74]. In particular, according to results provided by the work [63], bilinear pooling shows superior performance on several classification tasks, like fine-grained classification, indoor classification and material classification.

Following the pioneer work [69], other improvements were proposed to improve bilinear pooling in different aspects. Wang et al. [75] proposed G^2DeNet containing a Gaussian embedding, which combines bilinear pooling information (Gaussian's covariance) and first order information (Gaussian's center). This fusion between different order information enables to achieve better performance than original bilinear pooling. Kernel Pooling (KP) [76] extends bilinear pooling to higher order pooling and concatenates weighted representations of the first four orders into a more relevant global representation.

Another improvement track tries to find compact bilinear pooling solution. In the work [69], since the feature vector dimension D is equal to 512, the number of elements in the output matrix $\mathbf{A} \in \mathbb{R}^{D \times D}$ is more than 250.000. Such a cumbersome representation is not practical in many aspects. One of them is leading to a heavy classifier, containing numerous weights and bias. Moreover, in the first place, feature vector's dimension D becomes bigger

in more recent CNN architectures and this is a well-known tendency in the deep learning's network engineering field. As an example, D can reach 2048 for ResNet50 [37]. Dimension reduction for bilinear representation should be consequently considered. Gao et al. applied existing kernel approximations such as Random Maclaurin and Tensor Sketch, to produce approximated bilinear representations in lower dimension [77]. The authors show that the dimension can be reduced about 32 times while keeping almost identical performance as the original. Kong and Fowlkes considered a low-rank bilinear SVM to run classification based on bilinear representation [78]. The key advantage of this approach is that it avoids to explicitly compute the bilinear representation $\mathbf{A}$ (defined in the previous equation). Yu and Salzmann proposed SMSO that uses a 1×1 convolution layer and a global ℓ_2 pooling operation to obtain an approximated compact bilinear representation [79]. It leads output bilinear representation to be Gaussian-distributed which were shown to be favorable for better classification accuracy.

As the output of the bilinear pooling, bilinear matrix A is Symmetric Positive Definite (SPD) it lies in the Riemannian manifold and training a linear classifier in such manifold [69] is clearly sub-optimal. Indeed, the linear classifiers work more efficiently for features lying in the Euclidean space. In order to map the SPD matrix manifold into a Euclidean space, multiple works suggested using matrix-logarithm [80, 72, 81]. They reported improvement with linear classifiers for semantic segmentation and image classification. The logarithm scales the eigenvalues of the Singular Value Decomposition (SVD) of a SPD matrix $\mathbf{A}$ by $\log(\mathbf{A}) = \mathbf{U}\log(\mathbf{\Sigma})\mathbf{U}^T$. This normalization involves a SVD explicit computation which runs inefficiently on GPUs [46]. If we plug the normalization into the deep learning network, it slows down the inference speed of the network. Lin and Maji also proved that the alternative approach,

matrix square-root of a SPD $\mathbf{A}$: $\mathbf{A}^{1/2} = \mathbf{U}\mathbf{\Sigma}^{1/2}\mathbf{U}^T$ has similar performance and that $\mathbf{A}^{1/2}$ can be approximated by a variant of Newton iterations which speeds up this normalization on GPUs [46]. Hence, after this high-speed matrix square-root normalization is integrated into the network, the network can be end-to-end trained easily and benefits from normalized features as well.

iSQRT [82] further introduces Newton iterations into the backward propagation of the network so it speeds up the training process with normalized bilinear pooling. After coping with the slow training problem, Li et al. [83] applied iSQRT to deeper CNN, such as ResNet-101 [37] and DenseNet-201 [84], and trained the networks from scratch on ImageNet and Place365 large-scale dataset providing an obvious improvement over original networks. Hence, the second order information is no longer limited to small-scale classification tasks and can also help for general large-scale visual recognition.

In summary, there are two tracks to improve original bilinear pooling. One track is to obtain a more Compact Bilinear Pooling (CBP) and the other one is bilinear matrix normalization. More recently, exploring how to conceive an algorithm which incorporates these two techniques has been explored. However, Lin et al. concluded that the matrix normalization cannot be easily computed in the space of CBP made by kernel approximation approaches, such as Tensor Sketch [85]. In other words, directly running some matrix normalization on the CBP features is rather difficult. So the authors proposed to use the γ democratic algorithm which equalizes the contribution of each outer product $\mathbf{x}_n\mathbf{x}_n^T$ in the final representation $\mathbf{A}$ with a scalar weight. They argued that this algorithm has a similar effect as matrix normalization while working with CBP using Tensor Sketch. Gou et al. proposed an alternative where the local feature map $\mathbf{X}$ is transformed into

another map $\mathbf{Y}$ on which a bilinear pooling is applied so that this output is the normalized bilinear representation of $\mathbf{X}$ [86]. $\mathbf{Y}$ has the same size as $\mathbf{X}$ and preserves the original matrix structure. Hence, compact bilinear pooling, such as tensor sketch, can directly work with it. One drawback of this approach is that a SVD computation is still involved in the transformation, with the known drawbacks of SVD computation on GPUs. To solve the SVD issue, Yu et al. normalized $\mathbf{X}$ into $\mathbf{Y}$ with their faster and simpler RUN algorithm, which, however, only normalizes the maximum eigenvalue [87]. They later improved the CBP's Random Maclaurin (RM) with a Shifted Random Maclaurin (SRM) [88]. SRM needs smaller binary $(+1, -1)$ projection matrices and performs CBP faster.

Compared to the BoW, Fisher vectors or VLAD representations, the bilinear pooling does not fit any distribution on the training data and maybe ignore some relevant features from the training set, as shown in Chapter 4. However, since it extracts second order statistics, we can exploit some interesting results provided by these approaches to improve our Fisher vector features (see Chapter 4).

2.3.4 Multi-level outputs

All the methods described above extract features from the last convolutional layer. However, as the input image goes through the network layer by layer, features extracted from different levels also contain complementary and rich information. For material classification, both primitive information (like texture on a piece of material) and semantic information (like the object category) could be combined together in a more discriminative representation, as illustrated in Fig 2.2. In the following, we present some representative works which leverage features from different layers.

Figure 2.2 – If we just crop a local patch (surrounded in blue) from a coffee cup, the patch appearance suggests a smooth white material, so maybe it belongs to the "ceramic" class or to the "paper" class. But, if we scale up the view until we see the surrounding context, anyone will be pretty sure that the patch on the left image corresponds to a "ceramic", due to the shape and the reflectance properties of the object to which it belongs, meanwhile for the patch on the right image we could state that this patch belongs to the "paper" category.

Cimpoi et al. not only proposed to pool local features of the last convolutional layer of VGG-19 into orderless representation, but also found that after combining orderless representation with penultimate Fully Connected layer's output, there is an obvious increase of the classification accuracy [31]. The authors explained that the FC layer can be considered as a pooling method which is not orderless and which is able to capture the overall shape of the object present in the image. Shape features can be seen as complementary features to the orderless representation.

Andrearczyk and Whelan designed a network architecture called Texture CNN (T-CNN) where features from different layers are respectively average-

pooled into a compact feature vector and then all the compact vectors are concatenated into a global one [89].

Inspired by the findings in the work [31], Xue et al. extended the Deep Texture module [43] to the Deep Encoding Pooling Network (DEP) that feeds the output of the last convolutional layer of ResNet into two branches: Deep Texture module and global average pooling layer [13]. The outputs from the two branches are then fused with a bilinear operation. Hu et al. encapsulated the two-branch structure of the work [13] into a Learnable Encoding Module (LEM) and plugged it to the end of basic blocks in the ResNet-50 in order to encode multi-level texture representations [90].

In the bilinear pooling community, Dai et al. combined first-order features computed by average pooling and second order features computed by CBP with a simple concatenation [91]. They also tried to fuse multi-level features to get a better performance. Herarchical Bilinear Pooling [92] runs bilinear pooling on local features across different layers and thus enhances bilinear representation by capturing inter-part feature relations.

Ghose et al. explicitly modeled the extent-of-texture (EOT) and extent-of-shape (EOS) on a local group of feature vectors [93]. According to the EOT (resp. EOS), feature vectors are split into two groups and are encoded separately into a global representation for each group. In the end, with the guide of EOT (resp. EOS), two global representations are combined and finally aggregated into an image-wise representation with bilinear pooling.

Unlike these previous methods which concatenate pooled features from several layers, Zhai et al. propose to concatenate multi-layer feature maps [94, 95]. Then, in the work [94], they applied a module with a cascade structure, in which global image representation at actual level should guide the next level representation. At the end, a fusion module is introduced to jointly

exploit each level's global representation and to make strong classification prediction. In the work [95], they designed a different encoding module that, first generates multiple texture primitives and then encodes texture primitives at one position by its correlations to other local neighbors. Note that, at the end, the output is an orderless pooled representation and it is finally integrated with spatial ordered information.

2.4 Conclusions about the related works

All these works around material classification clearly show relevant trends while revealing clear weaknesses.

Indeed, the most accurate approaches are based on an end-to-end learning process that allows to make the different modules of the deep architecture cooperating towards a single goal, which is to minimize the current classification loss.

Furthermore, the pooling module is a key element of the workflow and orderless pooling is currently the main solution proposed in the state-of-the-art approaches. However, this module ignores that some parts of material images can be more relevant than others and it seems interesting to be able to weight the contribution of each local feature vector before pooling it in the global feature representation. This is the main idea proposed in Chapter 3 of this Thesis.

Also, fitting the training data with a parametric distribution seems to be adapted to our task and the Fisher Vectors clearly outperform the alternatives among the orderless pooling. However, since the deep features lie in a very high dimensional space, solutions have to be proposed to accurately model the data distribution. Sparse coding appears as a smart and accu-

rate solution [45], but should be designed to be embedded in an end-to-end trainable architecture. This is one contribution proposed in Chapter 4 of this Thesis.

Furthermore, we have seen that normalizing such matrices of second order statistics is crucial before the classification step. Especially, many smart solutions have been proposed for the output of the bilinear pooling. However, since the output matrices of our Fisher vector pooling are not PSD, these solutions have to be adapted. In Chapter 4 of this Thesis, we propose an accurate and fast normalization for the Fisher matrix.

Finally, we have noticed that exploiting the outputs of several layers improve the classification performances and this is also one idea we propose to exploit in our solutions.

In order to assess the quality of our contributions and to compare with these previous works, we need to study the existing material image datasets, in the next section.

2.5 Material Datasets

To study and validate material recognition methods described in the previous sections, material databases are always needed and we investigate them thoroughly in this section. In our opinion, they can be roughly grouped into three types and we consider their past, their present and their future. In the next three subsections, the three types will be presented according to this timeline.

2.5.1 Datasets taken under controlled conditions

Before the deep learning era, when deep neural networks weren't used to perform large scale image classification, the first group of material datasets were created with the goal of characterizing the appearance of material instances. BRTF (Bi-directional Reflection Transmittance Function) and BTF (Bidirectional Texture Function) are widely used models to output parameterized visual appearance with lighting and viewing condition inputs. In order to build BRTF/BTF models for real-world material instances, images in these kinds of datasets were collected under controlled conditions in labs, and the parameters of these conditions were provided.

Because these datasets focus on the study of material instances, for one instance, images with different visual appearance need to be extremely collected. Hence, the resulting BRTF/BTF model is able to perfectly describe this material instance and enables to produce synthesized images. On the other hand, in each category, the number of instances is rather limited and instances were carefully chosen by the dataset creators. In one words, this type of datasets can be well exploited to build instance-level features, that are invariant to different conditions, but these features may be less transferable to other instances of the same category which are not included in the dataset.

Below are the representative datasets in this category:

- Columbia-Utrecht Reflectance and Texture Database (CUReT) [26]: 61 material samples taken from 205 different lighting and viewing conditions. As only a single material instance is provided per class, no generalization can be done to classify object categories, due to a lack of intra-class variation.

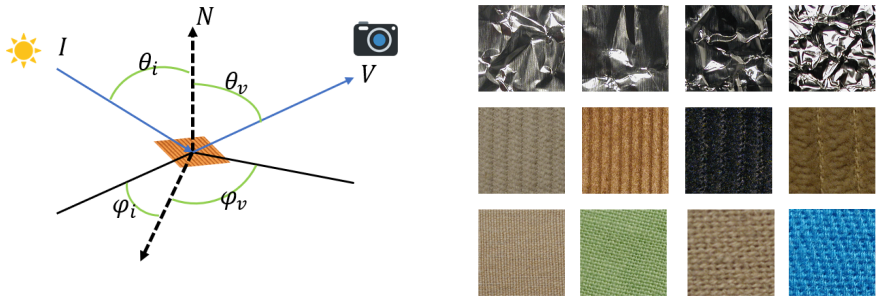


Figure 2.3 – Left: example of image acquisition setup for taking images under controlled conditions. From view point V, we measure BTF values (or take images) of a sample with different illumination sources (I), lighting directions $\{\theta_i, \phi_i\}$ and viewing directions $\{\theta_v, \phi_v\}$. Right: In each row, four samples of a category from KTH-TIPS2 [3] are shown. From top to bottom: aluminium, corduroy, cotton. Intra-class variation is dependent of the category.

- KTH-TIPS2 [3] was created to extend the CUReT database by providing variations in scale, as well as in pose and in illumination, and by imaging other samples for each category. As only four samples are provided per category, this still limits the representation of the intra-class variance of materials observed in real-world scenarios.

- UBO 2014 [27]: a larger dataset taken under controlled conditions , which consists of 7 material categories (carpet, fabric, felt, leather, stone, wallpaper, wood), each of which contains 12 material instances for being capable to represent the corresponding intra-class variances. Full BTF measurements were done using a bi-directional sampling of 151 viewing directions and of 151 lighting directions.

2.5.2 Real-world datasets

In contrast to datasets taken under controlled conditions, in real-world datasets, images are taken in the wild and therefore, the material visual appearances are more varied. It depends on unseen material instance, natural light, stochastic pose, etc. Moreover, images no longer necessarily show only the material but context information surrounding the target. Based on the fact that material images are collected from multiple online sources and images are taken under random conditions, it becomes possible to exploit invariant features of material categories. Most of the recent research studies, based on deep learning network to extract invariant features, can achieve good classification results with these datasets.

The representative datasets in this category are (refer to the gallery in the Fig 2.4):

- Flickr Material Dataset (FMD) [1] is a small but popular real-world material dataset, containing 10 categories and 100 images per category. Images were downloaded from flickr.com and they were carefully chosen to cover a wide range of visual appearance in one category. Masks, locating material region, are also provided for every image. They are helpful for studies where masking out clutter background is needed (e.g. when the influence of background context impacts the classification performance).

- DTD (Describable Textures Dataset) [32] is not a typical material dataset, because instead of defining categories by material name, like wood, water, a collection of images having the same texture attributes (e.g: dotted) is viewed as a category, as illustrated in Fig 2.4. According to earlier work [96], the authors believe that this way to describe

material is more apt to model human's perception of textures. The dataset contains 47 key attributes for a total of 5,640 images, most of them have limited surrounding background.

- Open Surfaces (OS) dataset [33]. OS comprises 25,357 images, each containing a number of high-quality texture/material segments. Many of these segments are annotated with additional attributes such as the material name, the viewpoint, the BRDF, and the object class. Material classes are highly unbalanced and for some of them, only tens of images are available.

- MINC-2500 (Materials in Context Database) [34] is a subset of MINC. Its large size makes it very suitable for training a deep CNN. Images correspond to patches cropped manually from material segments in the wild. Abundant background context appearing around target material makes this dataset quite challenging, see Fig 2.4. It contains 23 commonly-seen material categories and 2500 images per category.

- GTOS (Ground Terrain in Outdoor Scenes) [12]: a dataset for the study of ground terrain recognition, which can be implemented into autonomous driving systems to detect current ground terrain's condition. This dataset is challenging because some inter-class boundaries are ambiguous. For example, GTOS owns 'mud' and 'mud puddle' categories which are visually similar, as shown in Fig 2.5. The dataset consists of 30.000 images covering 40 common classes in outdoor scenes.

- COCO (Common Objects in context) dataset [97]: A dataset of images semantically segmented with 163K images (118K for training, 5K for validation, 20K for testing, 20K for challenging tests) with annotations for 91 stuff classes and 1 'other' class. It contains several material

classes, some of them are classified further into more accurate classes, like water whose instances appear in 'see', 'river', etc.

- 4D-Light [35] is the first medium-size dataset for light-field images. Different from RGB images, light-field images are taken with plenoptic camera, which not only captures light intensity and color in a scene, as a conventional camera does, but also records light directions with multi-view points. Light-field images can be seen as an alternative way to determine materials when it is difficult to determine a material with its surface reflectance or BTF. As in our study case we limited our investigations to RGB input images, light-field information was not investigated in our experiments. This dataset consists of 12 categories with 100 images per category.

2.5.3 Synthesized database

Material, or texture rendering technique, is widely used in computer graphics. As material databases are not abundant, and as synthesized material images can be generated quickly and annotated with no effort, synthesized database seems to be a good way to enrich existing database.

As discussed in subsection 2.5.1, UBO 2014 [27] contains BTF measurements for all of its 84 samples (7 categories $\times$ 12 samples per category). Combined with environment lighting maps (6 natural lights $\times$ 5 directions, taken from the work [98]) and 42 viewing points, a virtual camera can take 1260 synthesized images (30 illuminations x 42 viewing points). To each category corresponds 15.120 images (1260 images per sample x 12 samples per category). The number of images generated is therefore of 105.840 images (15.120 images per category x 7 categories).

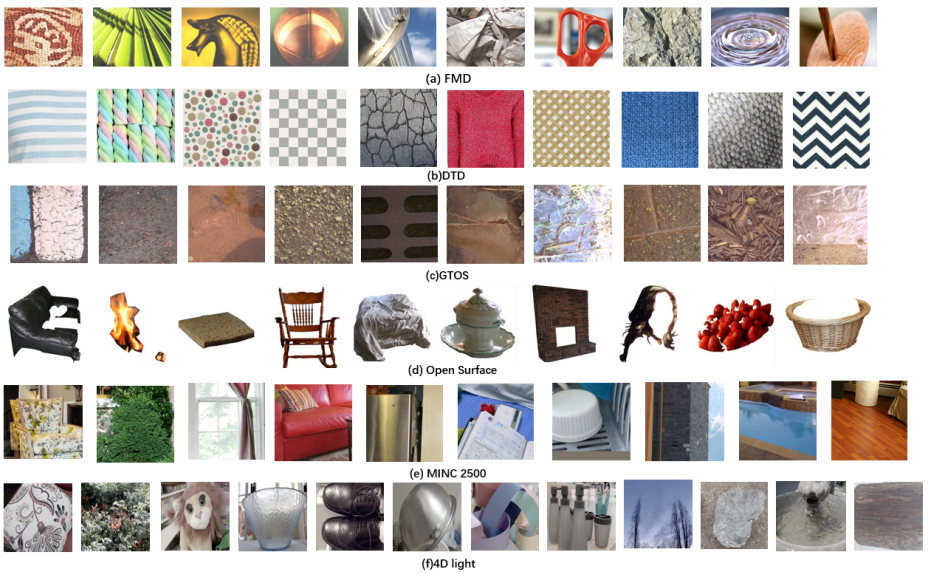


Figure 2.4 – Examples of image categories from Real-world datasets. Names of 'categories' from left to right: (a) fabric, foliage, glass, leather, metal, paper, plastic, stone, water, wood; (b) banded, braided, dotted, chequered, cracked, flecked, grid, knitted, scaly, zigzagged; (c) Painting, sand, mud puddle, stone asphalt, metal cover, paper, ice mud, stone brick, wood chips, plastic; (d) leather, fire, sponge, wood, fabric, ceramic, brick, hair, food, wicker; (e) fabric, foliage, glass, leather, metal, paper, plastic, stone, water, wood; (f) fabric, foliage, fur, glass, leather, metal, paper, plastic, sky, stone, water, wood.

Weinmann et al. conducted some experiments using synthesized images [27]. A classifier was trained and applied to a test dataset containing also real-world images. Thanks to its large scale, a synthesized training dataset can achieve a comparable performance to a small dataset containing real-world images only. Combining these two dataset together can consistently boost the classification accuracy. As reported in the work [27], synthesized images can be considered as a good complementary training data if the size of the

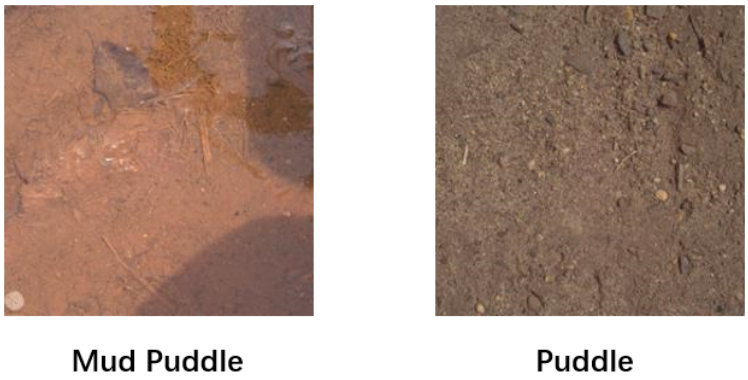


Figure 2.5 – In the GTOS dataset, some images from different categories may look very similar, like mud puddle images and puddle images. The subtle difference can be perceived by water's reflection in the 'Mud Puddle' image.

real-world images training dataset is too small.

Another application happens in semantic segmentation [4]. Like the COCO dataset, this dataset contains many stuff segments (see the Fig 2.6). The input source is a video game: "Grand Theft Auto 5" where a virtual world is created in a way to imitate real world's scenes. It cannot be directly used as a material database, but as synthesized images [27] do not included clutter background, it is a good start to study how to collect synthesized material segments with a virtual "wild" background. These material segments can be seen as complementary training samples. Unfortunately, to the best of our knowledge, so far there is no such material database which simulates materials in the wild.

As this thesis concerns material recognition in the wild, only some datasets discussed in the subsection 2.5.2 were used in the works presented in the following chapters.

Moreover, to demonstrate that the contributions proposed in this PhD

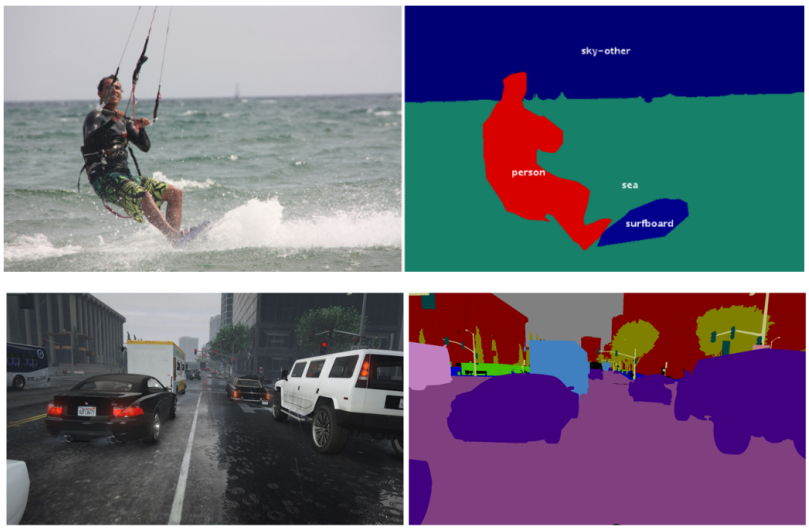


Figure 2.6 – We can crop material segments from the 'wild' from a COCO dataset image (top) or a synthesized image (bottom) described in [4]. Especially for the synthesized dataset, almost infinite samples can be cropped and annotated with no effort.

can be generalized into other recognition tasks, some other datasets, such as CUB-200 2011 [99] for fine-grained classification and MIT-67 [100] for indoor classification will be also involved. The details about these datasets will be introduced in the following chapters.

2.6 Conclusion

In this chapter, we have presented most of the works dealing with material image classification, starting from the first handcrafted features to deep features. We have emphasized the strengths and weaknesses of the state-of-the-art solutions. Indeed, orderless pooling on top of a parametric training distribution fitting seems to be the recent trend providing high classification performances. Nevertheless, these steps could be highly improved by:

- weighting the contribution of each local feature vector in the global pooling, thus taking advantage of the specificity of the material images that shows highly discriminative local areas next to very common (not discriminative) areas,
- paying attention to the quality of the deep feature distribution fitting, which lie in a high dimensional space,
- normalizing the second order statistic representation before applying the classification step.

We propose to exploit these ideas in the next chapters. The last section of this chapter was devoted to the different material datasets, which we have proposed to classify according to their characteristics that meet different demands in each epoch of material classification's history. Then, we decided to use real-world datasets for experiments, considering their suitability for CNN-based approaches. In the following two chapters, we will respectively dive into our main contributions.

Chapter 3

Confidence-based Local Feature Selection For Material Classification

As discussed earlier in this Thesis, the solutions based on Convolutional Neural Networks (CNN) have shown outstanding performances in the material classification task. Many well-known state-of-the-art CNN architectures, such as the ResNet family, are applying a Global Average Pooling (GAP, see Eq. 2.1) as pooling component to aggregate local feature vectors. Most of the time, this pooling operation helps to prevent overfitting but we claim that it has a serious weakness for specific images where small details are crucial to predict their category, such as material images. In this case, the details are lost in the global average, providing non accurate global features. Hence, we decided to find a way to select the most important local features before applying the GAP. However, it is not trivial to select local features, e.g., as shown by the results of Table 1.1, only keeping features of material areas (and removing background features) would not improve classification performance. In this chapter, we choose to add a branch in the classification

network that predicts the confidence the network should have in each local feature vector. The less confident features are filtered out before applying the GAP. Experimental results on three of the material datasets introduced in section 2.5 show that our approach, in terms of classification accuracy and output probability calibration, outperforms recent alternatives. We present these results in Section 3.2.2

3.1 Introduction

Image classification consists in predicting a single class for each input image. Today, many successful approaches rely on automatic extraction of local features with deep neural networks followed by a Global Average Pooling (GAP) layer that merges all the local features into a single global feature vector [37]. Then, a fully connected layer predicts the image class from this global feature vector. Because each local feature vector is evenly averaged with others, every vector equally contributes to the final decision. Consequently, when large areas of the images are ambiguous and useful information is mainly provided by a small part of feature vectors, averaging all the local features could lead to bad predictions.

This phenomenon has already been exemplified in Fig. 1.4 of chapter 1. Large parts of an image can be ambiguous when it comes to identifying the material of the pictured object which can lead to bad predictions. On the other hand, some other areas are very informative and should be emphasized. On the left column of Fig. 1.4 some small parts of the images are masked making the class prediction very difficult. When one has access to these details (right column), class prediction becomes much easier.

In this chapter, we propose a method to automatically select the most

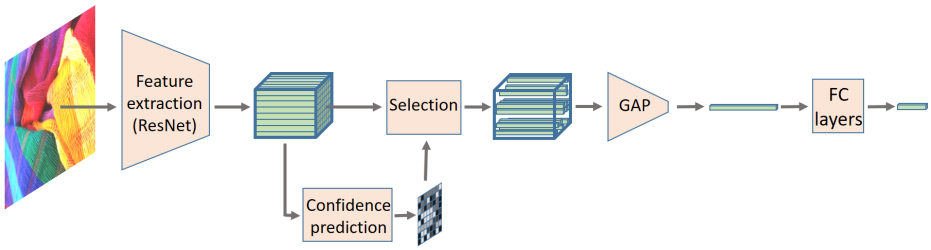


Figure 3.1 – The workflow of the proposed approach. See text for details.

informative local feature vectors before applying the GAP layer. The objective is to ensure that the most relevant features contribute to the final decision while the less informative ones are ignored. We hypothesize that the usefulness of each local feature vector is related to the confidence of the network when predicting the image class from this feature vector. Thus, we trained a two-branch network to output local predictions, as well as associated confidences. These predicted confidences are used to filter out the local feature vectors having lower confidence predictions before averaging all local features into a global feature vector (see Fig. 3.1).

Our contributions, detailed in the following section, are multiple:

- we address the problem of Global Average Pooling in the context of material classification by weighting local features;
- we adapt a very recent and successful approach, designed for global failure prediction [6], to local feature confidence prediction;
- we improve the calibration of the output probabilities for material classification;
- we provide both quantitative and qualitative results on three datasets.

3.2 Related works on model uncertainty and confidence

Since the works related to material classification have been largely discussed in the previous chapter, we concentrate here on the different approaches dealing with the confidence or uncertainty of the deep models.

3.2.1 Model uncertainty

Although modern neural network architectures bring big improvement to prediction accuracy, including CNN structure used for image classification task, their real-world implementation also bring safety concerns [101], especially when incorrect predictions may cause serious consequences in some applications, such as autonomous driving [102, 103] or in the nuclear domain with a monitoring system for critical infrastructure [104]. Hence, to avoid such disasters, an uncertainty or confidence output value made by the model is desirable, in order to show how uncertain or confident the actual prediction is. With such a confidence measure, safety systems are able to decide whether to rely on a model's prediction or to hand the input over to a human. This is part of a research field called failure prediction.

Model uncertainty is also commonly applied in active learning. Labelling data is an essential obstacle for many machine learning applications since it is laborious and costly. Active learning is a framework where a model learned from a small amount of labeled data provides a confidence value on unlabelled data. The data with low confidence would be labelled in the next iteration because for this data, the actual trained model is not confident about its prediction and it would be more likely to gain more improvement

if trained with this data. After several iterations of labelling limited by a budget, a model of good performance can be expected. Approaches based on uncertainty or confidence define and calculate the value of uncertainty or confidence, to select data points before labelling them.

In the next sections, we present state-of-the-art methods in failure prediction and active learning, which define and measure model uncertainty.

3.2.2 Class probability based approaches

Suppose that in the case of image classification, with an input image, a trained CNN outputs its prediction probability vector $\hat{\mathbf{p}} \in \mathbb{R}^K$ according to Eq. 1.1 and Eq. 1.2. A track of confidence or uncertainty prediction study is to deduce model's confidence or uncertainty value, denoted in the following by $conf$ or unc, from $\hat{\mathbf{p}}$. In this group of approaches, Maximum Class Probability and entropy methods are widely used. There also exists lots of task-agnostic improved version, dedicated to human pose estimation [105], image segmentation [106] or object detection [107], to name a few.

Maximum Class Probability (MCP)

One intuitive way to define a scalar confidence value $conf$ about the CNN's prediction on $\mathbf{I}$ is just to pick the maximum probability in the predicted vector:

$$conf = \max_{\forall k \in \mathcal{Y}} \hat{p}_k \tag{3.1}$$

Although MCP method is quite simple, it was demonstrated to be effective in active learning [108] and selective classification [109]. It is considered as a baseline of failure prediction [110].

Entropy

Entropy method, widely used in active learning [111, 112] defines the entropy of $\hat{\mathbf{p}}$ as model uncertainty (unc):

$$unc = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k \tag{3.2}$$

Intuitively, a model feels confident about its prediction when the distribution of $\hat{\mathbf{p}}$ is peaky, i.e. when the entropy value is small. On the other hand, the model hesitates among different classes when the distribution is flat, i.e. when the entropy value is high.

In this case, the confidence value is:

$$conf = \log K - unc = \log K + \sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k \tag{3.3}$$

Unlike the confidence of MCP which is bounded in $[0, 1]$, the confidence's range for the entropy method is $[0, \log K]$. It performs well in [113, 114] and is used as a baseline in [115, 116].

3.2.3 Confidence calibration

MCP and entropy are baselines to estimate confidence but more accurate solutions exist. Guo et al. discovered that for modern deep models, the prediction output is poorly calibrated [5, 117] and sensitive to adversarial attacks [118, 119]. Consequently some meaningless images used for attacks can be classified with high confidence value.

In order to introduce the notion of output calibration, we extracted two plots from the work [5] shown in Fig. 3.2. Suppose that we have N inputs, then we define $\{y^i, \hat{y}^i, conf^i)_{i=1,\dots,N}\}$ where y^i is the ground-truth category of the image $\mathbf{I}_i$ and $\hat{y}^i$ is its predicted category (Eq. 1.4). The confidence value

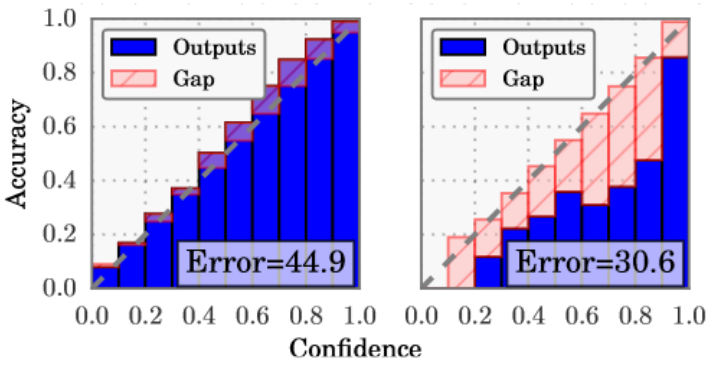


Figure 3.2 – Reliability diagrams for a 5-layer LeNet (left) and a 110-layer ResNet-110 (right) on CIFAR 100 dataset. These two plots are extracted from [5].

$conf^i$ can be obtained with one of the confidence prediction methods, such as MCP for example (Eq. 3.1). Then, according to their confidence values, the inputs are grouped into M equally-spaced bins ($M = 10$ in the two diagrams of Fig 3.2). In each bin m, $\mathcal{B}_m$ represents a set of indices of predictions whose confidence values fall into an interval $(\frac{m-1}{M}, \frac{m}{M}]$. And the height of blue bars in Fig 3.2 represents $acc(\mathcal{B}_m)$, i.e. the classification accuracy within bin m:

$$acc(\mathcal{B}_m) = \frac{1}{|\mathcal{B}_m|} \sum_{i \in \mathcal{B}_m} \mathbf{1}(y^i == \hat{y}^i). \tag{3.4}$$

The average confidence $conf(\mathcal{B}_m)$ is:

$$conf(\mathcal{B}_m) = \frac{1}{|\mathcal{B}_m|} \sum_{i \in \mathcal{B}_m} conf^i. \tag{3.5}$$

In the case of a well calibrated model, the average accuracy should be similar to the average confidence for every bin $m \in \{1, ..., M\}$ and visually the diagrams should plot an identity function (shown as gray dash lines in Fig 3.2). Fig. 3.2 shows a typical example illustrating that early deep models such as LeNet [120] from 1998 (left plot) provide well calibrated outputs whereas modern networks such as ResNet-110 (right plot) are over-confident.

It seems that the decrease of the error rate from 44.9% to 30.6% between these two networks comes at the cost of a reduction of the quality of the output calibration.

From this finding, Guo et al. [5] listed and tested a number of calibration approaches to remedy it. Among them, a simple extension of Platt Scaling [121], called **temperature scaling**, appears to be the most reliable. It introduces a scalar hyper-parameter, namely the temperature T, into the softmax operation (Eq. 1.2) so that the output is expressed as:

$$\hat{\mathbf{p}} = softmax(\hat{\mathbf{z}}, T) \tag{3.6}$$

where the kth element $\hat{p}_k$ is defined as:

$$\hat{p}_k = \frac{\exp\left(\hat{z}_k/T\right)}{\sum_i^K \exp\left(\hat{z}_k/T\right)}. \tag{3.7}$$

When $T > 1$, the temperature is softening the $softmax$ operation and thus making the distribution of $\hat{\mathbf{p}}$ more flat. For example, in the extreme case where $T \to \infty$, $\hat{\mathbf{p}}$ follows a uniform distribution and all the elements $\hat{p}_k$ are equal to $1/K$. Adding this scaling in the softmax operation, is decreasing the confidence of the network and, thus provides more calibrated outputs. Since T does not change the maximum index of $softmax$ in the Eq. 1.4, it is worth mentioning that this temperature scaling does not affect the model prediction accuracy.

Temperature scaling was proven effective in object detection [117] and viewed as an important ingredient to the method ODIN (Out-of-DIstribution detector for Neural networks) for detecting out-of-distribution samples [122].

3.2.4 Monte-Carlo Dropout

Recently, uncertainty prediction with Bayesian neural networks has gained a lot of interests [123, 115, 124, 125, 126]. Here, we present a representative

work, called Monte-Carlo Dropout [123, 115]. Initially, dropout is a regularization method which zeros out some activation outputs of some layer when training a CNN, and which is able to mitigate the overfitting problem [127, 128]. Monte-Carlo Dropout is a recent approach to approximate inference of Bayesian CNNs with the use of dropout. Different from the CNNs presented before, in Bayesian CNNs, some prior probability distribution is placed over the set of every layer's parameters $\boldsymbol{\Theta} = \left\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^L\right\}$:

$$\boldsymbol{\Theta} \sim P(\boldsymbol{\Theta}) \tag{3.8}$$

Furthermore, output $\hat{\mathbf{p}}$ becomes:

$$\hat{p}_k = P(y = k|\mathbf{I}, \boldsymbol{\Theta}) \tag{3.9}$$

According to Eq. 3.8, $\boldsymbol{\Theta} = \left\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^L\right\}$ does not contain parameters of fixed values but all of them follow a prior distribution. In other words, we only have access to probabilities to estimate their values. For example, with the same input image $\mathbf{I}$, a Bayesian CNN samples its parameters $\boldsymbol{\Theta}$ with the prior distribution and the output $\hat{p}_k$ can vary between two inferences. And if the number of inferences is large enough, we can observe $\hat{p}_k$ following a distribution of Eq. 3.9. The uncertainty is thus measured from the variance of the output's distribution $P(y = k|\mathbf{I}, \boldsymbol{\Theta})$. More formally, variance of the distribution over $\hat{p}_k$ can be related to an uncertainty value:

$$unc = \frac{1}{K} \sum_{k=1}^{K} \sqrt{\mathbb{E}_{P(\boldsymbol{\Theta})}[P(y = k|\mathbf{I}, \boldsymbol{\Theta})^2] - \mathbb{E}^2_{P(\boldsymbol{\Theta})}[P(y = k|\mathbf{I}, \boldsymbol{\Theta})]} \tag{3.10}$$

and the expectation $\mathbb{E}_{P(\mathbf{\Theta})}[P(y = k|\mathbf{I}, \mathbf{\Theta})]$ can be approximated by Monte-Carlo Dropout:

$$\begin{aligned}\mathbb{E}_{P(\mathbf{\Theta})}[P(y = k|\mathbf{I}, \mathbf{\Theta})] &= \int P(y = k|\mathbf{I}, \mathbf{\Theta})P(\mathbf{\Theta})\mathrm{d}\mathbf{\Theta} \\ &\approx \int P(y = k|\mathbf{I}, \mathbf{\Theta})Q(\mathbf{\Theta})\mathrm{d}\mathbf{\Theta} \\ &\approx \frac{1}{S}\sum_{s=1}^{S} P(y = k|\mathbf{I}, \hat{\mathbf{\Theta}}_s)\end{aligned} \tag{3.11}$$

with $\hat{\mathbf{\Theta}}_s \sim Q(\mathbf{\Theta})$ where $Q(\mathbf{\Theta})$ is the Dropout distribution. In practice, dropout operation is added after each layer l. For an image $\mathbf{I}$, we run its inference S times and the average output is the expectation in Eq. 3.11. For more details, please refer to the works [123, 115].

3.2.5 True Class Probability

Although MCP is simple and effective, it suffers from poorly calibrated issue [5]. In a recent work [6], another inconvenient property for failure prediction was pointed out: some failure examples with high MCP value overlap with successful examples (see Fig 3.3, left), making the distinction between them difficult. This problem occurs because of the overconfidence of modern deep models. Consequently, Corbiere et al. proposed to use the True Class Probability (TCP) as a confidence measure [6]. The right plot of Fig 3.3 shows that this measure helps to discriminate between the success and failure cases. They conclude that TCP is a better choice in term of confidence value.

More precisely, TCP is the probability prediction value of the ground-truth class y in $\hat{\mathbf{p}}$:

$$conf = \hat{p}_{k=y} \tag{3.12}$$

Unlike MCP, the ground-truth class y is required to evaluate the TCP. So it

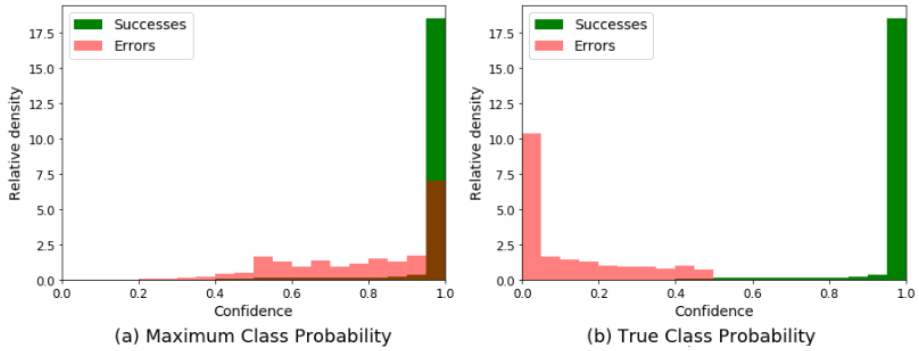


Figure 3.3 – Best viewed in color. Image samples in test dataset of CIFAR 100 are regrouped into confidence bins. Y-axis represents the percentage of the number of samples in one bin to the total number of samples. Left: confidence measured by MCP. Right: confidence measured by TCP. These plots are extracted from the work [6].

can be evaluated only for training data for which the ground-truth category is available. For test data, Corbiere at al. proposed to train a network to predict this value [6] and add a branch in their current network whose output is the predicted TCP value. During the training, when a minibatch of N input data points in one iteration is $\{(\mathbf{I}^i, y^i)_{i=1,\ldots,N}\}$ (where y^i is category label of image $\mathbf{I}_i$) in addition to cross entropy objective function defined by Eq 1.9 for classification, a mean square error objective function supervises the branch for accurate confidence prediction:

$$\mathrm{L}_{\mathrm{conf}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{conf}^i - \hat{p}_{k=y})^2, \tag{3.13}$$

where $\hat{conf}^i$ is predicted TCP value for image $\mathbf{I}_i$.

Sharing the same idea, Terrance DeVries et al. uses TCP predicted by neural networks to detect Out-of-Distribution samples [129]. Donggeun Yoo et al. predicted loss for active learning [130], where loss is equal to $-\log \hat{p}_{k=y}$. So the approach can be considered as a variant of TCP prediction.

3.2.6 Other alternatives

Applied to detect Out-of-Distribution samples, an idea of De Vries and Taylor consists in letting the network have partial access to ground truth information during training. How much the information the network needs is related to prediction uncertainty [129]. Jiang et al. proposed a new confidence measure, called 'Trust score' which is the ratio between the distance from a test sample to its second nearest class cluster and the distance to the predicted class cluster [131]. Dan Roth and Kevin Small introduced a similar method but measured a margin between probabilities of predicted class and second predicted class [132]. For SVM classifiers, uncertainty or confidence can be defined with the use of a decision boundary [133, 134, 135]. In the speech recognition field, bi-directional lattice RNN is adapted for confidence prediction [136, 137]. Finally, another group of methods is based on multiple independent models, such as several CNNs trained separately for the same task, to measure disagreement among them in term of uncertainty [138, 139, 140, 141]. While effective, the computational burden is also heavier than approaches based on a single model.

3.2.7 Conclusions

Finally, in this section, we have presented many alternatives to estimate the confidence of a deep neural network. Because the True Class Probability (TCP) is easy to get from the ground-truth and provides good results in the work [6], we propose to leverage this information to weight the contribution of each local feature vector of our feature map, before applying the Global Average Pooling. Note that none of the previous papers have used the confidence to select local features, but instead mainly propose to exploit the

confidence to predict failure in the classification task.

Similar to our idea, Qiu proposed to weight the contribution of each local feature vector and to compute a Global Weighted Average Pooling (GWAP) [142]. The main problem of this solution is that it increases the number of trainable parameters without adding any supervision, increasing the risk of overfitting. Indeed, the weights in the work [142] are learned by back-propagating the gradient of the classification loss and are not related to the confidence of the network. On the contrary, our solution consists in supervising the weight learning with a confidence map, as detailed in the next section. We will show in the experimental section, that our approach outperforms the GWAP proposed by Qiu.

3.3 Our approach

3.3.1 Deep neural network with Global Average Pooling

Let us denote a training sample as $(\mathbf{I}, \mathrm{y})$ where $\mathbf{I} \in \mathbb{R}^{W \times H \times 3}$ is an RGB image and $\mathrm{y} \in \mathcal{Y} = \{1, ..., K\}$ is its ground truth category. Recent deep networks such as the ResNet series can be decomposed into three parts:

- the feature extractor f_{conv} constituted by convolutional layers;
- the Global Average Pooling (GAP) f_{avg} that discards any spatial information;
- a fully-connected (FC) layer f_{FC} followed by a Softmax function.

Hence, the output is the predicted distribution $\hat{\mathbf{p}}$ of the probabilities over all the classes:

$$\hat{\mathbf{p}} = Softmax(f_{FC}(f_{avg}(f_{conv}(\mathbf{I})))). \tag{3.14}$$

Note that $\hat{\mathbf{p}}$ is a K-dimensional vector $\hat{\mathbf{p}} = [\hat{p_1}, \hat{p_2}, ..., \hat{p_K}]$.

While training the network, the parameters are updated in order to minimize the cross-entropy loss (over a batch of images) L_{ce} (see Eq. 1.9) between the ground-truth category y and the predicted $\hat{\mathbf{p}}$.

Since the FC layer and the GAP layer are linear transforms, they can be switched in the process so that the FC layers are applied before the GAP. The predicted probabilities are then:

$$\hat{\mathbf{p}} = Softmax(f_{avg}(f_{FC}(f_{conv}(\mathbf{I})))). \tag{3.15}$$

Obviously, in this case, the fully-connected layer is applied individually to each local feature vector returned by f_{conv}, in the form of 1x1 convolutions, as shown in the left workflow of Figure 3.5. This formulation is interesting for our approach since it represents individual processing of each feature vector.

This architecture has shown very good results in many classification applications, but it might not be the optimal solution for material image classification. Indeed, as discussed in the introduction and confirmed in many successful orderless aggregation solutions proposed for this task, large areas of material images can be ambiguous about the class of the considered image, while some details appear to be very discriminative. A simple average of all the local features into a global vector can lead to loss of useful information. This is illustrated with the two images from Fig. 3.4, where we propose to have a look at the map prediction provided by the network without applying the GAP:

$$\widehat{\mathbf{P}} = Softmax(f_{FC}(f_{conv}(\mathbf{I}))), \tag{3.16}$$

where each local feature vector $\mathbf{v_i}$ is associated with one local prediction at the i^{th} location in the map as follows:

$$\hat{\mathbf{p}}_i = \widehat{\mathbf{P}}_{:,i}. \tag{3.17}$$

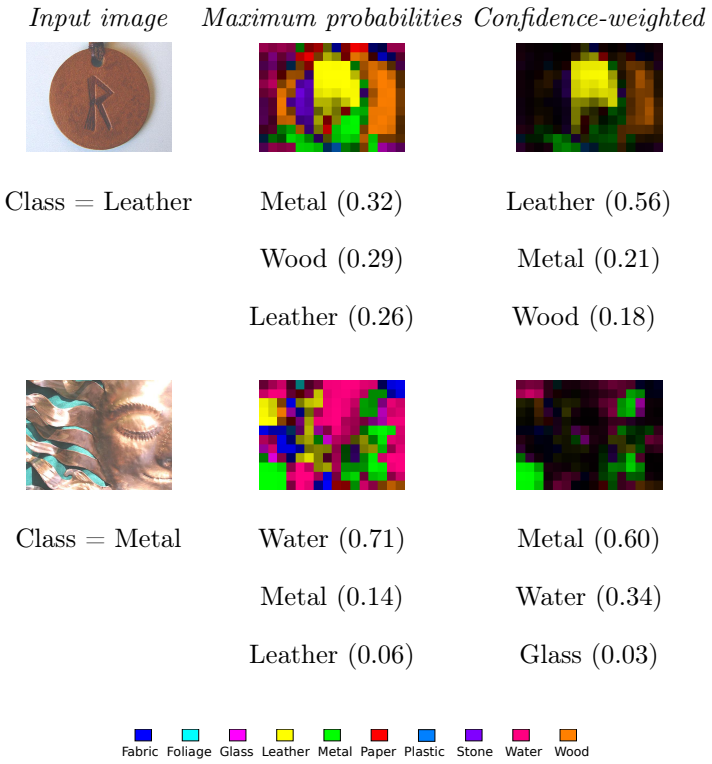


Figure 3.4 – Local decision maps for two different images. The two right columns show the categories and scores of the locally maximum probabilities before (second column) and after (third column) weighting them with the corresponding local confidences.

The first column of this figure shows two images with their ground truth category. The second column shows, for each local feature vector $\mathbf{v_i}$, the category $\hat{y}_i$ that locally has the maximum score as well as its score $(\hat{\mathbf{p}}_i)_{\hat{y}_i}$:

$$\hat{y}_i = \underset{k \in \mathcal{Y}}{\operatorname{argmax}} (\hat{\mathbf{p}}_i)_k, \tag{3.18}$$

$$(\hat{\mathbf{p}}_i)_{\hat{y}_i} = \max_{k \in \mathcal{Y}} (\hat{\mathbf{p}}_i)_k. \tag{3.19}$$

As shown at the bottom of the figure a false color is associated to each

category. The lightness of these colors is proportional to the score of the corresponding category $(\hat{\mathbf{p}}_i)_{\hat{y}_i}$, i.e. dark colors mean that the associated probability is low whereas lighter colors represent high probabilities.

Below each illustration, we mention the three most probable categories provided by the whole network, including the GAP. In this classical case, each image gets a single global probability vector and these three mentioned categories are the ones that get the highest probabilities. We can see that, for both examples, the most probable category is not the ground truth one, leading to an incorrect classification for these images. We can also notice that most of the local predictions are associated with very light colors, showing that the network is overconfident in most of the cases, even for non-correct predictions.

As illustrated in the last column of Fig. 3.4, our aim is to select the most important local feature vectors, and remove the least important ones, before applying the GAP in the network. We propose to relate the "importance" of each local feature vector to its associated class prediction confidence.

3.3.2 Predicting local confidences

In order to select the most important local feature vectors, we propose to train a branch of our network to predict the confidence of the category prediction related to each local feature vector. In this aim, we take inspiration from [6] that deals with failure prediction in image classification task. In this paper, the authors tried to find out which images are potentially misclassified by estimating the True Class Probability (TCP) along with the category prediction. We propose to adapt this approach in order to predict local TCP that help us to select the most confident local feature vectors.

As defined in [6], the TCP is the predicted probability of the ground truth

category y of the considered image:

$$TCP = \widehat{p}_{k=y} \tag{3.20}$$

Given a local feature vector, a high TCP means that this vector leads to a prediction that has a high probability to be the correct class, which means that we should trust it. On the contrary, if the TCP of a local feature vector is low, this means that it predicts a low probability for the correct class and, so, should not be considered in the final global decision.

Obviously, at test time, the ground truth category is not available and therefore neither is the TCP. Thus, we propose to add a branch f_{conf} in our network whose the aim is to predict the TCP of each local feature vector. As illustrated on the right of Fig. 3.5, the input of this branch is the feature map extracted from the image and its output is a predicted TCP map:

$$\widehat{\mathbf{TCP}} = f_{conf}(f_{conv}(\mathbf{I})) \tag{3.21}$$

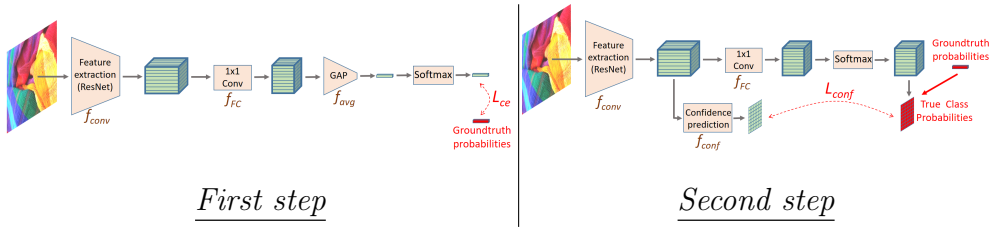


Figure 3.5 – The two successive training steps. See text for details.

The idea of this new branch is that the network is learning if some local features are rather ambiguous or not with respect to the category they predict. The details about the structure of f_{conf} are available in the next section.

Thus, if the network is able to automatically predict the TCP of each local feature vector, we can use these predictions as the confidence we should

Table 3.1 – The two steps of the proposed learning scheme.

Step	Loss	Frozen parameters	Learned parameters
Step 1	L_{ce}	f_{conf}+f_{conv}	f_{FC}
Step 2	L_{conf}	$f_{conv} + f_{FC}$	f_{conf}

have in each vector and select the most confident ones before applying their average (see Fig. 3.1).

In order to illustrate the intuition behind our idea, we show in the last column of Fig. 3.4 how the local probabilities are transformed when they are weighted by their corresponding confidence (TCP). It is worth mentioning that this weighting scheme is just presented for illustration. In practice, the confidences are used to select the most confident local feature vectors, with a threshold, as detailed below.

3.3.3 The training process

The whole training process is composed of two steps as shown in Fig 3.5. During the first step, the classification network is trained with the cross-entropy loss L_{ce}. After reaching convergence, the parameters of the trained network are frozen and the confidence prediction branch is trained. To this end, we feed the classification network with images and their ground truth category in order to evaluate their ground-truth TCP map **TCP** (confidence map). Then, we train the confidence prediction branch f_{conf} so that it is able to automatically predict the TCP map for each image by minimizing L_{conf}, the mean square error between the ground truth map **TCP** and the predicted one $\widehat{\mathbf{TCP}}$. A summary of the two training steps is provided in Table 3.1.

3.4 Experiments

In this section, we present the experimental results provided by our approach in a material classification task. The tests are conducted over three datasets and the results are compared with recent alternatives.

3.4.1 The datasets

Three classical material datasets are used for testing (see Fig. 2.3 and Fig. 2.4 for image examples). The Flickr Material Dataset (FMD) [1] is a popular benchmark material dataset which contains 10 categories with 100 images per category. KTH-TIPS-2b [3] (called hereafter KTH) has 11 categories with 432 images for each category. The 4D-light dataset [35] is a light-field material dataset which consists of 12 categories with 100 images per category.

For FMD and 4D-Light, we run a 5-fold experiment by splitting the dataset into 5 non-overlapping subsets. For each run, 4 subsets were used for training and 1 for testing. For KTH, following the experiments from [43], we randomly chose half of the images for training (216 per category) and half for testing. The results are also averaged over 5 runs.

3.4.2 Tested approaches

Our method is compared with several recent and classical approaches. Here, a brief presentation to each of them are given (see more details about the approaches in the section 3.2.2)

The **baseline** is a classical network without weighting scheme as illustrated in the left of Fig. 3.5 (first step).

Since output probability calibration is one aim of our framework, we pro-

pose to compare our results with the **temperature scaling** solution [5]. As recommended by the authors, this approach required a validation set to fix the temperature. Thus, for this method, about 10% of the images for each category were randomly extracted from the training set to constitute the validation set. The test set is the same for all the approaches for fair comparison. The baseline was also tested on this reduced training set (mentioned as "90% Training" in the Tables) for information.

The entropy of the predicted class probabilities could be seen as a confidence score and is used in some recent papers [115, 116]. Indeed, a peaky prediction vector (low entropy) means that the network is confident in its prediction, while a flat probability vector (high entropy) shows that the network is hesitating between the different classes. In our experiment, we propose to compare our method with a local selection based on the entropy of each local classification prediction. In the experiment, we have chosen the threshold that performs best for this approach. We have also tested the maximum probability (**MaxProb**) as a confidence measure.

The Monte-Carlo Dropout approach [123] is denoted **MCDropout** in the Tables.

The Global Weighted Average Pooling (**GWAP**) is similar to our approach, except that it predicts a score map without any additive supervision than the classification loss [142]. In this case, the architecture is similar to our proposed solution with two branches that are simultaneously trained with a single cross-entropy loss L_{ce}.

Finally, we are also presenting the results of state-of-the-art approaches for each of the three material datasets used. Even if the architectures are different, the results inform us about the best current results provided on these datasets.

3.4.3 Experimental settings

For all the tested models, the network backbone is ResNet-50 [37] pretrained on the ImageNet dataset [143].

Our confidence prediction block f_{conf} is composed of 3 successive 3x3 convolutional layers with respectively 384 kernels with ReLu, 192 kernels with ReLu and 1 kernel with a Sigmoid. The input of this block is the concatenation of the feature maps from the two last convolutional blocks of the backbone.

As previously explained, the aim of our solution is to filter out the least confident local feature vectors before applying the GAP. One threshold has to be fixed in order to decide which vectors should be discarded. For all our experiments, we have chosen to remove the feature vectors whose associated predicted confidence is lower than 0.2. This threshold is fixed for all the runs and all the datasets.

For all the approaches, channel-wise normalization is applied (zero mean and unit variance) as a pre-processing. For data augmentation, all images are resized to 384x384. 8% to 100% of the area of each image is randomly cropped, transformed with a random aspect ratio between $\frac{3}{4}$ and $\frac{4}{3}$ of the original aspect ratio, and resized to 352x352. Additionally, random horizontal and vertical flip with a probability of 50% is applied to each image. At test time, we just use the images with their original sizes.

We use Adagrad as optimization algorithm with a mini-batch size of 8. The learning rate starts from 0.01 at step 1 and from 0.001 at step 2 and is divided by 10 after 5 epochs with no improvement about minimizing training loss at step 1 and after 30 epochs at step 2.

3.4.4 Results

The first results are presented in Table 3.2, where three criteria are provided: the classification accuracy, the Expected Calibration Error (ECE) [144] and the Negative Log Likelihood (NLL) [5].

ECE calculates the weighted average difference between classification accuracy and average confidence in each bin. As mentioned in the sec 3.2.3, samples in the test dataset can be regrouped into each bins $m \in \{1, ..., M\}$ and we can calculate $acc(\mathcal{B}_m)$ and $conf(\mathcal{B}_m)$, following respectively Eq. 3.4 and Eq. 3.5. The ECE value of these predictions is:

$$ECE = \sum_{m=1}^{M} \frac{|\mathcal{B}_m|}{N} |acc(\mathcal{B}_m) - conf(\mathcal{B}_m)| \tag{3.22}$$

where $|\mathcal{B}_m|$ is the number of predictions in $\mathcal{B}_m$. The smaller the ECE value, the better is the confidence calibration. And the minimum ECE value is zero with perfect calibration, i.e. $acc(\mathcal{B}_m) == conf(\mathcal{B}_m)$ for every bin $m \in \{1, ..., M\}$.

And NLL is refered to cross entropy loss (see Eq. 1.9) in the context of deep learning and is considered as a standard criteria for the quantity of probability prediction given by a model [145]. Its low value demonstrates well-calibrated probabilities.

In Table 3.2, we can see that the temperature scaling overall actually improves the output calibration over the baseline with the same settings, while preserving the accuracy. Indeed, the single aim of this approach is to calibrate the output probabilities of the network without modifying the classification accuracy of the baseline, since the probability ranking is not modified by this scaling. Nevertheless, for the KTH dataset, we can see that the scaling does not improve the calibration. We think that it is due to the high diversity within each category of this dataset, that makes it difficult

Table 3.2 – The results of the tested approaches on the three datasets. Averages over 5 runs.

	FMD			KTH			4D-Light		
Approaches	ECE	NLL	Acc[c]	ECE	NLL	Acc[c]	ECE	NLL	Acc[c]
Baseline	0.080	0.517	83.2	0.060	0.54	82.1	0.074	0.537	83.1
Baseline[a]	0.087	0.543	83.1	0.064	0.55	81.9	0.061	0.535	83.0
Temperature[b]	0.071	0.529	83.1	0.120	1.20	81.9	**0.049**	0.532	83.0
Entropy	0.070	0.510	83.1	0.060	0.54	82.1	0.073	0.537	83.1
MaxProb	0.079	0.517	83.2	0.060	0.54	82.1	0.074	0.537	83.1
MCDropout	0.081	0.516	83.0	0.060	0.54	82.2	0.073	0.537	83.0
GWAP	0.067	0.525	83.3	0.063	0.55	81.7	0.063	0.529	84.0
Our	**0.061**	**0.470**	**84.8**	**0.058**	**0.52**	**83.1**	0.058	**0.527**	**84.8**

[a] *90% Training*

[b] *90% Train./10% Val.*

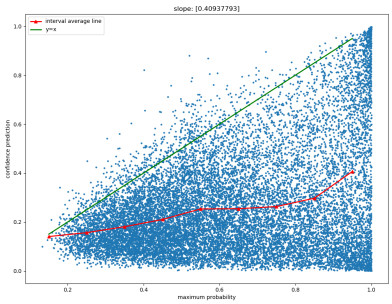
[c] *Acc: abbreviation of Accuracy(%)*

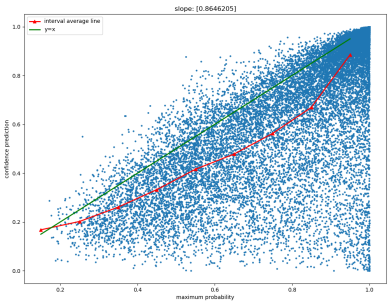
Table 3.3 – Comparison of the classification accuracy (%) with the state-of-the-art solutions on the three datasets.

Approaches	FMD	KTH	4D-Light
LFV+FC-CNN [61]	83.5	**83.1**	-
Deep Ten [43]	80.2	82.0	84.1
FV-CNN [31]	82.4	81.1	82.6
B-CNN [63]	80.5	80.2	84.3
Confidence prediction (Our)	**84.8**	**83.1**	**84.8**

to estimate the temperature scaling on the validation set. Interestingly, the entropy-based approach also reduces the calibration error on FMD but does not improve the accuracy over the baseline. Overall, entropy- and maximum probability-based approaches have very small impacts on the results. The GWAP approach provides inconsistent improvement 'for the calibration quality and slightly improves the accuracy. We can notice that our approach clearly outperforms all the tested methods for the three criteria. Indeed, by discarding the least confident local feature vectors, our model is able to predict calibrated and accurate probabilities. It is worth mentioning that the architectures of our solution and GWAP are identical. This clearly shows that supervising the second branch with the True Class Probabilities is a good solution to predict accurate confidences and select the best local features.

Finally, we propose to compare the accuracy provided by our method with state-of-the-art solutions designed for material classification (see Table 3.3). The reported results have been extracted from the published papers, when available. Despite the simplicity of our approach, we notice that





Incorrect local predictions *Correct local predictions*

Figure 3.6 – Confidence prediction versus the maximum predicted probability for the FMD test set.

it outperforms all the recent state-of-the-art solutions designed for material classification. These results confirm that it is very interesting to concentrate the category decision on specific areas of material images and that predicting the confidence of each local feature vector is an effective way to do that.

3.4.5 Predicted confidence analysis

In this section, we propose to look deeper at the maximum probabilities and associated predicted confidences by analysing the two plots of Fig. 3.6. Each point of these plots is associated with one local feature vector. The vertical axis is the confidence predicted by our confidence-branch for this vector and the horizontal axis is the value of the maximum probability ($\max_{k \in \mathcal{Y}} \hat{p_k}$, see Eq. 3.19) predicted by our classification-branch. If this maximum probability corresponds to the true category of the feature vector, the point is drawn on the right plot. Otherwise, it is drawn on the left plot. Thus, on the left plot, even if the maximum probability is high, the associated predicted confidence should be low, because the category prediction is not correct. On the right

plot, the predicted confidence should follow the maximum probability value. For each plot, we have drawn in red the average trend of confidence. We can see that the confidence we are predicting is rather stable at a low value for the incorrect predictions and that it is almost linearly increasing with the max probability for correct predictions. These plots show that our confidence-branch predicts accurate confidences that allow to remove ambiguous local feature vectors.

3.4.6 Threshold analysis

As described in the previous subsection 3.4.3, for all the tests, we set a threshold value and only input those local feature vectors with associated predicted confidence higher than that value into the next average pooling.

In this section, with the FMD dataset, the impact of threshold value on the classification accuracy is analyzed. The results are shown in Fig 3.7, where horizontal axis is the threshold value, increasing from 0 to 1, and vertical axis is classification accuracy(%). According to this figure, the highest accuracy peak can be found when the threshold is around 0.2. Hence, the threshold of 0.2 is used in our approach. It should be mentioned that, in the case where all the feature vectors are eliminated in one image, the average pooling will take all the feature vectors as input. Consequently, the accuracy value with the threshold 1 is equal to the one obtained when the threshold is 0.

3.4.7 Binary or non-binary weights

In our approach, selection procedure plus average pooling could be viewed as a special case for weighted average pooling where each feature vector is weighted by a binary value, i.e. '0' or '1'. '1' means to select the feature vector

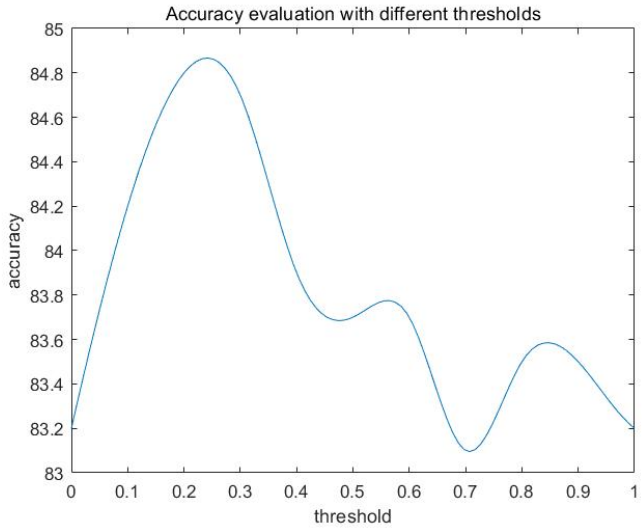


Figure 3.7 – Classification accuracy(%) versus threshold for the FMD test set

and '0' indicates that corresponding vector should be eliminated. And this value is derived from binarizing predicted confidence value, i.e. $\widehat{TCP}$, with a threshold:

$$weight = \begin{cases} 1, & \text{if } \widehat{TCP} \geq threshold \\ 0, & \text{otherwise} \end{cases}$$

In this section, we investigate our approach's performance if weighting each feature vector directly with its associated predicted confidence. The results with binary and non-binary weights are provided in Table 3.4. According to this table, both binary and non-binary method improve classification performance over baseline. However, non-binary method has mixed performance for probability calibration, showing that it is recommended to keep and to delete feature vectors rather than to weight them with soft weights. In other words, feature vectors with medium confidence are as important

Table 3.4 – The results of our approach with binary and non-binary weights

	FMD			KTH			4D-Light		
Approaches	ECE	NLL	Acc[c]	ECE	NLL	Acc[c]	ECE	NLL	Acc[c]
Baseline	0.080	0.517	83.2	0.060	0.54	82.1	0.074	0.537	83.1
Binary	0.061	**0.470**	**84.8**	**0.058**	**0.52**	**83.1**	**0.058**	**0.527**	**84.8**
Non-binary	**0.052**	0.479	84.7	0.090	0.57	83.0	0.072	0.554	84.3

[c] *Acc: abbreviation of Accuracy(%)*

as those with high confidence for probability calibration. Furthermore, feature vectors with low confidence are harmful for probability calibration and should be removed.

3.4.8 Performance on large-scale dataset

In the previous experiments, we have validated our proposed approach on three small-scale material datasets. And in this section, we choose a large-scale material dataset, MINC-2500, containing 23 categories and 2500 images per category. We run a 5-fold experiment with the splits provided by dataset's developers. The experimental settings are the same as those in the subsection 3.4.3. As for learning scheme, the whole network without confidence prediction branch is learned in order to obtain the results of baseline. Then, we build confidence prediction network by copying the baseline network as the main branch and by adding a confidence prediction branch with randomly initialized parameters. We first train the confidence prediction branch and then finetune the whole network with target ture class probability maps **TCP** produced by the baseline network.

The results are shown in the Table 3.5. This time, our approach (third row) is not as good as baseline (second row). In particular, we found that the values of ECE and NLL are much bigger than those given by the baseline. In order to find out the reason, we did two tests based on our approach and recorded their results (fourth and fifth rows). Speaking of the tests, we cheated the selection procedure and we removed (resp. kept) bad (resp. good) local feature vectors by comparing their associated ground-truth confidence scores with the threshold. Here, 'good' local feature vector means its corresponding local classification prediction is correct and 'bad' local feature vector means wrong local prediction. According to the results of the two tests, if bad local feature vectors were all well detected, our approach would have greatly improved the performance in comparison with the baseline. In other words, these results imply that our approach has difficulties in giving low enough confidence score to these bad vectors so as to delete them later. The reason is that, when training the confidence prediction network, most of training samples given by the baseline network are either good local feature vectors with peaky probability distribution of their prediction or bad ones with relatively uniform probability distribution. Consequently, the confidence prediction network cannot be fully trained with hard samples which have bad local feature vector with peaky probability distribution. However, during the test phase, for each image, the baseline network learns to select certain local feature vectors and let them dominate the final classification prediction after global average pooling. To to be dominant, the probability distribution of corresponding local predictions become peaky. But unlike what happens in the training phase, the hard samples appear more often in the wrongly-classified images. Due to insufficient training with hard samples, the confidence prediction network is not able to detect them and to

Table 3.5 – The results of our approach and baseline for MINC-2500

Approaches	ECE	NLL	Accuracy(%)
baseline	0.0702	0.7594	78.57
ours	0.1539	1.2990	78.36
ours(bad local vectors removed)	0.0124	0.2459	95.49
ours(good local vectors kept)	0.1352	1.0862	80.11

give them low enough confidence scores. Even worse, those hard samples are not deleted but can be further emphasized if other less salient local feature vectors are detected and deleted. Hence, after processed by our approach and GAP, wrong final predictions risk being more wrong and encourage ECE and NLL to simultaneously have high values.

3.5 Conclusions

In this chapter, we have proposed an original solution for material classification. Since material images present large ambiguous areas that do not help or even influence the classification process, our idea consists in removing these parts from the feature maps before taking the average final decision. To this end, we have proposed to add a branch in the classical network in order to predict the confidence associated with each local feature vector. This branch is trained to predict the True Class Probability (TCP) during the learning step. This TCP can be seen as a confidence and allows us to filter out ambiguous or disturbing local feature vectors before applying Global Average Pooling. Experimental results on three small-scale datasets show that our solution outperforms other alternatives and classical models for both the ac-

curacy of the network and the output probability calibration. In order to select the most confident feature vectors, a fixed threshold has been used in this chapter, it has been set empirically. Future works will consist to train the network to predict this value.

Chapter 4

Sparse coding and normalization for deep Fisher score representation

In Chapter 3, we have proposed a solution to select the most relevant local features from an image before pooling them in a global representation. In this Chapter, we propose to pay attention to the pooling step itself. Among the orderless pooling strategy, the Fisher Scores have been shown to outperform many alternatives on classification tasks. However, they require to fit a model on the training data and their performances are very dependent on the quality of this model and on the normalization steps applied to these second order statistics. In this chapter, we propose to embed the Fisher scores in an end-to-end trainable deep network by concentrating on two crucial elements: adapting the encoding to the deep features and normalizing the extracted second order statistics. Therefore, we propose to make use of a deep sparse coding module that allows to sample the center of each Gaussian function from the learned subspace and thus to better fit the high dimensional data distribution. Second, we introduce a new normalization module that com-

putes an approximate square root matrix normalization well adapted to the Fisher vectors. These processing steps are embedded in a deep network so that all the modules work together for the sole purpose of improving classification performance. Experimental results show that this solution clearly outperforms the existing approaches in the context of material, indoor scenes or fine-grained image classification.

4.1 Introduction

Deep neural networks have emerged as an essential solution for performing classification tasks. In these networks, convolutional layers extract accurate local features that are pooled to a global feature vector which is sent to fully connected layers for classification. The first networks neglected the pooling step and directly sent the set of local features in the dense layers [36], while the series of ResNet apply a global average pooling to decrease the dimension of the global feature vector and hence reduce the number of parameters of the network [37]. Orderless pooling was widely used before convolutional neural networks (CNN) with bags of visual words (BOW) [38], VLAD [39] or Fisher Vectors [40] and has shown to provide good results when applied to CNN features [31, 41]. Among them, Fisher Vectors (FV) were the most promising because they generalize VLAD and BOW. The main idea of FV is to model the distribution of the training data with a Gaussian mixture and to characterize each data point with the derivatives over the model parameters. It appears that two main steps are crucial in such an approach [40]: the data distribution has to be accurately fitted by the Gaussian mixture and the provided second order statistics have to be carefully normalized. In this chapter, we propose to embed the Fisher representation in an end-to-end

trainable network by concentrating on these two steps.

First, a Gaussian Mixture Model (GMM) seems not to be well adapted to the deep local features since they are lying in a very high dimensional space and the deep local features require too many Gaussians to be accurately modeled [45]. Liu et al. proposed a smart solution to overcome this problem which consists in sampling the center of each Gaussian from a subspace and therefore benefiting from an infinite number of Gaussians to fit in the data distribution [45]. The authors showed that this problem can be solved by a classical sparse coding method. Unfortunately, their approach cannot take advantage of end-to-end training of the feature extraction, the pooling and the classification layers. To cope with this problem, we propose in this chapter, to make use of the deep sparse coding module introduced in the work [146].

Second, a recent study has shown that the normalization of the second order statistics has a strong impact on the classification performance [46]. The authors proposed in particular to use a square-root matrix normalization combined with element-wise square-root and l_2 normalization for bi-linear pooling. Unfortunately, unlike the bi-linear pooling used in the work [46], our Fisher representation does not provide a square matrix, thus rendering the solution from the work [46] unusable. Thus, in this chapter, we propose to adapt the square-root matrix normalization to non square matrices and to embed this original module in a deep network.

By combining these two main contributions, we propose an original end-to-end trainable deep network that:

- extracts accurate features from the images ;
- pools them into a deep Fisher representation ;

- normalizes these statistics.

By backpropagating the gradient of the classification loss, we are able to make all these modules collaborate, with the sole objective of improving the performance of the classification task. Experimental tests on three different datasets and three different backbone architectures showed that our solution outperforms many alternatives.

4.2 Related work

4.2.1 Orderless pooling

As detailed in Chapter 2, orderless pooling was widely used before the emergence of the CNN-based solutions. The most popular approaches were based on bags of visual words (BOW) [38], VLAD [39] or Fisher Vectors [40]. Inspired by these early methods, some works evaluated the Fisher vectors or VLAD from deep features for texture or image classification [31, 41]. They showed improvements over the SIFT-based counterparts but, in their workflow, the dictionary or Gaussian mixture model were learned independently from the deep features and from the classifier, providing opportunities for significant improvements.

As a consequence, more recent works focused on embedding orderless pooling in deep networks to allow end-to-end training. Passalis and Tefas inserted a Bag-of-Features pooling in deep neural networks thanks to radial basis function neurons [65]. The output of the pooling module is a histogram of the visual words (0^{th} order statistics) learned on the training set.

Instead of counting the occurrences of the visual words in one image, VLAD-based approaches aggregate the residuals between the local features

and their nearest visual words (1^{st} order statistics). NetVLAD is the first network that solves this task with an end-to-end training [42] and was later improved by Zhang et al. with Deep Ten [43]. It has been shown that first order statistics are more accurate to characterize images in classification tasks. Furthermore, the Fisher vectors go further by using first and second order statistics. Deep FisherNet is an embedded implementation of the GMM Fisher vector [44]. Lin et al. introduced NetFV which extends NetVLAD by appending the second order statistics [63]. The main disadvantage of all these approaches is that they rely on a limited number of codewords or Gaussian centers, which prevents accurate modeling of the data distribution in high-dimensional deep feature spaces [45].

One interesting solution to cope with this problem was proposed by Li et al. [66]. The authors compute Fisher vectors from a Mixture of Factor Analyzers (MFA), instead of the classical GMM. Their solution is embedded in a deep network which is trainable end-to-end. The idea of MFA is to approximate the data manifold by low dimensional linear spaces and, in this sense, is similar to the idea of sparse coding [45]. Nevertheless, even if the MFA module is embedded in a deep network, the authors showed that an accurate initialization of the weights of the network is required to obtain good performance. This initialization consists in running an Expectation-Maximization algorithm on the set of local features that have to be saved in memory. Furthermore, it appears that this second order representation has high computation costs and requires a high number of parameters to learn. Moreover, it occupies a very large memory space (500k dimensions which is more than the image itself) [147].

Another group of second-order pooling works is based on bilinear coding, such as BCNN [63] which is also an end-to-end trainable network. It aggre-

gates feature vectors by sum-pooling their outer products. Since this pooled representation always has a large size, SMSO [79] proposed to compress the bilinear pooled features while simultaneously improving the classification performance.

Our method is inspired by the work of Liu et al. [45] detailed in the next section. More recently, they also proposed an improved version of their work , called HSCFV [148]. It uses two dictionaries to code input features and consequently, doubles the dimension size of the Fisher vector. Nevertheless, their approach is not embedded in a deep CNN for end-to-end training.

Our method combines all the benefits of these previous solutions:

- it is embedded in an end-to-end trainable network;
- it samples an infinite number of Gaussian centers from a learned subspace;
- it does not require any heavy computation or storage to initialize the weights.

4.2.2 Normalization

As a post-processing step after orderless pooling, normalization plays an important role in improving the performance of the classifier. Perronin et al. observed that the representation pooled by Fisher vectors is degraded by burstiness issues where discriminative but relatively rare visual features are overwhelmed by those that are more frequent [58]. To alleviate this problem, some papers propose element-wise signed square rooting and L2-normalization [58, 59]. This normalization combination is also widely adopted in several successive orderless pooling works [42, 63, 45].

Besides the burstiness issue, Lin and Maji argued that the output of bilinear pooling should be normalized by matrix-logarithm functions in order to preserve the distances between elements in the manifold [46]. Such normalization has been applied successfully with linear classifiers for semantic segmentation and image classification [80, 72, 81] . The logarithm scales the eigenvalues in the Singular Value Decomposition (SVD) of a Symmetric Positive Definite (SPD) matrix A such as $\log(A) = Q \log(\Sigma) Q^T$. Unfortunately, as the SVD decomposition is computed inefficiently on GPUs [46], it slows down the network's inference speed. Nevertheless, Lin and Maji proposed a fast alternative approach with comparable performance. It is based on a variant of Newton iterations [46]. This solution approximates the matrix square-root and can be embedded in a network that can be trained end-to-end.

Unfortunately, this approach is exclusively designed for SPD matrices, such as the outputs of the bilinear pooling, and it cannot directly be applied to our Fisher representations that are rectangular and non symmetric matrices. Therefore, we propose, in this chapter, a new normalization step for such second order statistics matrices. This normalization can also be embedded in a deep network.

4.3 Deep sparse coding Fisher vector

Fig. 4.1 illustrates the complete workflow of our solution whose successive steps are detailed in this section. Our network starts with a pre-trained backbone of convolutional layers, on top of which a dictionary sparse coding with a LISTA module is applied. Then, the Fisher vectors are extracted from these features and normalized before being sent to a fully connected layer.

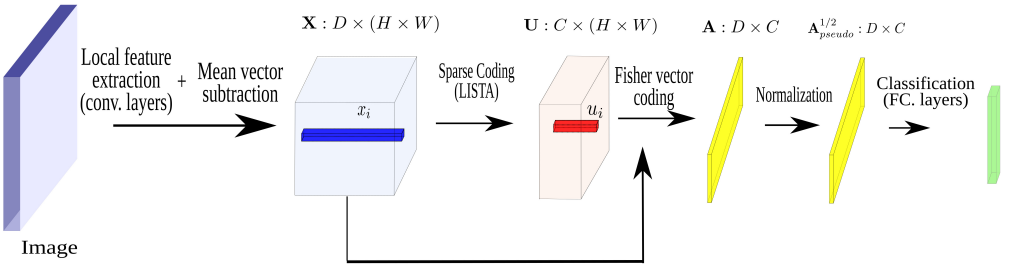


Figure 4.1 – Workflow of the proposed solution.

4.3.1 Mean Vector Subtraction

As shown in Fig. 4.1, the network backbone provides a feature map $\mathbf{X} \in \mathbb{R}^{D \times H \times W}$ where D,H,W are its depth, height and width. As only orderless pooling is discussed in this chapter, the feature map is reshaped to a 2D matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$ where $N = HW$. Therefore, its spatial arrangement is omitted and the column i of the matrix represents a D-dimensional local feature vector $\mathbf{x}_i = \mathbf{X}_{:,i}$. Before applying the proposed dictionary encoding, it is worth mentioning that these local feature vectors $\mathbf{x}_i$ are centered to zero mean:

$$\mathbf{x}'_i = \mathbf{x}_i - \frac{1}{N}\left(\sum_{i=1}^{N} \mathbf{x}_i\right). \tag{4.1}$$

Thanks to this pre-processing, the following sparse coding process does not need to waste its effort to first estimate this global offset, and thus spares more resources (i.e. atoms in the dictionary) on more accurate feature space modeling. We noticed that the classification performances are improved when adding this step (see more details in the section 4.5.4). **For simplicity, in the rest of this chapter, we omit the prime on the $\mathbf{x}_i$ so feature vector $\mathbf{x}_i$ and feature map $\mathbf{X}$ are always the results after mean vector subtraction.**

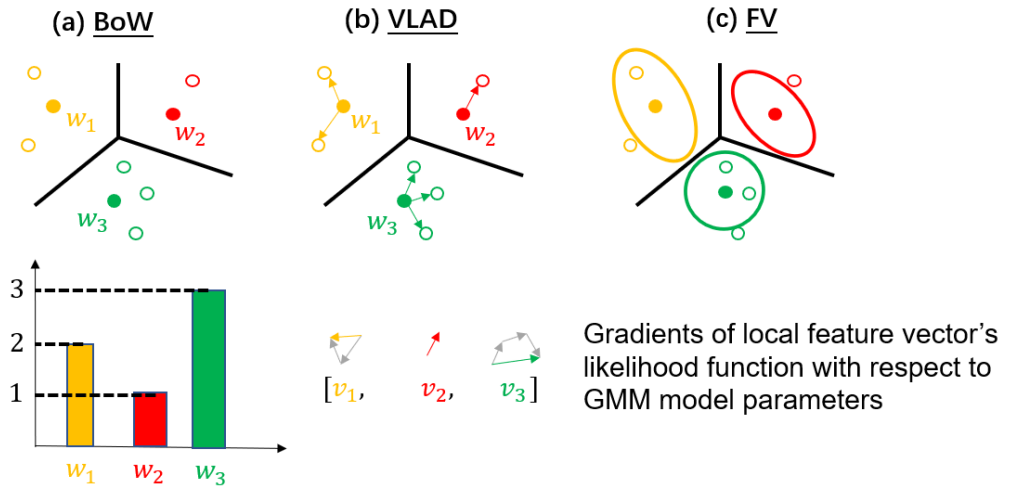


Figure 4.2 – Illustrating the ideas of BoW, VLAD and FV. (a) BoW: counting the number of local feature vectors (hollow circles) around their respective nearest codeword (filled circles). (b) VLAD: sum pooling residual vectors between each codeword and its assigned local feature vectors. (c) FV: gradient vectors which update Gaussian Mixture Model to better fit local feature space.

4.3.2 Codewords based approaches

Given a group of local feature vectors $\mathbf{X}$ extracted by the network, codewords based approaches encode them into an aggregated representation in 2 steps. First step is codebook generation procedure in which, with local features of training images, a set of prototype features, i.e. codewords, is learned to model feature space. Second step is feature encoding which maps each local feature vector $\mathbf{x}_i$ into one or a number of codewords and calculates its encoded vector. Then the sum pooling of encoded vectors in one image is the aggregated representation.

In this section, we will briefly resume former state-of-the-art methods:

Bag of Words (BoW) [52, 38], Fisher score vector [40] and Vector of Locally Aggregated Descriptor (VLAD) [39] in order to better understand how this group of approaches works. Their respective ideas are also illustrated in the Fig. 4.2.

Bag of Words (BOW)

In the codebook generation procedure, BoW learns a set of codewords, called codebook $\{\mathbf{w}_k\}_{k=1}^K$, by $k$means clustering. Then in the second step, each local feature vector $\mathbf{x}$ is assigned to its nearest codeword $\mathbf{w}_j$ and its encoded vector is a one-hot vector $\mathbf{v}$ where only its jth element is 1 and others are zero:

$$v_j = \begin{cases} 1, & \text{if } j = \underset{k}{\operatorname{argmin}} ||\mathbf{w}_k - \mathbf{x}|| \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

After the sum pooling of encoded vectors in one image, BoW representation is a histogram counting the number of local feature vectors assigned to each codeword (see also Fig. 4.2a)

Fisher Vector encoding and GMM-based model

As one of the most common Fisher vector encoding approaches, Gaussian Mixture Model based Fisher Vector Coding (GMMFVC) first learns a codebook, called Gaussian Mixture Model (GMM) and estimates a probability density function:

$$P(\mathbf{x}|\boldsymbol{\theta} = \{w_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k\}_{k=1}^K) = \sum_{k=1}^{K} w_k \mathcal{N}_k(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k), \tag{4.3}$$

where $\boldsymbol{\theta}$ is a set of the model's parameters and consists of each kth Gaussian component $\mathcal{N}_k$ parameters: Gaussian center vector $\boldsymbol{\mu}_k$, diagonal vector

of Gaussian covariance matrix $\boldsymbol{\sigma}_k$ and the mixture weight w_k. Compared to $k$mean clustering, GMM considers not only cluster centers but also covariances which describe the shape of clusters (see also Fig. 4.2c)

In the feature encoding step, we compute gradients of the likelihood function with respect to GMM model parameters $\boldsymbol{\theta}$ on each data points. Then, we sum up the gradients on the data points and concatenated the results into a vector $\mathbf{g}_{\boldsymbol{\theta}}^{\mathbf{X}}$:

$$\begin{aligned}\mathbf{g}_{\boldsymbol{\theta}}^{\mathbf{X}} &= [g_{w_k}^{\mathbf{X}}, \mathbf{g}_{\boldsymbol{\mu}_k}^{\mathbf{X}}, \mathbf{g}_{\boldsymbol{\sigma}_k}^{\mathbf{X}}]_{k=1}^{K}, \\ &= [\sum_{i=1}^{N} \nabla_{w_k} \log P(\mathbf{x}_i|w_k), \sum_{i=1}^{N} \nabla_{\boldsymbol{\mu}_k} \log P(\mathbf{x}_i|\boldsymbol{\mu}_k), \sum_{i=1}^{N} \nabla_{\boldsymbol{\sigma}_k} \log P(\mathbf{x}_i|\boldsymbol{\sigma}_k)]_{k=1}^{K}.\end{aligned} \tag{4.4}$$

This output $\mathbf{g}_{\boldsymbol{\theta}}^{\mathbf{X}}$, called Fisher Score, is related to the direction in which the parameters $\boldsymbol{\theta}$ should be updated in order to make the model globally better fit the feature sparce after adding $\mathbf{x}_i \in \mathbf{X}$ [40] in one image.

Note that the Fisher Vector representation is the Fisher Score $\mathbf{g}_{\boldsymbol{\theta}}^{\mathbf{X}}$ scaled by inverse square root of Fisher Information Matrix (FIM). Since this FIM plays not a very significant role in the image classification task [58], usually the Fisher Score representation is used to describe an image. **Hence, in the rest of the chapter, the Fisher Vector refers to the Fisher Score**.

Vector of Locally Aggregated Descriptor (VLAD)

The VLAD can be considered as a simplified version of Fisher vector. Its codebook generation is done by $k$means clustering rather than GMM. And each local feature vector $\mathbf{x}$ is encoded into a concatenation of residual vectors: $[\mathbf{v}_1, ..., \mathbf{v}_K]$. In the concatenation, only the difference to its nearest codeword $\mathbf{w}_j$ is recorded into the corresponding jth vector $\mathbf{v}_j$ and other elements are

zero.

$$\mathbf{v}_j = \begin{cases} \mathbf{x} - \mathbf{w}_j, & \text{if } j = \underset{k}{\operatorname{argmin}}(||\mathbf{w}_k - \mathbf{x}||) \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

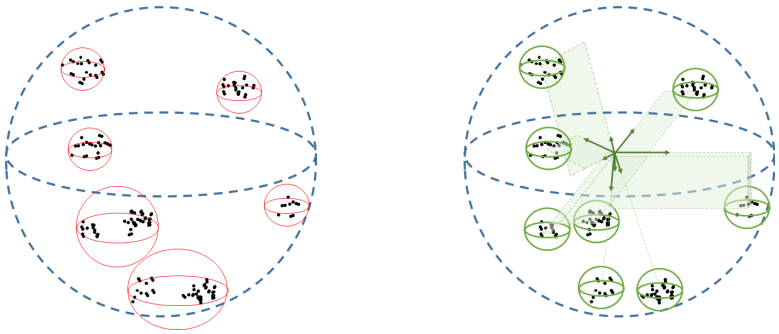
Lastly, the VLAD representation for one image is obtained by accumulating the concatenation of different local feature vectors(see also Fig. 4.2b)

Problematic

Among the three approaches, Fisher Vectors (FV) are the most promising because VLAD and BOW are special cases of FV. BoW can be considered as the gradient w.r.t the mixture weight parameters and VLAD is the gradient w.r.t the model's Gaussian centers. Although FV is performant, as each Gaussian component is only able to cover a local area of the whole feature space, the number of Gaussian components are related to the size of the feature space. Deep learning features in the last layers always have high dimensions. Their feature space is so large that numerous Gaussian components are needed. This consequently makes the code generation procedure and the following classification difficult, due to the increase of the number of learned parameters. In the following section, we will introduce an approach which can effectively increases the number of Gaussians.

4.3.3 From subspace sampling to sparse coding

In order to increase the number of Gaussians that model the distribution of the data, we take advantage of the idea from [45] that samples the Gaussian mean vectors in a subspace spanned by a set of bases. Each mean vector is coded in this "dictionary" $\mathbf{B} \in \mathbb{R}^{D \times C}$ with a code $\mathbf{u} \in \mathbb{R}^C$ drawn from a zero-



Gaussian Mixture *Sparse Coding*

Figure 4.3 – Some data in a high dimensional space (illustrated by the sphere). Left: With GMM the data distribution is not well fitted because of the limited number of Gaussians. Right: With Sparse Coding, the Gaussian centers are coded sparsely in an adapted basis (green arrows) allowing to create unlimited number of Gaussians and so to fit better in the data distribution. The sparsity is illustrated by the low number of basis required to code each center position (lines, planes or parallelograms).

mean Laplacian distribution (to enforce sparsity). Then, each local feature vector $\mathbf{x}$ extracted from the images, associated with the code $\mathbf{u}$, is drawn from a Gaussian distribution $\mathcal{N}(\mathbf{Bu}, \boldsymbol{\sigma})$ centered on $\mathbf{Bu}$. Fig. 4.3 illustrates this approach.

Assuming that the covariance matrix is diagonal, and that its diagonal elements are constant σ, then using pointwise maximum to approximate the integral of the distribution, Liu et al. showed that the logarithm of the likelihood of $\mathbf{x}$ can be estimated as [45]:

$$log(P(\mathbf{x}|\mathbf{B})) = \min_{\mathbf{u}} \frac{1}{\sigma^2}||\mathbf{x} - \mathbf{Bu}||_2^2 + \lambda||\mathbf{u}||_1, \tag{4.6}$$

where λ is the scale parameter of the Laplacian distribution of $\mathbf{u}$.

Interestingly, this equation represents the classical problem of sparse cod-

ing. Liu et al. proposed to use an off-the-shelf sparse coding solver to learn the dictionary $\mathbf{B}$ and infer the code $\mathbf{u}$ [45]. Obviously, making use of such independent solver is a good solution to minimize the reconstruction error of $\mathbf{x}_i$ with a sparse code, but it neglects the main goal which is to improve the performance in the classification task.

Hence, we propose in the next section to embed a sparse coding module in a deep neural network that is trained end-to-end. The main advantage of such an approach is that it is learning a dictionary and sparse codes that are accurate to discriminate the different categories in the current dataset.

4.3.4 Embedding sparse coding with LISTA

Our aim is to find a solution for the following equation:

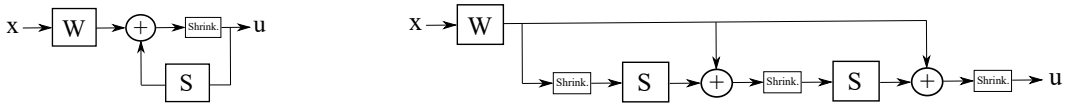
$$\min_{\mathbf{u}} f(\mathbf{u}) + \lambda||\mathbf{u}||_1 \tag{4.7}$$

where $f(\mathbf{u}) = ||\mathbf{x} - \mathbf{B}\mathbf{u}||_2^2$, $\mathbf{x}$ is a data point, $\mathbf{B}$ the dictionary and $\mathbf{u}$ the sparse code of $\mathbf{x}$.

One way to solve this equation is to resort to an Iterative Shrinkage Thresholding Algorithm (ISTA) [149] that iteratively approximates the solution with:

$$\mathbf{u}^k = \mathcal{T}_{\lambda t_k}(\mathbf{u}^{k-1} - t_k \nabla f(\mathbf{u}^{k-1})), \tag{4.8}$$

where $\mathcal{T}_\alpha(.)$ is a component-wise vector shrinkage function such that $[\mathcal{T}_\alpha(\mathbf{h})]_i = (|h_i| - \alpha)_+ sign(h_i)$, t_k is the step size at iteration k and ∇ is the gradient operator.



ISTA block diagram *Learned ISTA block diagram*

Figure 4.4 – Block diagrams of ISTA and LISTA. LISTA is an unfolded version of ISTA (3 iterations here).

Evaluating the gradient of $f(u)$ defined above, we get:

$$\begin{aligned}\mathbf{u}^k &= \mathcal{T}_{\lambda t_k}(\mathbf{u}^{k-1} - 2t_k\mathbf{B}^T(\mathbf{B}\mathbf{u}^{k-1} - \mathbf{x})),\\ &= \mathcal{T}_{\lambda t_k}((\mathbf{I} - 2t_k\mathbf{B}^T\mathbf{B})\mathbf{u}^{k-1} + 2t_k\mathbf{B}^T\mathbf{x}),\\ &= \mathcal{T}_{\lambda t_k}(\mathbf{S}\mathbf{u}^{k-1} + \mathbf{W}\mathbf{x}),\end{aligned}$$

where $\mathbf{S} = \mathbf{I} - 2t_k\mathbf{B}^T\mathbf{B}$ and $\mathbf{W} = 2t_k\mathbf{B}^T$.

As mentioned in the work [146], this equation can be illustrated as a recurrent block diagram as in Fig. 4.4, left. Fortunately, Gregor and Lecun proposed a fast approximation of ISTA called Learned ISTA (LISTA) [146]. This is an unfolded version of ISTA, with a fix number of iterations K, that can be plugged into a neural network to provide a sparse code $\mathbf{u} = \mathbf{u}^{k=K}$ ($K = 3$ in Fig.4.4, right). Embedding this LISTA module in our CNN is an effective solution to learn a dictionary and sparse codes that help to discriminate between the categories of the current task.

4.3.5 Dictionary based Fisher coding

When a classical GMM is used to model the data distribution, the Fisher code is based on the partial derivatives of the posterior probabilities with respect to the weights, the mean and the standard-deviation parameters of the model [40]. In our case, the model is based on a learned dictionary.

Moreover, we use a particular Fisher coding, as in the work [45], evaluated as the partial derivative of the log probability of the local features with respect to the dictionary itself:

$$\frac{\partial log(P(\mathbf{x}|\mathbf{B}))}{\partial \mathbf{B}} = \frac{\partial \frac{1}{\sigma^2}||\mathbf{x}-\mathbf{B}\mathbf{u}||_2^2 + \lambda||\mathbf{u}||_1}{\partial \mathbf{B}} = (\mathbf{x}-\mathbf{B}\mathbf{u})\mathbf{u}^T, \tag{4.9}$$

where $\mathbf{u} = \mathbf{u}^K$, i.e. output of K-iterations LISTA. Then, the Fisher Score representation of feature map $\mathbf{X}$ can be calculated through:

$$\mathbf{A} = \frac{1}{N}\left(\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{B}\mathbf{u}_i)\mathbf{u}_i^T\right), \tag{4.10}$$

where $\mathbf{u}_i$ is the sparse code of ith local feature vector $\mathbf{x}_i$ in the feature map $\mathbf{X}$.

This module is very easy to insert in our deep network and provides the pooled features from the input image. These features are then sent to the last fully connected layers for classification. All these modules are constituting our CNN which can be trained end-to-end (see Fig. 4.1).

4.4 Fisher vector normalization

As mentioned earlier, the second order statistics tend to excessively emphasize very few coordinates, ignoring potential discriminative features [46]. To cope with this problem, many normalization solutions have been proposed. In this chapter, we take advantage of the approach proposed in the paper [46] to normalize our Fisher vectors. Below, we first detail the solution [46] and then, explain its extension to non-square matrices.

4.4.1 Bilinear square matrix normalization

Assuming that the network backbone provides a feature map $\mathbf{X} \in \mathbb{R}^{D\times N}$, where N and D are its spatial resolution and depth. This set of local feature

vectors $\mathbf{x}_i \in \mathbb{R}^D$ can be orderless pooled into a global feature vector by using bilinear pooling [77]. The output of the bilinear pooling is evaluated as:

$$\mathbf{A} = \frac{1}{N}\left(\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T\right). \tag{4.11}$$

$\mathbf{A}$ is a $(D \times D)$ symmetric positive definite (SPD) matrix.

While element-wise square-root normalization helps in improving the performance of the complete framework, Lin and Maji have shown that the results can be further boosted by applying a spectral normalization, i.e. scaling the eigenvalues of the associated covariance matrix [46]. One way to do that is to transform the matrix $\mathbf{A}$ to its square-root $\mathbf{A}^{1/2} = \mathbf{Q}\mathbf{\Sigma}^{1/2}\mathbf{Q}^T$, where $\mathbf{A} = \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^T$ is the singular value decomposition (SVD) of $\mathbf{A}$.

However, the computation of SVD is poorly supported on GPUs and Lin et al. [46] suggest applying a variant of the Newton's method to solve $f(\mathbf{Z}) = \mathbf{Z}^2 - \mathbf{A} = 0$ where one iteration k is:

$$\mathbf{Y}_{k+1} = \frac{1}{2}\mathbf{Y}_k(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k), \mathbf{Z}_{k+1} = \frac{1}{2}(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k)\mathbf{Z}_k \tag{4.12}$$

By initializing $\mathbf{Y}_0 = \mathbf{A}$ and $\mathbf{Z}_0 = \mathbf{I}$, $\mathbf{Y}_k$ and $\mathbf{Z}_k$ converge to $\mathbf{A}^{1/2}$ and $\mathbf{A}^{-1/2}$ in very few iterations (even one) and requires only matrix multiplications (no inverse).

The process is illustrated in Fig. 4.5 and 4.6.

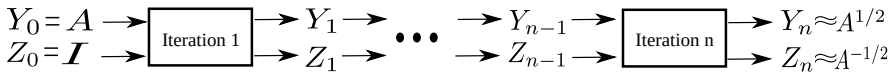


Figure 4.5 – Overview of the Newton's method .

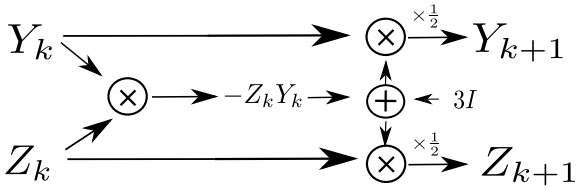


Figure 4.6 – Workflow of the iteration k of the Newton's method.

4.4.2 Matrix Normalization for Fisher score representation

Going back to our Fisher representation (eq. 4.10), we start with a second-order matrix expressed as:

$$\mathbf{A} = \frac{1}{N}\left(\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{B}\mathbf{u}_i)\mathbf{u}_i^T\right), \tag{4.13}$$

where $\mathbf{B} \in \mathbb{R}^{D\times C}$ is a dictionary having C atoms and $\mathbf{u}_i \in \mathbb{R}^C$ is the sparse code of centered $\mathbf{x}_i$ by Eq. 4.1.

From Eq. 4.10, we notice that $\mathbf{A} \in \mathbb{R}^{D\times C}$ is not square and not symmetric, so the normalization presented above for bilinear square matrices cannot be directly applied to the Fisher representation. Indeed, since $\mathbf{A}$ is not SPD, its SVD is given as $\mathbf{A} = \mathbf{Q}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{Q} \neq \mathbf{V}$ and where $\mathbf{\Sigma} \in \mathbb{R}^{D\times C}$ is not square.

In order to apply spectral normalization, we are looking for a so-called pseudo square root matrix $\mathbf{A}^{1/2}_{pseudo}$ defined as:

$$\mathbf{A}^{1/2}_{pseudo} = \mathbf{Q}\mathbf{\Sigma}^{1/2}_{pseudo}\mathbf{V}^T, \tag{4.14}$$

where $\mathbf{\Sigma}^{1/2}_{pseudo}$ is calculated by square rooting the diagonal elements of $\mathbf{\Sigma}$ (there is no matrix $\mathbf{\Sigma}^{1/2}$ such that $\mathbf{\Sigma} = \mathbf{\Sigma}^{1/2}\mathbf{\Sigma}^{1/2}$).

Likewise [46], in order to avoid SVD computation, we turn to Newton's method to evaluate such a $\mathbf{A}^{1/2}_{pseudo}$ matrix. As Newton's method only accepts

SPD square matrix as input, $\mathbf{A}$ is first transformed into the square SPD matrix $\mathbf{D}$ evaluated as:

$$\begin{aligned}\mathbf{D} &= \mathbf{A}^T\mathbf{A}, \\ &= \mathbf{V}\mathbf{\Sigma}^T\mathbf{Q}^T\mathbf{Q}\mathbf{\Sigma}\mathbf{V}^T, \\ &= \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T,\end{aligned} \tag{4.15}$$

which is independent of $\mathbf{Q}$.

Since $\mathbf{\Sigma}$ is asymmetric, we introduce a temporary matrix $\mathbf{H} = [\mathbf{I}_C|0]^T \in \mathbb{R}^{D\times C}$ and decompose $\mathbf{\Sigma}$ as:

$$\mathbf{\Sigma} = \mathbf{H}\tilde{\mathbf{\Sigma}}, \tag{4.16}$$

where $\tilde{\mathbf{\Sigma}}$ is a square diagonal matrix and $\mathbf{I}_C$ is a $C \times C$ identity matrix.

Hence, Eq.4.15 can be derived into:

$$\begin{aligned}\mathbf{D} &= \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T, \\ &= \mathbf{V}\tilde{\mathbf{\Sigma}}^T\mathbf{H}^T\mathbf{H}\tilde{\mathbf{\Sigma}}\mathbf{V}^T \\ &= \mathbf{V}\tilde{\mathbf{\Sigma}}^2\mathbf{V}^T.\end{aligned} \tag{4.17}$$

This equation is the SVD of the matrix $\mathbf{D}$.

Feeding the previous Newton workflow with $\mathbf{D}$ and an identity matrix, we obtain $\mathbf{D}^{1/2} = \mathbf{V}\tilde{\mathbf{\Sigma}}\mathbf{V}^T$ and $\mathbf{D}^{-1/2} = \mathbf{V}\tilde{\mathbf{\Sigma}}^{-1}\mathbf{V}^T$ and feeding again this workflow with $\mathbf{D}^{1/2}$ and an identity matrix, we obtain $\mathbf{D}^{1/4} = \mathbf{V}\tilde{\mathbf{\Sigma}}^{1/2}\mathbf{V}^T$ and $\mathbf{D}^{-1/4} = \mathbf{V}\tilde{\mathbf{\Sigma}}^{-1/2}\mathbf{V}^T$.

Finally, we have access to $\mathbf{A}^{1/2}_{pseudo}$ thanks to:

$$\begin{aligned}
\mathbf{A}\mathbf{D}^{-1/4} &= \mathbf{Q}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\tilde{\mathbf{\Sigma}}^{-1/2}\mathbf{V}^T, \\
&= \mathbf{Q}\mathbf{\Sigma}\tilde{\mathbf{\Sigma}}^{-1/2}\mathbf{V}^T, \\
&= \mathbf{Q}\mathbf{H}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{\Sigma}}^{-1/2}\mathbf{V}^T \\
&= \mathbf{Q}\mathbf{H}\tilde{\mathbf{\Sigma}}^{1/2}\mathbf{V}^T \\
&= \mathbf{Q}\mathbf{\Sigma}^{1/2}_{pseudo}\mathbf{V}^T \\
&= \mathbf{A}^{1/2}_{pseudo}.
\end{aligned} \tag{4.18}$$

Hence, without any SVD computation, this solution allows us to spectrally normalize a non SPD matrix $\mathbf{A}$ as $\mathbf{A}^{1/2}_{pseudo}$ very efficiently. Furthermore, this workflow can be easily embedded in a end-to-end trainable deep network.

4.5 Experiments

In order to show that our solution generally helps the classification performance, we run experiments on three datasets, which present strong differences in terms of tasks and scales. The three datasets and their experimental settings are detailed in section 4.5.1 and 4.5.2. Next, the training strategy of our network is shown in section 4.5.3. In section 4.5.4, results and comparisons are presented and discussed.

4.5.1 Datasets

We run experiments on three datasets to demonstrate the versatility of our solution for different image classification tasks. Note that we always make use of official training-test splits released with the datasets.

Table 4.1 – Pretrained backbone network's layer[a] after which our DPM is plugged.

	MIT-67			CUB-200			MINC-2500
	AlexNet	VGG-16	ResNet-50	AlexNet	VGG-16	ResNet-50	ResNet-50
position	FC-6	conv5-3	conv5-3[b]	conv5	conv5-3	conv5-3[b]	conv5-3[b]

[1] *Activation function included*

[2] *name of the network's residual block*

As introduced in Chapter 2, the dataset MINC-2500 [34] contains 23 commonly-seen material categories and 2,500 images per category, and is a challenging large-scale dataset as material classes show great intra-class variability in the real-word environment (see image examples in Fig. 2.4). MIT Indoor 67 [100] is a medium but widely accepted benchmark for indoor scene classification task with 67 indoor categories and 100 images in each category. CUB-200-2011 [99] provides 11,788 images of 200 bird species and is considered as a fine-grained classification dataset because inter-class difference between bird species is subtle and sometime barely noticeable. In our experiments, although object bounding box and part annotation are available, only bird images are used as input and no more information is exploited.

4.5.2 Experimental settings

Deep Pooling Module (DPM) - Our DPM is composed of a 1×1 convolution layer, a LISTA module with two iterations, the Fisher encoding layer and normalization process which includes matrix normalization (see section 3.1), element-wise square root and l_2 normalization. Then DPM is followed by a fully connected layer with softmax activation for classification.

Depending on dataset scales and for fair comparison with other works,

we use different backbones and training strategies.

MIT-67 and CUB-200 2011 settings - We adopt the settings of state-of-the-art methods [79, 63]. The input image size is 448x448 and the backbone networks are either the pretrained VGG-D (a.k.a VGG-16) or Alexnet. Our DPM is plugged after ReLU function of the last convolutional layer or of Fully Connected layer(see Table 4.1). The 1×1 convolutional layer in the DPM keeps the input feature size and the sparse code in LISTA has 100 elements.

MINC-2500 settings - The network backbone is the pretrained ResNet-50 [37](see Table 4.1). With the 1×1 convolutional layer in the DPM, the input feature size is reduced into 128 and the size of sparse code in LISTA is 32. While training, we follow the data augmentation settings of [13]. The input image is resized to 256x256. 8% to 100% of the area of the of image is cropped with a random aspect ratio between $\frac{3}{4}$ and $\frac{4}{3}$ and the crop is resized to 224x224. Random horizontal and vertical flip with a probability of 50% is applied to each image. At test time, we use central crop of 224x224 as input.

4.5.3 Training details

In the training phase, three consecutive steps are conducted. First, we run a PCA on a small subset of feature vectors (around $10,000$) extracted from the backbone outputs and initialize the 1×1 convolutional layer of our DPM with these PCA parameters. Second, inspired by the work [150], we apply a warming-up process that consists in training our DPM and FC layer (while the backbone is frozen) with an objective function which is the sum of the cross-entropy loss and the sparse coding loss (see Eq. (4.6)). Finally, the whole network is fine-tuned end-to-end under the supervision of the sole cross-entropy loss.

The optimization algorithm is a gradient descent with a mini-batch size of 64, a weight decay of $5e^{-4}$ and a momentum of 0.9. The learning rate is 0.004 during the warming-up. During the end-to-end finetuning, it starts from 0.004 and is divided by 10 after every 40 epochs.

4.5.4 Results

Comparison with state-of-the-art - The top-1 classification accuracy of our approach and many alternatives are presented in Table 4.2. The results of the related works are extracted directly from the papers that are referenced in this table. Note that our CNN is trained on single-scale images while many state-of-the-art approaches are trained on multi-scales, so we have carefully selected the results that allows fair comparisons, but still some results in Table 4.2 are from multi-scale training (see Table 4.2's comments).

The methods called 'Off-the-shelf' use independent modules that are not fine-tuned together while the 'End-to-End' group contains approaches that use end-to-end trainable networks. We notice that the results provided by fine-tuned networks overall outperform those of the Off-the-shelf solutions. This shows that it is always better to make the modules work together to optimize the same loss instead of independently optimizing them. Besides end-to-end learning attribute, our approach is built upon Deep Fisher Score Representation via Sparse Coding (SCFVC) which produces more discriminative second-order pooled features than the classical Fisher vector or VLAD. The proposed effective combination of these two advantages make our method outperform the alternatives for all the datasets and backbones.

Solution analysis - In order to assess the added advantage of our different contributions, we conduct several experiments by varying the hyper-parameters and by ablating some modules. The study was done with the

Table 4.2 – Comparison of the classification accuracy (%) with closed-related alternatives on three datasets and three backbone architectures.

	Approaches	MIT AlexNet	MIT VGG16	MIT ResNet50	CUB AlexNet	CUB VGG16	CUB ResNet50	MINC ResNet50
Off-the-shelf	Baseline	58.4[151]			53.3[151]	60.4[63]		
	GMMFVC	64.3[45]	72.6[a][148]		61.7[45]	70.1[a][148]		
	SCFVC	68.2[45]	77.6[a][148]		66.4[45]	77.3[a][148]		
	HSCFVC		79.5[a][148]			80.8 [148]		
End-to-end	Baseline		64.51[79]	76.45[79]		70.4[63]	76.45[79]	79.1[79]
	Deep Ten			71.3[13]				80.4[13]
	NetVLAD					81.9[63]		
	NetFV		78.2[63]			79.9[63]		
	FisherNet		76.4[66]					
	MFAFVNet	69.89[b][66]	78.01[b][66]					
	B-CNN		77.6[63]			84.0[63]		79.05[79]
	SMSO		79.45[79]	79.68[79]		85.01[79]	85.77[79]	81.3[79]
	Our	70.15	80.22	84.85	76.8	84.28	84.47	81.5
	Our(+norm)	**70.60**	**81.24**	**85.52**	**77.49**	**85.8**	**87.38**	**81.8**

[a] *These methods were trained with VGG19 (not VGG16) with 2 scales, whereas the other approaches from the column are trained with a single scale.*

[b] *Since MFAFVNet works on patches and not on images, we have selected in [66] the results provided with the nearest patch scale from our settings (*160×160*).*

Table 4.3 – Ablation study of our workflow on the MIT-67 dataset

LISTA	Warming Up	Sub-Mean	Matrix Norm	Accuracy
				76.72
✓				77.16
✓	✓			80.22
✓	✓	✓		80.60
	✓	✓	✓	80.67
✓	✓	✓	✓	**81.24**

MIT-67 dataset and the VGG-16 network and the results are summarized in the Tables 4.3, 4.4 and 4.5.

In Table 4.3, we start from the baseline network and consecutively add the proposed modules, in order to assess their individual impact on the results. When the LISTA module is not in the network, it is replaced by a 1×1 convolutional layer providing the codes u_i.

As introduced in the section 4.3, warming up is one of the three steps in the training phase. The goal is to train the newly added DPM and FC layers before fine-tuning the whole network. According to Table 4.3, this warming up step boosts the performance from 77.16% to 80.22%, showing that an accurate initialization is important for our DPM and classifier.

Likewise, we notice that the proposed matrix normalization and mean subtraction also provide improvements of the results. For example, the matrix normalization increases the accuracy from 80.60% to 81.24%.

Finally, the impact of the LISTA module is measured with two different tests. Starting from the baseline and adding LISTA improves the results from 76.72% to 77.16% and adding LISTA to the whole process helps to increase

Table 4.4 – Impact of the number of iterations in LISTA on the accuracy.

LISTA	linear	i=1	i=2	i=3	i=4	i=5
Accuracy	80.67	81.04	81.24	81.34	81.04	80.30

Table 4.5 – Impact of the dictionary size on the accuracy.

N° codewords	50	100	200	300	400	512
Accuracy	80.37	81.24	80.90	80.00	80.22	80.22

from 80.67% to 81.24%.

LISTA is composed of several iterations which unfold ISTA loop process to solve a sparse coding problem. We investigate the performance of the LISTA across different numbers of iterations from 0 to 5 in Table 4.4, where 0 means that the LISTA module is replaced by a 1×1 convolutional layer. In this Table, we notice that 2 or 3 iterations provide the best performance. After 3 iterations, the results start decreasing. Our intuition is that too many iterations of LISTA produce better sparse codes but neglect the aim of the whole process which is to get perfect classification performances. For all the tests conducted in this chapter, the default value of 2 iterations provides good results.

We also conducted an analysis on the number of codewords required in the LISTA module. We measure the classification accuracy for a range of codeword numbers from 50 to 512 in Table 4.5. Note that the dimension 512 is the same as the last convolutional layer's output dimension. We notice that, due to overfitting, when the number of codewords is higher than 100, lower accuracy is observed.

These experimental results have shown that the proposed approach outperforms many alternatives for different classification tasks. They also help to understand the impact of each contribution on the whole process.

4.6 Conclusion

Classical Fisher vectors are strong pooled representations for classification but require many Gaussians when applied on high-dimensional deep features. One way to cope with this problem is to code sparsely the Gaussian centers in an adapted basis in order to increase the number of available Gaussians and better fit in the data distribution. In this chapter, we have shown that this coding can be embedded in a deep network allowing to adapt the basis and sparse code such that they optimize the classification performance. To further improve the fisher representation power, a new matrix normalization and mean subtraction have been implemented into our approach. We have also proposed a training strategy that can easily but effectively initialize the network parameters before finetuning. With the support of the end-to-end learning and a powerful Fisher score representation, our method outperforms the tested alternatives on three different datasets.

Like almost all the previous GMM-based approaches, our approach uses only diagonal covariance matrices for the Gaussians to fit the data. As future works, we would like to learn non-diagonal matrices in order to further refine the training data fitting.

Today, many works are exploiting bi-linear pooling in many different tasks, but since these second order statistics lie in very high-dimensional space, the recent trend consists in compacting these features. Nevertheless, normalizing the matrices of compact bi-linear pooled features is not easy and

not efficient. We think that the normalization proposed in this chapter can help in normalizing compact bi-linear representations. This is the topic of our current research.

Chapter 5

Conclusion, Limitations and Perspectives

5.1 Conclusion

Material image classification is one crucial task in computer vision because it is involved in many real applications such as robotics or automatic waste sorting, and because it can help in many other problems such as fine-grained image classification. It consists in correctly classifying images with target material from one given category. In the beginning of 2010s, thanks to their superior performances, the deep convolutional neural networks (CNN) arise and become a promising tool to solve many computer vision problems, including image classification. Deep networks have also been introduced into material classification. By simply transferring a network pretrained on a large-scale image classification task, better accuracy is achieved than former state-of-the-arts. However, unlike object recognition, classifying materials require some specific processing. In this thesis, we mainly focus on two peculiarities of material images:

- Large areas of material surface are visually ambiguous and discriminative information mainly resides in tiny areas, e.g. Fig 1.4.

- In contrast to objects, material instances in one image show spatial orderless arrangement, e.g. Fig 1.6.

Thus, directly transferring the CNN architectures, initially designed for objects classification, is inevitably sub-optimal. Specifically, referring to the two properties described above, we identified two drawbacks of actual CNNs:

- For one of the most popular architecture of CNNs, called ResNet, its Global Average Pooling aggregates local feature vectors. This operations overwhelms relevant but small features by ambiguous and numerous local regions, consequently producing less discriminative global features.

- We have demonstrated classification improvement by using more sophisticated orderless pooling, like classical Fisher Vectors. Unfortunately, inaccurate estimation about data distribution occurs in the context of high-dimensional deep features, because the number of Gaussians in the Gaussian Mixture Model is limited.

In this PhD we contributed to solve them in the following ways:

- In order to identify features coming from ambiguous or discriminative regions, we proposed to add in a classical network a branch which predicts confidence values, deduced from the True Class Probability (TCP), associated with each local feature vector. Then, we exploit these TCP values to filter out ambiguous local features, i.e. the ones associated with lower confidences. Consequently, a more powerful representation is obtained after applying the Global Average Pooling to the

rest of the local features. According to experimental results on three real-world material datasets, our approach outperforms other classical models on the criteria of both classification accuracy and output probability calibration.

- To fix the issue of limited Gaussians, we noticed that sparsely coding the Gaussian centers in an adapted basis is a promising way to increase the number of available Gaussians because it can better fit the data distribution. In Chapter 4, to take one more step than previous works, we implemented this coding process into a deep learning network. This allows the basis and sparse code to be optimized for improving the classification performance. Furthermore, a new normalization and mean subtraction were also embedded into our approach. We demonstrated that our approach further enhances the discrimination power of the produced Fisher score representations. In practice, we also proposed a training strategy which facilitates initializing parameters in the embedded coding module. With the support of an end-to-end learning of these accurate Fisher score representations, our method outperforms state-of-the arts on three different datasets.

5.2 Limitations

For both solutions we have proposed, the training process is composed of two stages which is one weakness of these approaches. Indeed, for the local feature selection (Chapter 3), we need to train the classifier in the first stage and then freeze it during the second stage, while training the TCP predictor. Although this strategy produces stable TCP values, it is clear that separately training the classifier and the TCP prediction branch is slightly complicated.

Likewise, for our end-to-end Fisher vectors (Chapter 4), adding a warming-up training step before training the whole network is helpful as shown in Table 4.3. Compared to the related method in [66], our approach is obviously more adapted to deep CNN's training and needs less computational memory, but it still requires a two-stages training process to get the best results.

Choosing hyper parameters is another limitation for both solutions. In Chapter 3, the ambiguous local features are filtered out based on their predicted TCP values when they are lower than a threshold and this threshold has been empirically fixed. Although this value has not a strong impact on the results on the different tested datasets, it may not be optimal for every image. A similar situation happens in Chapter 4 with more hyper-parameters such as the size of the sparse codes, or the number of iterations in the LISTA module. Although we discussed their impact to the accuracy in Tables 4.4 and 4.5, in practice, facing a new task, it would be better if our method could automatically deduce them from the task rather than requiring them to be set manually.

Furthermore, our feature selection approach in the chapter 3 is more adaptive to small datasets where only the classifier is trained. Our approach is a good alternative way to realise the selection process when using pretrained feature extractor layers. Once these layers can learn themselves to select the most important feature vectors, our approach is less effective. On the contrary, our end-to-end Fisher vectors approach can be applied to either large or small datasets.

5.3 Perspectives

Bilinear pooling is also a second-order pooling to our Fisher vectors (Chapter 4). However, it does not require to fit the data distribution and as a result, it is a non-parametric pooling method. Despite its simplicity, it outperforms many state-of-the-art orderless pooling methods in several tasks of fine-grained images classification as shown in the work [69]. As discussed in Section 2.3, there are two research lines to improve bilinear pooling, where one is to obtain compact representation and the other one is its normalization. Incorporating these two techniques becomes a new research trend. The main limitation is that directly running matrix normalization on the compact bilinear pooling (CBP) features is proven infeasible [85]. Even if some solutions have been proposed [85, 86, 87, 88], the problem is only partially solved and strong limits remain.

For example, Gou et al. proposed to transform the input feature matrix to a "pseudo square-root", so that a classical bilinear pooling can be applied on it and directly provide a normalized bilinear representation [86]. Unfortunately, the computation of this matrix requires to apply the singular value decomposition (SVD) on the input matrix and we know that SVD is not well supported on GPU. One interesting future work could be to make use of our GPU-supported Newton method from Chapter 4 to efficiently calculate this pseudo square-root matrix.

In the case of material image classification with small datasets, another direction is to conceive a hybrid method which takes advantages of our two methods, as they respectively improve material classification at different steps of the workflow. Indeed, the selection of the most relevant features is a good way to decrease the computational load of the extraction of the

second order statistics which require to compute many outer products between the local feature vectors. Besides the gain of efficiency, the selection solution is expected to delete noisy local feature vectors, and thus helps to extract a more accurate global Fisher score representation.

Bibliography

[1] Lavanya Sharan, Ruth Rosenholtz, and Edward H. Adelson. Accuracy and speed of material categorization in real-world images. *Journal of Vision*, 14(10), 2014.

[2] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[3] Barbara Caputo, Eric Hayman, and P Mallikarjuna. Class-specific material categorisation. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1597–1604. IEEE, 2005.

[4] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[5] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[6] Charles Corbiere, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Perez. Addressing failure prediction by learning model confidence. In *Conference on Neural Information Processing Systems 2019 (NIPS 2019)*, 2019.

[7] B. Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962.

[8] Edward H Adelson. On seeing stuff: the perception of materials by humans and machines. In *Human vision and electronic imaging VI*, volume 4299, pages 1–12. International Society for Optics and Photonics, 2001.

[9] Raw food texture (rft), 2016. `http://projects.ivl.disco.unimib.it/minisites/rawfoot/index.php`.

[10] Chu He, Shuang Li, Zixian Liao, and Mingsheng Liao. Texture classification of polsar data based on sparse coding of wavelet polarization textons. *IEEE Transactions on Geoscience and Remote Sensing*, 51(8):4576–4590, 2013.

[11] Umasankar Kandaswamy, Donald A Adjeroh, and Moon-Chuen Lee. Efficient texture analysis of sar imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 43(9):2075–2083, 2005.

[12] Jia Xue, Hang Zhang, Kristin Dana, and Ko Nishino. Differential angular imaging for material recognition. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pages 764–773, 2017.

[13] Jia Xue, Hang Zhang, and Kristin Dana. Deep texture manifold for ground terrain recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2018.

[14] Adrien Depeursinge, Omar S Al-Kadi, and J Ross Mitchell. *Biomedical texture analysis: fundamentals, tools and challenges*. Academic Press, 2017.

[15] Loris Nanni, Alessandra Lumini, and Sheryl Brahnam. Local binary patterns variants as texture descriptors for medical image analysis. *Artificial intelligence in medicine*, 49(2):117–125, 2010.

[16] Mohammad Peikari, Mehrdad J. Gangeh, Judit Zubovits, Gina Clarke, and Anne L. Martel. Triaging diagnostically relevant regions from pathology whole slides of breast cancer: A texture based approach. *IEEE Transactions on Medical Imaging*, 35(1):307–315, 2016.

[17] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, 1981.

[18] B. Julesz and J. Bergen. Human factors and behavioral science: Textons, the fundamental elements in preattentive vision and perception of textures. 1987.

[19] Alan C. Bovik, Marianna Clark, and Wilson S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE transactions on pattern analysis and machine intelligence*, 12(1):55–73, 1990.

[20] Anil K Jain and Farshid Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern recognition*, 24(12):1167–1186, 1991.

[21] Mark R Turner. Texture discrimination by gabor functions. *Biological cybernetics*, 55(2):71–82, 1986.

[22] Song-Chun Zhu. Statistical modeling and conceptualization of visual patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):691–712, 2003.

[23] Bangalore S Manjunath and Wei-Ying Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on pattern analysis and machine intelligence*, 18(8):837–842, 1996.

[24] Jitendra Malik and Pietro Perona. Preattentive texture discrimination with early vision mechanisms. *JOSA A*, 7(5):923–932, 1990.

[25] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43(1):29–44, 2001.

[26] Kristin J Dana, Bram Van Ginneken, Shree K Nayar, and Jan J Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG)*, 18(1):1–34, 1999.

[27] Michael Weinmann, Juergen Gall, and Reinhard Klein. Material classification based on training data synthesized using a btf database. In *European Conference on Computer Vision*, pages 156–171. Springer, 2014.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[30] Patrick Wieschollek and Hendrik Lensch. Transfer learning for material classification using convolutional networks. *arXiv preprint arXiv:1609.06188*, 2016.

[31] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3828–3836, 2015.

[32] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[33] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Opensurfaces: A richly annotated catalog of surface appearance. *ACM Transactions on graphics (TOG)*, 32(4):1–17, 2013.

[34] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[35] Ting-Chun Wang, Jun-Yan Zhu, Ebi Hiroaki, Manmohan Chandraker, Alexei A Efros, and Ravi Ramamoorthi. A 4d light-field dataset and cnn architectures for material recognition. In *European Conference on Computer Vision*, pages 121–138. Springer, 2016.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. International Conference on Learning Representations (ICLR'15)*, 2015.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

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

[39] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(9), 2012.

[40] J. Sánchez, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105, 12 2013.

[41] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 392–407, 2014.

[42] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[43] Hang Zhang, Jia Xue, and Kristin Dana. Deep ten: Texture encoding network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 708–717, 2017.

[44] Peng Tang, Xinggang Wang, Baoguang Shi, Xiang Bai, Wenyu Liu, and Zhuowen Tu. Deep fishernet for image classification. *IEEE transactions on neural networks and learning systems*, 30(7):2244–2250, 2019.

[45] Lingqiao Liu, Chunhua Shen, Lei Wang, Anton van den Hengel, and Chao Wang. Encoding high dimensional local features by sparse coding based fisher vectors. In *Advances in Neural Information Processing Systems(NIPS)*, 2014.

[46] Tsung-Yu Lin and Subhransu Maji. Improved bilinear pooling with cnns. In Gabriel Brostow Tae-Kyun Kim, Stefanos Zafeiriou and Krystian Mikolajczyk, editors, *Proceedings of the British Machine Vision Conference(BMVC)*, pages 117.1–117.12. BMVA Press, September 2017.

[47] Ying Nian Wu, Song Chun Zhu, and Xiuwen Liu. Equivalence of julesz ensembles and frame models. *International Journal of Computer Vision*, 38(3):247–265, 2000.

[48] Jianwen Xie, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Learning sparse frame models for natural image patterns. *International Journal of Computer Vision*, 114(2):91–112, 2015.

[49] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

[50] Song Chun Zhu, Xiu Wen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient markov chain monte carlo-toward a" trichromacy" theory of texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):554–569, 2000.

[51] Song-Chun Zhu, Cheng-En Guo, Yizhou Wang, and Zijian Xu. What are textons? *International Journal of Computer Vision*, 62(1):121–143, 2005.

[52] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[53] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

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

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

[56] 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.

[57] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer, 2014.

[58] 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.

[59] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1578–1585, 2013.

[60] Jonathan Delhumeau, Philippe-Henri Gosselin, Hervé Jégou, and Patrick Pérez. Revisiting the vlad image representation. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 653–656, 2013.

[61] Yang Song, Fan Zhang, Qing Li, Heng Huang, Lauren J O'Donnell, and Weidong Cai. Locally-transferred fisher vectors for texture classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4912–4920, 2017.

[62] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010.

[63] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear convolutional neural networks for fine-grained visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1309–1322, 2017.

[64] Mandar Dixit and Nuno Vasconcelos. Object based scene representations using fisher scores of local subspace projections. *Advances in neural information processing systems*, 2016.

[65] N. Passalis and A. Tefas. Learning bag-of-features pooling for deep convolutional neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5766–5774, 2017.

[66] Yunsheng Li, Mandar Dixit, and Nuno Vasconcelos. Deep scene image classification with the mfafvnet. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5746–5754, 2017.

[67] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

[68] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.

[69] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015.

[70] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis and the controlled generation of natural stimuli using convolu-

tional neural networks. In *Bernstein Conference 2015*, pages 219–219, 2015.

[71] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[72] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2965–2973, 2015.

[73] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*, 2016.

[74] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 1821–1830, 2017.

[75] Qilong Wang, Peihua Li, and Lei Zhang. G2denet: Global gaussian distribution embedding network and its application to visual recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2730–2739, 2017.

[76] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, pages 2921–2930, 2017.

[77] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 317–326, 2016.

[78] Shu Kong and Charless Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 365–374, 2017.

[79] Kaicheng Yu and Mathieu Salzmann. Statistically-motivated second-order pooling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 600–616, 2018.

[80] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pages 430–443. Springer, 2012.

[81] Zhiwu Huang and Luc Van Gool. A riemannian network for spd matrix learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[82] Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 947–955, 2018.

[83] Qilong Wang, Jiangtao Xie, Wangmeng Zuo, Lei Zhang, and Peihua Li. Deep cnns meet global covariance pooling: Better representation and generalization. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[84] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[85] Tsung-Yu Lin, Subhransu Maji, and Piotr Koniusz. Second-order democratic aggregation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 620–636, 2018.

[86] Mengran Gou, Fei Xiong, Octavia Camps, and Mario Sznaier. Monet: Moments embedding network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3175–3183, 2018.

[87] Tan Yu, Yunfeng Cai, and Ping Li. Toward faster and simpler matrix normalization via rank-1 update. In *European Conference on Computer Vision*, pages 203–219. Springer, 2020.

[88] Tan Yu, Xiaoyun Li, and Ping Li. Fast and compact bilinear pooling by shifted random maclaurin. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3243–3251, 2021.

[89] Vincent Andrearczyk and Paul F Whelan. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognition Letters*, 84:63–69, 2016.

[90] Yuting Hu, Zhiling Long, and Ghassan AlRegib. Multi-level texture encoding and representation (multer) based on deep neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4410–4414. IEEE, 2019.

[91] Xiyang Dai, Joe Yue-Hei Ng, and Larry S Davis. Fason: First and second order information fusion network for texture recognition. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7352–7360, 2017.

[92] Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. Hierarchical bilinear pooling for fine-grained visual recognition. In *Proceedings of the European conference on computer vision (ECCV)*, pages 574–589, 2018.

[93] Shuvozit Ghose, Pinaki Nath Chowdhury, Partha Pratim Roy, and Umapada Pal. Modeling extent-of-texture information for ground terrain recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4766–4773. IEEE, 2021.

[94] Wei Zhai, Yang Cao, Jing Zhang, and Zheng-Jun Zha. Deep multiple-attribute-perceived network for real-world texture recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3613–3622, 2019.

[95] Wei Zhai, Yang Cao, Zheng-Jun Zha, HaiYong Xie, and Feng Wu. Deep structure-revealed network for texture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11010–11019, 2020.

[96] Nalini Bhushan, A Ravishankar Rao, and Gerald L Lohse. The texture lexicon: Understanding the categorization of visual texture terms and their relationship to texture images. *Cognitive Science*, 21(2):219–246, 1997.

[97] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft

coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[98] Hdr environment lighting map, November 2013. `http://www.pauldebevec.com/`.

[99] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[100] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009.

[101] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[102] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[103] Simon Hecker, Dengxin Dai, and Luc Van Gool. Failure prediction for autonomous driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1792–1799. IEEE, 2018.

[104] Ondrej Linda, Todd Vollmer, and Milos Manic. Neural network based intrusion detection system for critical infrastructures. In *2009 international joint conference on neural networks*, pages 1827–1834. IEEE, 2009.

[105] Buyu Liu and Vittorio Ferrari. Active learning for human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4363–4372, 2017.

[106] Suyog Dutt Jain and Kristen Grauman. Active image segmentation propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2864–2873, 2016.

[107] Keze Wang, Xiaopeng Yan, Dongyu Zhang, Lei Zhang, and Liang Lin. Towards human-machine cooperation: Self-supervised sample mining for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1605–1613, 2018.

[108] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[109] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *arXiv preprint arXiv:1705.08500*, 2017.

[110] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[111] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009.

[112] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

[113] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[114] Wenjie Luo, Alex Schwing, and Raquel Urtasun. Latent structured active learning. *Advances in Neural Information Processing Systems*, 26:728–736, 2013.

[115] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017.

[116] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 93–102, 2019.

[117] Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. 2018.

[118] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[119] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[120] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[121] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[122] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[123] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *The International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

[124] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[125] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[126] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

[127] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[128] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural

networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[129] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[130] Donggeun Yoo and In So Kweon. Learning loss for active learning. pages 93–102, 2019.

[131] Heinrich Jiang, Been Kim, Melody Y Guan, and Maya Gupta. To trust or not to trust a classifier. *arXiv preprint arXiv:1805.11783*, 2018.

[132] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *European Conference on Machine Learning*, pages 413–424. Springer, 2006.

[133] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[134] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *International journal of computer vision*, 108(1):97–114, 2014.

[135] Xin Li and Yuhong Guo. Multi-level adaptive active learning for scene classification. In *European Conference on Computer Vision*, pages 234–249. Springer, 2014.

[136] Qiujia Li, PM Ness, Anton Ragni, and Mark JF Gales. Bi-directional lattice recurrent neural networks for confidence estimation. In *ICASSP*

2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6755–6759. IEEE, 2019.

[137] Anton Ragni, Qiujia Li, Mark JF Gales, and Yongqiang Wang. Confidence estimation and deletion prediction using bidirectional recurrent neural networks. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 204–211. IEEE, 2018.

[138] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[139] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018.

[140] Juan Eugenio Iglesias, Ender Konukoglu, Albert Montillo, Zhuowen Tu, and Antonio Criminisi. Combining generative and discriminative models for semantic segmentation of ct scans via active learning. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 25–36. Springer, 2011.

[141] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[142] Suo Qiu. Global weighted average pooling bridges pixel-level localization and image-level classification. *CoRR*, abs/1809.08264, 2018.

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

[144] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[145] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[146] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proc. International Conference on Machine learning (ICML'10)*, 2010.

[147] Pierre jacob, David Picard, Aymeric Histace, and Edouard Klein. Efficient codebook and factorization for second order representation learning. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[148] Lingqiao Liu, Peng Wang, Chunhua Shen, Lei Wang, Anton Van Den Hengel, Chao Wang, and Heng Tao Shen. Compositional model based fisher vector coding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2335–2348, 2017.

[149] I. Daubechies, M. Defrise, and C.D. Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.*, 57:1413–1457, 2004.

[150] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

[151] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[152] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Appendix A

Details of CNN's three components

A.1 Convolution component

Let denote input 3D-tensor of the layer l as $\mathbf{X}^l \in \mathbb{R}^{C_l \times W_l \times H_l}$ and every spatially local feature vector at spatial position(w, h) is represented as $\mathbf{x}^l_{w,h} \triangleq \mathbf{X}^l_{:,w,h} \in \mathbb{R}^{C_l}$, where C_l, W_l, H_l are the number of channels, width, and height. For instance, input image can be denoted as $\mathbf{X}^1 \in \mathbb{R}^{C_1 \times W_1 \times H_1}$ and $C_1 = 3$ as color images are defined by red, green, blue channels. On the image, $\mathbf{x}^1_{w,h}$ is a pixel located at (w,h).

Convolution layer l is equipped with C_{l+1} convolution kernels

$$\boldsymbol{\mathcal{E}}^l \in \mathbb{R}^{C_{l+1} \times C_l \times KW_l \times KH_l} \tag{A.1}$$

where KW_l and KH_l are kernel's width and height. These kernels extract local features as their window size is smaller than input features: $KW_l < W_l$ and $KH_l < H_l$. Every cth kernel $\mathbf{E}^l_c \triangleq \boldsymbol{\mathcal{E}}^l_{c,:,:,:} \in \mathbb{R}^{C_l \times KW_l \times KH_l}$ now proceeds to convolution by making dot product at the position (w, h) with a combination

of $\mathbf{x}_{w,h}^{l}$ and it's neighbors:

$$\mathbf{x}_{w,h}^{l+1} = [\mathbf{E}_1^l \odot \phi(\mathbf{x}_{w,h}^l), ..., \mathbf{E}_{C_{l+1}}^l \odot \phi(\mathbf{x}_{w,h}^l)] \in \mathbb{R}^{C_{l+1}} \tag{A.2}$$

where $\odot$ is dot product operation and 3D-tensor $\phi(\mathbf{x}_{w,h}^l) \in \mathbb{R}^{C_l \times \ KW_l \times KH_l}$ is created by cropping all the neighboring feature vectors centered at (w, h) within a window of size $KW_l \times KH_l$.

For simplicity, previously, we omit the description of bias and activation function. lth layer's bias parameters, denoted as $\mathbf{b}^l = [b_1^l, b_2^l, ..., b_{C_{l+1}}^l]$, are added in an element-wise way to the results obtained after the convolution operated by the kernels $\boldsymbol{\mathcal{E}}^l$:

$$\mathbf{x}_{w,h}^{l+1} = [\mathbf{E}_1^l \odot \phi(\mathbf{x}_{w,h}^l) + b_1^l, ..., \mathbf{E}_{C_{l+1}}^l \odot \phi(\mathbf{x}_{w,h}^l) + b_{C_{l+1}}^l] \in \mathbb{R}^{C_{l+1}} \tag{A.3}$$

As observed in the Eq A.3, every element in the $\mathbf{x}_{w,h}^{l+1}$ is the result of linear combination of a kernel, local feature vectors and a bias value. The rule of activation function is to separate two linear convolution layers by introducing non-linearity. ReLU (Rectified Linear Unit) is a common choice of activation function:

$$\begin{aligned}\mathbf{x}_{w,h}^{l+1} &= [ReLU(\mathbf{E}_1^l \odot \phi(\mathbf{x}_{w,h}^l) + b_1^l), ..., ReLU(\mathbf{E}_{C_{l+1}}^l \odot \phi(\mathbf{x}_{w,h}^l) + b_{C_{l+1}}^l)], \\ &= [max(\mathbf{E}_1^l \odot \phi(\mathbf{x}_{w,h}^l) + b_1^l, 0), ..., max(\mathbf{E}_{C_{l+1}}^l \odot \phi(\mathbf{x}_{w,h}^l) + b_{C_{l+1}}^l, 0)] \in \mathbb{R}^{C_{l+1}},\end{aligned} \tag{A.4}$$

The layer l repeats the operation of Eq.A.4 at every position on the input $\mathbf{X}^l$. As a consequence, a collection of results is lth layer's output or $(l+1)$th layer's input $\mathbf{X}^{l+1}$.

A.1.1 Local Pooling Layer

It should mention that multiple convolution layers form a block. Between two consecutive blocks, in order to decrease computation burden and increase the kernel's receptive field, there is a local pooling layer which can be viewed as a convolution layer with a down-sampling kernel. It slides over $\mathbf{X}^l$ and spatially reduce the input into $\mathbf{X}^{l+1}$ of smaller size.

A.1.2 Residual Connection and Batch Normalization.

Besides AlexNet and VGG networks, we also did some experiments with ResNet networks [37] which contain two supplementary elements in the convolution component. Skip connection structure connects two tensors at different layers with element-wise addition. This structure is very useful to solve vanishing gradient problem while training a network with numerous layers. ResNet network always use batch normalization [152] at the end of Eq. A.3 which aims at normalizing feature tensors by re-centering and re-scaling and consequently making the whole network training faster and more stable.

In resume, an example of convolution component's structure is schematized by the Fig A.1.

A.2 Pooling component

In the previous description related to convolution layers, there exists already a type of pooling layer: local pooling layer which is performed locally on the input feature tensor. On the contrary, pooling component works globally on the whole input tensor. It is dedicated to aggregating $\mathbf{X}^l$ into a feature vector representation $\mathbf{a}^{l+1} \in \mathbb{R}^{C_{l+1}=C_l}$, visualized as schema blocks in the red

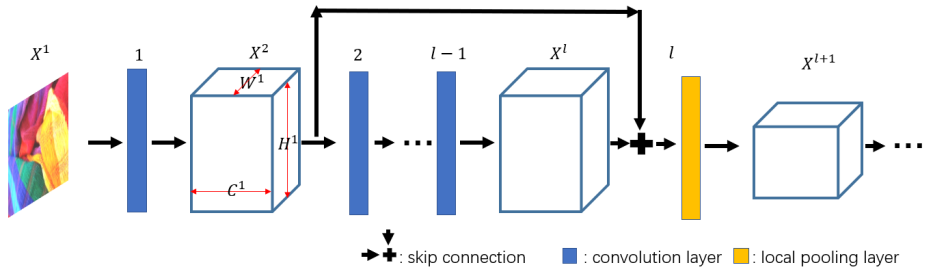


Figure A.1 – These schemas describe the first block of convolutional components, which is composed of multiple convolution layers and one pooling layer. Every convolution layer applies Eq A.4 to its input and the pooling layer downsamples feature tensor $\mathbf{X}^l$. Note that batch normalization is not explicitly drawn here since it can be fused as a part of convolution layer.

bounding box of Fig 1.3.

We can take advantage of downsampling techniques used in the local pooling layer, such as max pooling and average pooling, but this time in a global manner. Moreover, there exist many other state-of-the-art pooling methods, dedicated to producing more discriminative global representation. Deeper explanations are given in the section 2.3.

A.3 Fully connected component

In a CNN, Fully Connected (FC) layer transform feature vector $\mathbf{a}^{(l)} \in \mathbb{R}^{C_l}$ into another feature vector. In particular, the output of the last FC layer is a class prediction vector, or called the output of a CNN: $f(\mathbf{I}) = \hat{\mathbf{z}} \in \mathbb{R}^K$. K is the number of given categories. Generally speaking, a FC layer can be seen as a convolution layer whose kernel width KW and height KH are equal to 1 :

$$\boldsymbol{\mathcal{E}}^l \in \mathbb{R}^{C_{l+1} \times C_l \times KW_l=1 \times KH_l=1} \tag{A.5}$$

And the output can be calculated by:

$$\begin{aligned}\mathbf{a}^{l+1} &= [ReLU(\mathbf{E}_1^l \odot \phi(\mathbf{a}^l) + b_1^l), ..., ReLU(\mathbf{E}_{C_{l+1}}^l \odot \phi(\mathbf{a}^l) + b_{C_{l+1}}^l)], \\ &= [max(\mathbf{E}_1^l \odot \phi(\mathbf{a}^l) + b_1^l, 0), ..., max(\mathbf{E}_{C_{l+1}}^l \odot \phi(\mathbf{a}^l) + b_{C_{l+1}}^l, 0)] \in \mathbb{R}^{C_{l+1}},\end{aligned} \tag{A.6}$$

Note that there is no ReLU activation function for the last FC layer. Instead, using softmax to the last layer's output $\hat{\mathbf{z}}$ can transformed it into probability output $\hat{\mathbf{p}}$, as conformed in the Eq 1.2.

Appendix B

French translations

Table des matières

Figure B.1 – Exemples d'images de matériaux et leurs catégories. Images de Flickr Material Database(FMD) [1].

B.1 Introduction

La classification des matériaux est une tâche de reconnaissance visuelle étroitement liée à la classification des textures et dédié à classer les images de textures/matériaux en catégories telles que les tissus, l'eau, l'acier, le feuillage, ... (voir Fig. B.1). En tant qu'une des perceptions visuelles de base, la reconnaissance des matériaux a une différence évidente avec la reconnaissance d'objets parce que son entrée concerne des informations visuelles provenant de surfaces, au lieu d'objets [8]. Apprendre un système à reconnaître les matériaux dans une image est un challenge mais très utile pour de nombreuses tâches de vision par ordinateur, comme la classification des aliments [9], l'imagerie satellitaire ou aérienne [10, 11] , la reconnaissance de terrain au sol et l'analyse d'images médicales [14, 15, 16]. De plus, des algorithmes de reconnaissance de matériaux peuvent être implémentés dans des systèmes visuels robotiques qui permettent la recherche de produits, la manipulation d'objets ou la navigation autonome sur la surface constituée d'un matériau spécifique [13].

En 2012, un réseau de neurones convolutifs [28] (Convolutional Neural Network(CNN) en anglais) a battu le record de l'accuracy de classification

d'images dans ImageNet ILSVRC [29], une très grande base de données de reconnaissance d'objets. Pour la recherche sur la classification de matériaux, les CNNs pré-entrainés sur ImageNet remplacent les filtres conçus par des experts, en s'appuyant sur leurs caractéristiques hautement discriminantes pour la reconnaissance d'objets [30, 31]. Plus récemment, de nouvelles approches ont déjà trouvé des solutions intelligentes et originales pour mieux s'adapter au classification de matériaux.

Puisque ces solutions sont toutes basées sur des réseaux de neurones profonds, nous avons construit nos contributions sur de telles architectures. La première section de ce chapitre présente le workflow général des réseaux de neurones profonds et la seconde présente les motivations et les idées principales de nos contributions.

B.1.1 Classification des images avec CNN

propagation avant

Pour une tâche de classification d'images sur K classes, le but est de classer correctement une image I dans sa catégorie de vérité terrain $y \in \mathcal{Y} = \{1, ..., K\}$. La propagation avant d'un CNN peut être abstraite comme une fonction $f(.)$ qui projette une image dans un vecteur de prédiction $\hat{\mathbf{z}} \in \mathbb{R}^K$:

$$\hat{\mathbf{z}} = f(\mathbf{I}). \tag{B.1}$$

Ensuite, cette prédiction $\hat{\mathbf{z}}$ est transformée en un vecteur de probabilité $\hat{\mathbf{p}}$ avec une fonction $softmax(.)$:

$$\hat{\mathbf{p}} = softmax(\hat{\mathbf{z}}). \tag{B.2}$$

l'indice de l'élément avec la probabilité la plus élevée est choisi comme

catégorie prédite $\hat{y}$:

$$\hat{y} = \underset{k \in \mathcal{Y}}{\operatorname{argmax}}\, \hat{p}_k. \tag{B.3}$$

Dans le cas d'une prédiction correcte, la catégorie prédite est égale à la classe de vérité terrain :

$$\hat{y} = y. \tag{B.4}$$

Structure d'un réseau de neurones convolutifs

Si nous regardons la fonction f d'un CNN, sa structure est une sequence de couches. Selon leurs propriétés, nous avons regroupé ces couches en trois composants séquentiels : convolution, pooling et classification.

Dans le composant de convolution, l'extraction sur une image en entrée est réalisée avec des couches de convolution répétées plus des couches de pooling locales. Elles sont empilées de manière hiérarchique. Cela signifie que les premières couches extraient des caractéristiques primitives de bas niveau, telles que des bords ou des couleurs, tandis que les dernières couches combinent ces caractéristiques de bas niveau en caractéristiques sémantiques de haut niveau, telles que des mains, des roues ou des arbres. Ensuite, le composant de pooling agrège les caractéristiques locales du composant de convolution en un vecteur de caractéristiques. Enfin, le composant de classification fournit les probabilités prédites pour les catégories considérées.

B.1.2 Contributions

L'objectif de cette thèse est d'améliorer les performances de classification des matériaux sur la base des CNN. Pour atteindre cet objectif, nous nous concentrons sur deux étapes clés qui font partie de la composante de pooling, à savoir la sélection de caractéristiques locales et le pooling sans ordre.

Feature selection

Aujourd'hui, comme mentionné dans la section précédente, dans la plupart des architectures CNN, après des couches convolutives empilées extrayant les caractéristiques locales de l'image d'entrée, en tant qu'un composant de pooling utilisé souvent, la couche Global Average Pooling (GAP) fusionne toutes les caractéristiques locales en un seul vecteur de caractéristiques global [37]. Ensuite, le composant de classification prédit la classe d'images en fonction de ce vecteur global de caractéristiques. Avec cette approche classique, chaque vecteur de caractéristiques locales contribue également à la décision finale grâce à l'opération de moyennage. Cependant, lorsque de grandes zones des images sont ambiguës ou lorsque des informations utiles sont principalement fournies dans certaines zones minuscules, la moyenne de toutes les caractéristiques locales peut être sous-optimale. Et nous montrerons que cela est d'autant plus vrai dans les images de matériaux. Un exemple illustratif est montré sur la Fig. 1.4: dans la colonne de gauche, certaines parties petites mais informatives sont masquées et cela rend la prédiction de classe très difficile avec les zones grandes et ambiguës. Une fois que l'on a accès à ces détails (colonne de droite), la prédiction de classe devient beaucoup plus facile. Cependant, comment choisir des zones discriminantes et éliminer les caractéristiques ambiguës n'est pas facile. Dans le chapitre 3, afin de sélectionner automatiquement les vecteurs de caractéristiques locales les plus informatifs, nous proposerons d'utiliser un score de confiance qui représente l'utilité des vecteurs de caractéristiques locales sur la zone de chaque image. En exploitant une approche très récente et réussie, conçue pour la prédiction des défaillances, nous proposons de prédire la confiance des caractéristiques locales avec une branche supplémentaire dans le réseau. Seuls les vecteurs de caractéristiques locales avec des prédictions de confiance

plus élevées sont conservés et moyennés dans un vecteur de caractéristiques global. Nous fournirons aussi des résultats quantitatifs et qualitatifs sur trois bases de données de matériaux et démontrerons que notre méthode augmente non seulement l'accuracy de la classification, mais aussi améliore également le calibrage des probabilités en sortie.

Pooling sans ordre

Par rapport aux tâches de reconnaissance d'objets, la reconnaissance de matériaux a ses propres propriétés. L'une de ces propriétés est son arrangement spatial sans ordre. Comme illustré sur la figure 1.6, en tant qu'un objet, les parties discriminantes d'une voiture, comme les roues, les vitres, ont de fortes relations spatiales et topologiques. Pour les images de matériaux, de telles relations prédéfinies entre les surfaces n'existent pas et ne doivent pas être prises en compte dans le vecteur global de caractéristiques.

Le pooling sans ordre est apparue récemment et devient intéressante pour les tâches de classification des matériaux car elle agrège les caractéristiques sans prendre en compte leurs arrangements spatiaux dans l'image. Néanmoins, l'intégration du pooling sans ordre dans une architecture profonde présente deux inconvénients principaux de notre point de vue. Premièrement, parce que les caractéristiques profondes se trouvent dans des espaces de grande dimension, des outils spécifiques sont nécessaires pour modéliser avec précision leurs distributions. En nous inspirant de l'approche de [45], nous avons proposé de l'implémenter comme un module entraînable dans un réseau de neurones profonds afin de modéliser avec précision les caractéristiques profondes.

Deuxièmement, le traitement des statistiques de second ordre telles que celles fournies par les vecteurs de Fisher n'est pas facile et nécessite des

étapes de normalisation successives [46]. Par conséquent, nous avons proposé une approche nouvelle et originale pour normaliser les matrices non carrées ou symétriques qui représentent des statistiques de second ordre. À notre connaissance, c'est la première fois que la normalisation matricielle s'applique à une représentation basée sur Fisher.

Ces deux contributions principales sont intégrées dans une architecture profonde afin que le réseau final puisse être entraîné de bout en bout avec la fonction objectif de classification.

B.1.3 Organisation de la thèse

Le reste de la thèse est organisé comme suit. Tout d'abord, nous décrivons les travaux connexes dans le chapitre B.2 , y compris les solutions classiques de classification des matériaux et les approches basées sur CNN. Certaines parties de ce chapitre font référence à notre étude publiée dans la conférence *International Conference on Big Data, Machine Learning and Applications (BIGDML)* en 2019.

Le chapitre B.3 présente notre première contribution principale, brièvement introduite dans la section B.1.2. Certaines parties de ce chapitre renvoient à notre papier publié dans *International Conference on Image and Vision Computing, New Zealand (IVCNZ)* en 2020.

Notre deuxième contribution principale, brièvement introduite dans la section B.1.2, est précisément discutée au chapitre B.4. Certaines parties de ce chapitre, liées au sparse coding, renvoient à notre papier publié dans *International Conference on Computer Analysis of Images and Patterns (CAIP)* en 2021. Une version étendue de cet article avec normalisation a été soumise en juillet 2021 pour publication au journal *Computer Vision and Image Understanding*.

Enfin, le chapitre B.5 tire des conclusions, révèle les défis et tendances actuels dans le domaine de la classification des matériaux et propose des perspectives pour les travaux futurs.

B.2 Chapitre 2

Dans ce chapitre, nous explorons et présentons les travaux les plus remarquables dans le contexte de la classification des matériaux, en commençant par les caratéristques conçus par les experts [7, 17, 18, 19, 20, 21, 22, 23, 24, 53, 54, 55] jusqu'aux solutions basées sur Deep-CNN [30, 40, 58, 59, 60, 31, 61, 62, 41, 45, 64]. Les articles les plus récentes [65, 42, 43, 44, 63, 66, 67, 69, 77] révèlent que le pooling sans ordre et l'apprentissage de bout en bout sont deux éléments essentiels pour la classification des matériaux. Les approches les plus performantes sont basées sur un processus d'apprentissage de bout en bout qui permet de faire coopérer les différents modules de l'architecture profonde vers un seul objectif, qui est de minimiser la perte de classification actuelle. Et le pooling sans ordre, qui est actuellement la principale solution proposée dans les états de l'art, exploite efficasement une des propriétés des matériaux: l'arrangement spatial sans ordre sur une image. Néanmoins, les deux groupes de méthodes pourraient être grandement améliorées par :

- pondérer la contribution de chaque vecteur de caractéristiques locales dans le pooling global, profitant ainsi de la spécificité des images des matériaux qui montrent des zones locales très discriminantes à côté de zones très communes (non discriminantes),

- faire attention à la qualité de l'estimation de distribution de caractéristiques profondes, qui se situe dans un espace de grande dimension,

- normaliser la représentation statistique de second ordre avant d'appliquer l'étape de classification.

Ces remarques seront les points de départ de nos solutions originales détaillées dans les prochains chapitres(B.3 et B.4).

Parce que nos contributions devraient être confrontées aux travaux récents, nous proposons également, dans ce chapitre, de présenter et de classer les nombreuses bases de données de matériaux. Specifiquement, les bases de données de matériaux (voir Fig 2.4) nous utilisons dans les prochaines chapitres ont été créés dans lesquels les images ont été acquises dans des conditions non contrôlées et n'étaient pas nécessairement remplies par le matériel cible, y compris les informations contextuelles.

B.3 Chapitre 3

Aujourd'hui, de nombreuses approches réussies reposent sur l'extraction automatique de caractéristiques locales avec des réseaux de neurones profonds suivis d'une couche du pooling. Spécifiquement, dans ce chapitre, c'est «Global Average Pooling(GAP)» qui fusionne toutes les caractéristiques locales en un seul vecteur de caractéristiques. Ensuite, une couche de classification prédit la classe d'image à partir de ce vecteur de caractéristiques. Parce que chaque vecteur de caractéristiques locales est uniformément moyenné avec les autres, chaque vecteur contribue également à la décision finale. Par conséquent, lorsque de grandes zones des images sont ambiguës et que les informations utiles sont principalement fournies par une petite partie des vecteurs de caractéristiques, la moyenne de toutes les caractéristiques locales pourrait conduire à de mauvaises prédictions. C'est surtout le cas pour la classificaiton des matériaux.

Dans ce chapitre, nous faissons une hypothèse que l'utilité de chaque vecteur de caractéristiques locales est liée à la confiance du réseau lors de la prédiction de la classe d'images à partir de ce vecteur de caractéristiques. Et la confidence est définit par 'True class probability(TCP)' qui est la probabilité prédite de la catégorie de vérité terrain. Nous avons appris un réseau à deux branches pour produire des prédictions locales, ainsi que des confidences associées. Ces confiances prédites sont utilisées pour supprimer les vecteurs de caractéristiques locales ayant des confiances inférieures avant de faire la moyenne de toutes les caractéristiques locales(voir la figure 3.1).

Nous présentons les résultats expérimentaux fournis par notre approche pour la classification de matériaux. Les tests sont menés sur trois bases de données (KTH [3],Flickr Material Dataset(FMD) [1] et 4D-Light [35]) et les résultats sont comparés avec des alternatives récentes qui sont 'temperature scaling' [5], 'Entropy' [115, 116], 'MaxProb' [108, 109, 110], 'MC-Dropout' [123] et GWAP [142].

Les premiers résultats sont présentés dans le tableau 3.2, où trois critères sont fournis : l'accuracy de la classification, Expected Calibration Error(ECE) [144] et Negative Log Likelihood(NLL) [5]. ECE et NLL mesurent le degré de mauvais calibrage des probabilités de sortie. Ils sont faibles pour des probabilités bien calibrées. On peut remarquer que notre approche surpasse nettement toutes les méthodes testées pour les trois critères. En effet, en supprimant les vecteurs de caractéristiques locales les moins sûrs, notre modèle est capable de prédire des probabilités calibrées et précises. Il est à noter que les architectures de notre solution et GWAP sont identiques. Cela montre clairement que la supervision de la deuxième branche avec les TCPs est une bonne solution pour prédire des confiances précises et sélectionner les meilleures fonctionnalités locales. Enfin, nous proposons de comparer

l'accuracy apportée par notre méthode avec des solutions des etats de l'art pour la classification des matériaux (voir tableau 3.3). Malgré la simplicité de notre approche, nous remarquons qu'elle surpasse toutes les solutions. Ces résultats confirment qu'il est très intéressant de concentrer la décision de catégorie sur des zones spécifiques d'images de matériaux et que prédire la confiance de chaque vecteur de caractéristiques locales est un moyen intelligent de le faire.

B.4 Chapitre 4

Après avoir amélioré Global Average Pooling au chapitre B.3, nous nous concentrons maintenant sur un autre algorithme de pooling : Fisher Scores qui se sont avérées être des caractéristiques globales précises pour la classification. L'idée principale de FV est d'estimer la distribution des données avec un modèle de mélange de gaussiens et de caractériser chaque point de données avec les dérivées sur les paramètres du modèle. Cependant, un modèle de mélange de gaussiens ne semble pas bien adapté aux caractéristiques locales profondes car elles se situent dans un espace de très grande dimension et nécessitent trop de gaussiennes pour estimer cet espace avec précision [45]. Liu et al. ont proposé une solution intelligente pour surmonter ce problème qui consiste à échantillonner le centre de chaque gaussienne à partir d'un sous-espace et donc de bénéficier d'un nombre infini de gaussiennes pour s'adapter à la distribution des données. Malheureusement, leur approche ne peut pas profiter de l'intérêt principal du CNN, à savoir apprendre de bout en bout les couches de l'extraction de caractéristiques, du pooling et de classification. Pour résoudre ce problème, nous avons proposé d'implémenter la méthode comme un module entraînable dans un réseau de neurones profonds par un

algorithme proposé dans [146].

Deuxièmement, une étude récente a montré que la normalisation des statistiques de second ordre a un fort impact sur les performances de classification [46]. Malheureusement, contrairement au pooling bilinéaire utilisé dans [46], notre représentation de Fisher ne fournit pas de matrice carrée, et il rend ainsi la solution de [46] inutilisable. Ainsi, dans ce chapitre, nous proposons d'adapter la racine carrée d'une matrice aux matrices non carrées et d'intégrer ce module original dans un réseau profond.

Nous menons des expériences sur trois bases de données, qui présentent de fortes différences en termes de tâches et d'échelle. L'accuracy de classification top-1 de notre approche et de nombreuses alternatives sont résumées dans le tableau 4.2. Les méthodes appelées « Off-the-self » utilisent des modules de pooling qui sont entraînées avec les caractéristiques fournies par un réseau pré-entraîné et leurs sorties sont transmises à un classificateur qui est entraîné à une étape suivante. Le groupe « end-to-end » contient des approches qui utilisent des réseaux entraînables de bout en bout. Nous remarquons que les résultats fournis par les réseaux « End-to-End » surpassent globalement ceux des solutions sur « Off-the-shelf ». Cela montre qu'il est toujours préférable de faire fonctionner les modules ensemble pour optimiser la même fonction objectif au lieu de les optimiser indépendamment.

Outre l'attribut d'apprentissage de bout en bout, notre approche est basée sur [45] qui produit les caractéristiques en second ordre plus discriminantes que le vecteur de Fisher classique ou VLAD. La combinaison intelligente proposée de ces deux avantages fait que notre méthode surpasse les alternatives pour tous les bases de données et les backbones.

B.5 Chapitre 5

B.5.1 Conclusion

La classification des images des matériaux est une tâche cruciale en vision par ordinateur car elle est impliquée dans de nombreuses applications réelles telles que la robotique ou le tri automatique des déchets, et parce qu'elle peut aider dans de nombreux autres problèmes tels que la classification d'images finement granuleuses. Elle consiste à classer correctement les images avec des matériaux cibles d'une catégorie donnée. Au cours des dernières années, grâce à leurs performances supérieures, les réseaux de neurones convolutifs (CNN) sont apparus et sont devenus un outil prometteur pour résoudre de nombreux problèmes de vision par ordinateur, y compris la classification d'images. Les réseaux profonds ont également été introduits dans la classification des matériaux. En transférant simplement un réseau pré-entraîné sur une tâche de classification d'images à grande échelle, un meilleur accuracy est obtenu que l'ancien état de l'art. Cependant, contrairement à la reconnaissance d'objets, la classification des matériaux nécessite un traitement spécifique. Dans cette thèse, nous nous intéressons principalement à deux particularités des images matérielles :

- Les grandes zones de la surface des matériaux sont visuellement ambiguës et les informations discriminantes résident principalement dans des zones minuscules, voir Fig 1.4.

- Contrairement aux objets, les instances de matériaux dans une image montrent un arrangement spatial sans ordre, voir Fig 1.6.

Ainsi, transférer directement les architectures CNN, initialement conçues pour la classification d'objets, est inévitablement sous-optimal. Plus pré-

cisément, en se référant aux deux propriétés décrites ci-dessus, nous avons identifié deux inconvénients des CNN réels:

- Pour l'une des architectures les plus populaires de CNN, appelée ResNet, son Global Average Pooling agrège les vecteurs de caractéristiques locales. Ces opérations font que des caractéristiques pertinentes mais petites sont submergées par des régions locales ambiguës mais nombreuses, produisant par conséquent des caractéristiques globales moins discriminantes.

- Nous avons démontré une amélioration de la classification en utilisant un pooling sans ordre plus sophistiqué, comme les vecteurs de Fisher classiques. Malheureusement, une estimation inexacte de la distribution des données se produit dans le contexte de caractéristiques profondes de grande dimension, car le nombre de gaussiennes dans le modèle de mélange gaussien est limité.

Dans cette thèse, nous avons contribué à les résoudre des manières suivantes:

- Afin d'identifier des caractéristiques provenant de régions ambiguës ou discriminantes, nous avons proposé d'ajouter dans un réseau classique une branche qui prédit des valeurs de confiance: la probabilité de classe de vérité térrain (TCP), associée à chaque vecteur de caractéristiques locales. Ensuite, nous exploitons ces valeurs TCP pour filtrer les caractéristiques locales ambiguës qui sont associées à des confiances plus faibles. Par conséquent, une représentation plus puissante est obtenue après avoir appliqué le Global Average Pooling au reste des caractéristiques locales. Selon les résultats expérimentaux sur trois bases de données de matériaux, notre approche surpasse les autres modèles classiques sur les critères de précision de la classification et de calibrage de

la probabilité de sortie.

- Pour résoudre le problème des gaussiennes limitées, nous avons remarqué que le sparse coding des centres gaussiens dans une base adaptée est un moyen prometteur d'augmenter le nombre de gaussiennes disponibles car il peut mieux s'adapter à la distribution des données. Dans le chapitre B.4, pour franchir un pas de plus que les travaux précédents, nous avons implémenté ce processus de codage dans un réseau d'apprentissage profond. Cela permet d'optimiser la base et le sparse code pour améliorer les performances de classification. De plus, une nouvelle normalisation et une soustraction moyenne ont également été intégrées à notre approche. Nous avons démontré qu'ils améliorent encore le pouvoir de discrimination des représentations du score de Fisher. En pratique, nous avons également proposé une stratégie d'apprentissage qui facilite l'initialisation des paramètres dans le module du sparse coding. Avec le soutien d'un apprentissage de bout en bout de ces représentations précises du score de Fisher, notre méthode surpasse l'état de l'art sur trois bases de données différents.

B.5.2 Limites

Pour les deux solutions que nous avons proposées, le processus d'apprentissage est composé de deux étapes, ce qui est une faiblesse de ces approches. En effet, pour la sélection de caractéristiques locales (Chapitre B.3), nous devons entraîner le classifieur dans la première étape puis le geler pendant la deuxième étape, quand on entraîne le prédicteur TCP. Bien que cette stratégie produise des valeurs TCP stables, il est clair qu'entraîner séparément le classificateur et la branche de prédiction TCP est légèrement compliqué. De

même, pour nos vecteurs Fisher de bout en bout (Chapitre B.4), l'ajout d'une étape d'entraînement d'échauffement avant d'entraîner l'ensemble du réseau est favorable pour la classfication, comme indiqué dans le tableau 4.3. Par rapport à la méthode associée dans [66], notre approche est évidemment plus adaptée à l'entraînement de CNN et nécessite moins de mémoire de calcul, mais elle nécessite toujours un processus d'entraînement en deux étapes pour obtenir les meilleurs résultats.

Comment choisir les hyper paramètres est une autre limitation pour les deux solutions. Dans le chapitre B.3, les caractéristiques locales ambiguës sont filtrées en fonction de leurs valeurs TCP prédites lorsqu'elles sont inférieures à un seuil et que ce seuil a été fixé de manière empirique. Bien que cette valeur n'ait pas un fort impact sur les résultats des différentes bases de données testés, elle n'est peut-être pas optimale pour chaque image. Une situation similaire se produit dans le chapitre B.4 avec plus d'hyper-paramètres tels que la taille des sparse codes , ou le nombre d'itérations dans le module LISTA. Bien que nous ayons discuté de leur impact sur l'accuracy dans les tableaux 4.4 et 4.5, en pratique, face à une nouvelle tâche, il serait préférable que notre méthode puisse les déduire automatiquement de la tâche plutôt que de demander à les définir manuellement.

B.5.3 Perspectives

Etant également un pooling de second ordre, le pooling bilinéaire est également intéressant par rapport à nos vecteurs de Fisher (Chapitre B.4), car il n'est pas nécessaire de s'adapter à la distribution des données et, par conséquent, il s'agit d'une méthode de pooling non paramétrique. Malgré sa simplicité, il surpasse de nombreuses méthodes du pooling sans ordre dans plusieurs tâches de classification d'images finement granuleuses, comme in-

diqué dans [69]. Comme discuté dans la Section 2.3, il existe deux pistes de recherche pour améliorer le pooling bilinéaire, l'un consistant à obtenir une représentation compacte et l'autre à sa normalisation. L'intégration de ces deux techniques devient une nouvelle tendance de recherche. La principale limitation est que l'exécution directe de la normalisation matricielle sur les caractéristiques du pooling bilinéaire compact (CBP) s'avère infaisable [85]. Même si quelques solutions ont été proposées [85, 86, 87, 88], le problème est partiellement résolu et de fortes limites subsistent.

Par exemple, Gou et al. a proposé de transformer la matrice de caractéristiques d'entrée en une "pseudo racine carrée", afin qu'un pooling bilinéaire classique puisse y être appliqué et fournir directement une pooling bilinéaire normalisée [86]. Malheureusement, le calcul de cette matrice nécessite d'appliquer la décomposition en valeurs singulières (SVD) sur la matrice d'entrée et nous savons que SVD n'est pas bien supporté sur GPU. Un travail futur intéressant pourrait être d'utiliser notre méthode de Newton supportée par GPU du chapitre B.4 pour calculer efficacement cette matrice de pseudo racine carrée.

Une autre perspective est de concevoir une méthode hybride qui tire les avantages de nos deux méthodes, car elles améliorent respectivement la classification des matériaux à différentes étapes du flux de travail. En effet, la sélection des caractéristiques les plus pertinentes est un bon moyen de diminuer la charge de calcul de l'extraction des statistiques de second ordre qui nécessitent de calculer de nombreux produits externes entre les vecteurs de caractéristiques locales. Outre le gain d'efficacité, la solution de sélection devrait supprimer les vecteurs bruyants de caractéristiques locaux, et ainsi aider à extraire une représentation globale du score de Fisher plus précise.