Universität des Saarlandes
Max-Planck-Institut für Informatik

Multi-Scale Feature Learning for Material Recognition

Wenbin Li

Department of Computer Science
Saarland University

Supervised by

Dr. Mario Fritz

Reviewers

Dr. Mario Fritz

Prof. Dr. Bernt Schiele

A thesis submitted for the degree of
Master of Science of Saarland Univeristy

Saarbrücken, January 31, 2013

Non-plagiarism Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

(Wenbin Li)
Saarbrücken, January 31, 2013

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

(Wenbin Li)
Saarbrücken, January 31, 2013

Abstract

The recent progress in sparse coding and deep learning has made unsupervised feature learning methods a strong competitor to hand-crafted descriptors. In computer vision, success stories of learned features have been mostly reported for object recognition tasks. In this thesis, we investigate if and how unsupervised feature learning can be used for material recognition.

We start by applying a recently proposed generative model, the Spike-and-Slab Sparse Coding model (S3C) to single-scale feature learning on material data and show improved performance over hand-crafted descriptors on the FMD and KTH-TIPS2 databases. Based on our analysis of the results, we extend the model to multi-scale feature learning and propose two strategies to incorporate scale information into the learning procedure. Compared with the state-of-the-art manually designed feature descriptors, our results indicate that the learned multi-scale features can produce further improvement on the same material classification benchmarks. In addition, we explore the feasibility of transfer representation and show it is possible to encode a dataset with the model learned on a different dataset to still obtain an efficient representation.

Acknowledgements

First of all I would like to thank my advisor Dr. Mario Fritz for his guidance in this work, bringing thoughtful insight into every one of our discussions. I also would like to thank Prof. Bernt Schiele for being my second reviewer.

I am also thankful to all the nice people in our department for being kind and helpful all the time.

Special thanks go to Ian Goodfellow who provided friendly help for using the pylearn2 package and some insights for the basic model.

Many thanks to Xiaokun Wu who helps to read this thesis and provides inspiring comments.

I should also thank Saarbruecken Graduate School of Computer Science and Max Planck Institute for Informatics for providing financial support and excellent environment to do research.

Last but not least, I should express my thanks to my family for their understanding and continuous support.

Contents

Chapter 1

Introduction

Material recognition is a fundamental aspect of visual perception. It enables humans to make predictions about the world and interact with care. In everyday life, we interact with various objects and scenarios, but seldom run into difficulty to recognize the associated material category with ease. How do I acquire a stable grasp of an object? Will I get stuck in this ground? However it is not the case with machines, it always takes much effort to equip a machine with this kind of ability. In Figure 1.1, we show two examples of these scenarios, recognizing materials plays an important role in both grasp task and navigation task. Computer vision investigates how to build recognition system from only visual appearance. Hence an efficient material recognition solution will help to tackle a wide range of uses such as in context awareness and robot manipulation.

FIGURE 1.1: Grasp task (Left): A robotic hand developed by researchers at Saarland University with associates in Bologna and Naples can be gentle and dexterous enough to hold an egg. Navigation task (Right): rover Spirit from NASA got stuck on Mars.

In this chapter, we motivate our work in the field of computer vision study. Afterwards, we summarize the major contributions from this thesis. Finally, we briefly outline the rest content of this thesis.

1.1 Motivation

Early investigations started from the study of textures and formed the first standard database CUReT [13] where the task is more of instance identification as each material sample denotes single category and images are usually taken under very restricted conditions. Near-perfect recognition rate has been reported [42]. Yet later study increase the variation of category by adding more samples and gradually extend the settings, showing that material in real world scenarios is far from solved [7]. More recently, the task has been pushed even further to less constraint settings. Newer databases, like the Flickr Material Database (FMD) [29] have been proposed which collects photos from Flickr as samples for common material and demonstrate the difficulties of material recognition. In particular, they incorporate many different descriptors in a Bayesian framework and provides an initial result on the data set, yet traditional manually designed feature descriptor, like LBP [34] and its variants [33][36] have still been shown to be one of the most powerful measure of feature descriptors and able to achieve state-of-the-art performance. It is nontrivial to come up with good design of visual features and efforts are clearly needed to explore the question how we can automatically learn features for this challenging and relevant problem.

1.2 Contributions

The main contributions of our work are:

- We present the first study of applying unsupervised feature discovery algorithms for material recognition and show improved performance over hand-crafted feature descriptors. In particular, we will apply the Spike-and-Slab Sparse Coding model to encode image patch and therefore accumulate the statistics to represent the whole material image where features are learned in an automatic way.
- We show how to incorporate scale information in our methods and achieved significant improvements over manually designed descriptors. Motivated by the importance of scale information in general computer vision study and previous success of manually designed multi-scale feature descriptor, we make an effort to encode scale information and extend our single-scale feature learning framework. Further, we propose two different strategies for multi-scale feature learning.
- We investigate if the feature learned on one database can be generalize to new database. This is of great potential value for practical use. In typical supervised learning framework, model learned on a specific data set is not guaranteed to be

applicable to new unseen data unless there is prior knowledge to ensure similar probability distributions. In addition, collecting sufficient amount of training data is usually time consuming which unavoidably limit the use of the framework. The intuition behind appearance transfer is that we believe the learned features capture some general characteristics which are shared between different datasets and hence make it more easily generalize to new data. The S3C model has shown success in transfer learning on object recognition task and therefore we explore how this model can apply for material recognition task.

1.3 Outline

After this introductory chapter, we continue this thesis with an overview of related work in Chapter 2, where we will first give a brief review on hand-crafted feature descriptors for material recognition task, including both classical descriptor like LBP and two recently proposed ones, namely Local Quantized Pattern (LQP) and kernel descriptor, and then discuss unsupervised feature learning as the alternative approach. In chapter 3, we provide description on the standard datasets for material recognition and also the basic feature classification pipeline. Chapter 4 and 5 are the core parts of the work, where we start from the basic learning-based descriptor and extend it to multi-scale version. Finally, we conclude the thesis in Chapter 7 and also give pointers on future work in Chapter 8.

Chapter 2

Related Work

In this chapter, we discuss two different strategies for feature descriptor on material recognition task: the hand crafted feature which usually bases on the local pattern defined by experts and then count the statistics to represent the instance; the learned features which encode the image at patch level by applying different unsupervised learning techniques. Afterwards we investigate general material classification pipeline including how we perform feature representation and classification.

2.1 Feature Descriptor for Material Recognition Task

In this section, we will briefly introduce some commonly used texture feature descriptors, including the simple but powerful LBP and its variants proposed long time ago, and also two recent proposed descriptors like kernel descriptor and LQP.

2.1.1 Local Binary Pattern

Local Binary Pattern (LBP) is a simple yet efficient feature descriptor for texture which labels the pixels of an image by thresholding the neighborhood of each pixel with the center value and consider the result as a binary number. The original LBP descriptor [34] defines the pattern for the image pixels by thresholding the 3×3 neighborhood of gray scale value. This defines a local texture pattern

$$T = t(v_c, v_1, ..., v_8)$$

where v_c denotes the pixel value at central position, $v_1, ..., v_8$ represent corresponded pixel values within the neighborhood. Since much of the information in the texture

pattern is preserved by the joint difference distribution [33], we can approximate the original pattern with

$$T \approx t(v_1 - v_c, v_2 - v_c, ..., v_8 - v_c)$$

In order to achieve gray-scale invariance, only the sign of the pixel-wise difference is considered, hence it leads to the final formulation as

$$T \approx t(sign(v_1 - v_c), sign(v_2 - v_c), ..., sign(v_8 - v_c))$$

where $sign(x)$ is the sign function. Figure 2.1 shows an example of how the pattern is computed, the left figure shows the original pixel value located at the 3×3 neighborhood, and it is converted into the binary pattern as shown in the right figure following rules mentioned above.

$$sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

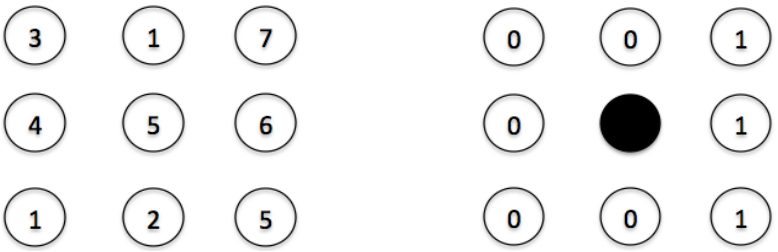


FIGURE 2.1: Example of LBP pattern. Image pixels within 3×3 neighborhood (left) and corresponding pattern by thresholding each pixel with the center value (right).

2.1.2 Extension to LBP

Many related approaches have been developed after the original LBP, including rotation invariant LBP, uniform LBP, multi-scale LBP and recently proposed Local Quantized Pattern.

Rotation Invariant LBP To achieve general rotation invariance, the 3×3 gridded neighborhood is extended to circular neighborhood. LBP is then defined as $LBP_{P,R}$, namely P equally spaced pixels on a circle of radius $R(R > 0)$ as illustrated on Figure 2.2. In particular, pixels which do not fall exactly in the center of pixels are estimated

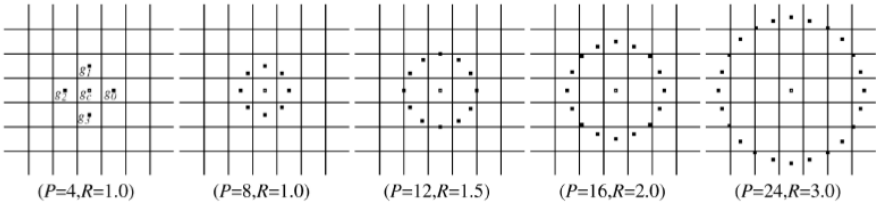


FIGURE 2.2: Example of circular neighborhood for different (P, R) [33].

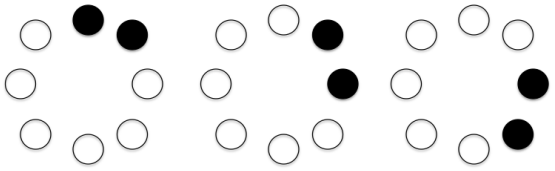


FIGURE 2.3: Example of rotation invariant binary pattern at $(8, R)$. Black and white circles correspond to bit values of 0 and 1. The three different binary patterns shown here are defined as the same one in rotation invariant LBP.

by interpolation. By assigning a unique identifier to each rotation invariant pattern, we can remove the effect of rotation.

Uniform LBP Another extension to LBP is the so-called uniform LBP. The idea is to distinguish the uniform pattern which is observed [40] to occur more commonly in texture images than others. A uniform local binary pattern is defined by binary pattern that contains at most two bitwise transitions from 0/1 when the bit pattern is traversed circularly. This greatly reduce the number of patterns from the original LBP.

Uniform, Rotation Invariant LBP To further incorporate the idea of the uniform binary pattern into the rotation invariant LBP, we get more compact variant named Uniform, Rotation Invariant LBP [33].

Multi-Scale LBP As pointed by [31], the most prominent limitation of the LBP descriptor is its small spatial support area which cannot capture large-scale structures that may be the dominant features of some textures. Muti-Scale LBP (MLBP) is introduced to tackle this issue. MLBP is accomplished by combining the information provided by multiple descriptor of varing (P, R) centering at the same pixel. Further extension has been made to the MLBP in [31].

2.1.3 Kernel Descriptor

Kernel Descriptor was first introduce in [4] and was later used for material recognition task [22]. The basic idea of kernel descriptor can be treated as a kernel view of orientation histograms like SIFT [30] and HOG [12]. By rewriting the feature vector of each pixel x as a weighted indicator vector $F(x) = m(x)\delta(x)$, where $m(x)$ is the magnitude of the image gradient at pixel x and $delta(x)$ is the orientation binning function at same location, one can then represent the feature at patch level via aggregating feature vectors over all the pixels as:

$$F(P) = \sum_{x \in P} \tilde{m}(x)\delta(x)$$

where $\tilde{m}$ is the normalized gradient magnitude and P represents an image patch. In consequence, this formulation yields the similarity measure between two image patches as:

$$F(P)'F(Q) = \sum_{x \in P} \sum_{x' \in Q} \tilde{m}(x)\tilde{m}(x')\delta(x)'\delta(x')$$

By introducing kernel functions $k_{\tilde{m}}(x, x') = \tilde{m}(x)\tilde{m}(x')$ and $k_\delta(x, x') = \delta(x)'\delta(x)$, we get a match kernel $K(P, Q)$ in kernel space:

$$K(P, Q) = \sum_{x \in P} \sum_{x' \in Q} k_{\tilde{m}}(x, x')k_\delta(x, x')$$

This framework can be extended for gradient, color and shape. In later application to material recognition task, two variants, namely the variance of gradient orientation and variance of gradient magnitude were proposed. Yet, one should note evaluating the kernels can be computationally expensive as image patches become larger. Hence, the author proposed the so-called sufficient finite-dimensional approximation to the match kernel. For further details, readers might refer to the original paper [4].

2.1.4 Local Quantized Pattern

Local Quantized Pattern (LQP) [23] extends the idea of manually designed features like LBP from the following aspects:

Geometrical Patterns The author investigated different geometrical patterns, including horizontal, vertical, horizontal-vertical, diagonal-antidiagonal and horizontal-vertical-diagonal-antidiagonal. This is illustrated in Figure 2.4 where we have highlighted the pixels to be considered in the neighborhood. Then the code is obtained by

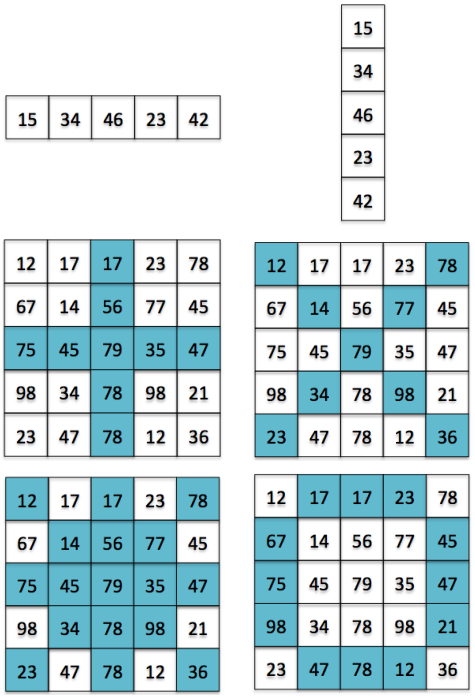


FIGURE 2.4: Some examples of geometrical pattern in LQP paper [23] including horizontal (H), vertical (V), diagonal (D), antidiagonal (A), horizontal-vertical (HV), diagonal-antidiagonal (DA), horizontal-vertical-diagonal-antidiagonal (HVDA) and disk-shaped region

comparing the intensity of pixel with either central pixel or the diametrically opposite pixel with binary or ternary coding similar to LBP.

Codebook Learning It use vector quantization for codebook learning which will be described later in this chapter and pick reasonable pattern size to allow run-time local pattern coding to be implemented by simple table lookup.

2.2 Feature Learning

In previous section,we have discussed manually designed feature descriptors for material recognition task. An alternative to this approach is doing feature learning which is usually patch-based and build on unsupervised learning framework. Here we provide a general overview of this technique. As described in[9], the patch-based feature learning framework can be decomposed into two phrases:

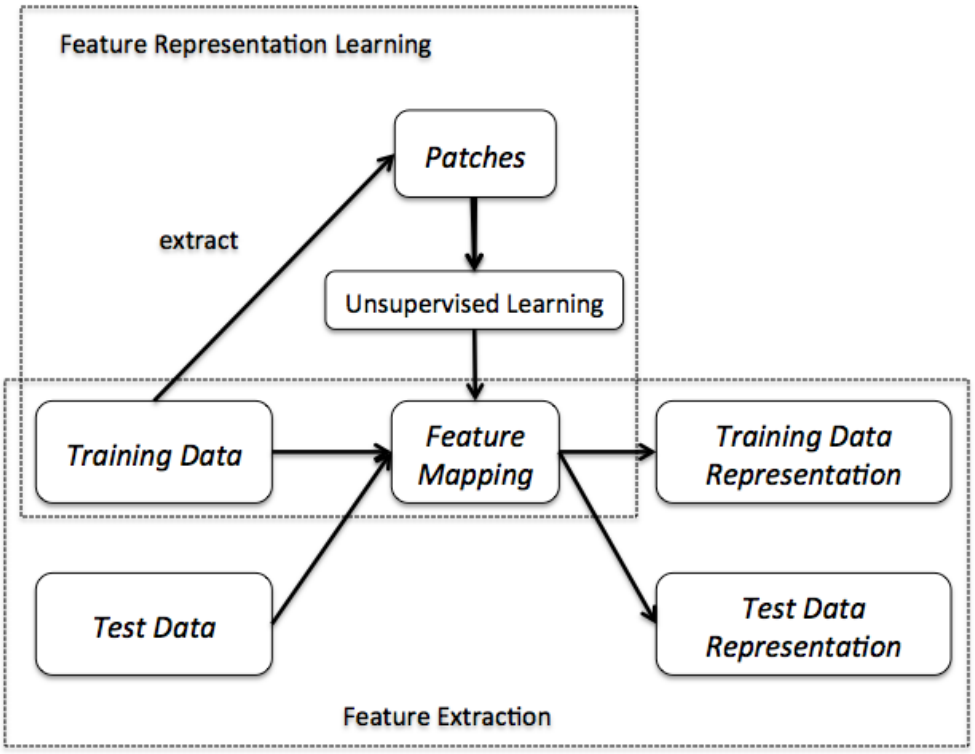


FIGURE 2.5: Illustration for feature learning framework.

Feature Representation Learning The first step is to extract random patches from training images and then pre-process these patches which involving techniques like normalization and whitening transformation [14] and then to apply unsupervised learning on the processed patches and acquire a feature-mapping (also known as codebook or dictionary).

Feature Extraction The next step is to perform feature extraction which involves extracting features from equally sub-patches covering the whole individual image, encode each patch with the dictionary and count the statistics from the codes to form the final histogram. In addition, one can also apply techniques like pooling to pool features over regions of image to reduce the number of feature values.

One commonly used unsupervised feature learning algorithm is K-means clustering. Other methods include sparse coding and restricted Boltzmann machine. All these will be covered in the following sections. In particular, we will briefly talk about a variant of sparse coding called the Spike-and-Slab Sparse Coding, which serves as the starting point of our work.

2.2.1 K-means Clustering

K-means clustering is a method for finding clusters and cluster centers in a set of unlabeled data and has been used in material recognition task for codebook learning [27] [11]. Given a dataset $\{x_1, ..., x_k\}$ with each $x_i \in R^n$, let K be the number of clusters, $\{\mu_k\}$ be the set of vectors representing the means (or centers) for corresponded clusters $\{C_k\}$. For each cluster assignment, the algorithm defines a criterion (also known as *distortion*) representing the sum of the square distances of each data point to the centers of its assigned cluster:

$$J = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

Given an initial set of centers, It alternates between two steps until convergence:

- For each center, assign each data point to the cluster that is closer to it than any other center.
- Compute means for each cluster and use it as the new center for the very cluster.

In application to the feature learning framework, the K centers found by the algorithm are treated as the dictionary, patches are coded as a K-dimension binary vector (also known as 1-of-K coding scheme), suppose a new patch x_i is assigned to cluster k, then:

$$x_{ij} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$$

Even though K-means has been widely used in various fields, it suffers several drawbacks, including:

Number of Clusters. In previous description, we assume the number of clusters k is already given, yet in practice, we have to pick it by our own. If k is larger than the true number of clusters in the data, it can happen that one cluster is split into two or more clusters; if k is smaller than the true number of clusters, then one several different clusters may collapsed into one cluster. Either way will produce a poor result.

Initial Assignment. The algorithm does not guarantee to converge to global optimal, thus it could be sensitive to initial assignment. A poor initial assignment can make the algorithm stuck into local minimal.

Similarity Measurement. K-means is based on Euclidean distance which greatly limit the type of data for clustering. Clusters found by the algorithm tend to have the shape of sphere, but for other cases the algorithm may again get stuck to local optimal. Further, it also not robust to outliers. These can be seen in the Figure 2.6.

Example Application In application to computer vision, the K-means algorithm is often used in "visual word models" where it's applied to raw patches or low-level descriptors [11] to learn a dictionary and then perform encoding. This is also one typical example of the so-called *vector quantization* technique.

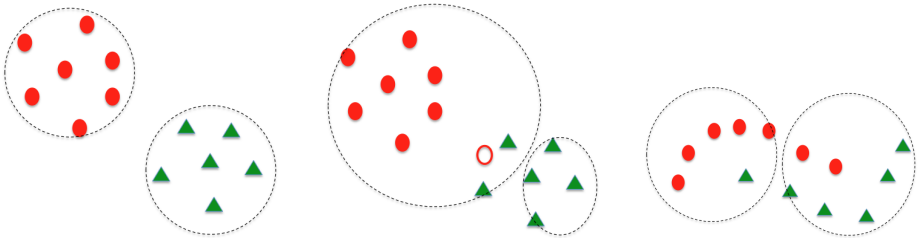


FIGURE 2.6: (Clusters found by K-means have the shape of sphere. (left) Outliers (marked by the hollow circle) may lead to poor clustering. (middle) Data with non-spherical shaper clusters may get pool local optimal solution from K-means.(right)

2.2.2 Sparse Coding

Though K-means is extremely fast in practice, the results it produced is often crude [8]. Researcher in machine learning community has sought to employ more powerful models to improve on it. Among these models comes the sparse coding model which consistently yields better results on some recognition tasks [5] [45]. Sparse coding was originally proposed by Olshauen and Field [35] as an unsupervised learning model of low-level sensory processing in humans. Given data $\{x_1, ..., x_k\}$ with each $x_i \in R^n$, it can be formulated as the following optimization problem:

$$\begin{aligned} \underset{b,a}{\text{minimize}} \quad & \sum_i \|x_i - \sum_j a_{ij} b_j\|_2^2 + \beta \|a_i\|_1 \\ \text{subject to} \quad & \|b_i\|_2 \leq 1, \forall j \in 1, ..., s \end{aligned}$$

$b = \{b_2, b_2, ..., b_s\}$ with $b_j \in R^n$ is called the *basis vector* and $a = \{a_1, ..., a_k\}$ with $a_i \in R^s$ is called the *activations*. The quadratic term in the optimization objective encourages each input x_i to be reconstructed as smoothly as possible (smoothing term) while the second term penalize the L_1 norm of the activations (regularization term).

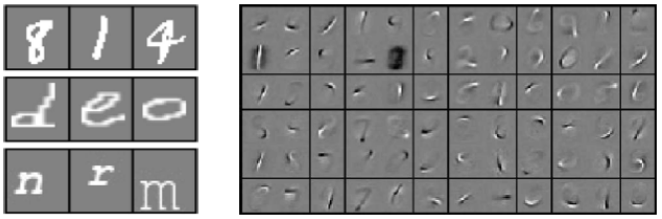


FIGURE 2.7: Examples taken from [37]. Left: Example images from the handwritten digit (top) / character (bottom) / font character (bottom). Right: Example words learned on handwritten digits with sparse coding model.

The regularization term enforces the activations to be sparse, which led to the name of the method.

This type of techniques is also known as *lasso estimator*, which belongs to a more general framework the *Shrinkage* method in the statistics literature as they reduce the value of weights by imposing a penalty on them

$$a^{shrinkage} = \underset{a}{\operatorname{argmin}} \underbrace{\|x_i - \sum_j a_{ij} b_j\|_2^2}_{loss} + \underbrace{f(a_i)}_{penalty}$$

whereas the first term measures the loss between the real value (target) and the predicted value (prediction) and the second term can vary to enforce different penalties (also known as regularizer), for example if f takes the form of L_1 norm , it corresponds to lasso; if f takes the form of L_2 norm, then it corresponds to ridge regression.

Example Application In computer vision, one related application of sparse coding is in the framework of *self-taught learning* [37]. The self-taught learning solves a supervised learning task given labeled and unlabeled data, but unlike semi-supervised learning [32], it does not assume the unlabeled data share the class labels or the generative distribution of the labeled data. The sparse coding mode is used to learn high-level representation which is supposed to be shared between the labeled and unlabeled data. Figure 2.7 is an example taken from the original paper [37]: the left side shows images from 3 different tasks including the handwritten digits, handwritten characters and font characters, sparse coding is applied to the handwritten digits to obtained the visual words, the right side shows some examples of these words. The strokes-like structures are shared between those different datasets.

2.2.3 Restricted Boltzmann Machine

Boltzmann machine (also known as Ising models in the statistical mechanics literature) is a particular type of undirected graphical with binary variables [16]. It is useful unsupervised learning especially for structured data yet hampered by its computational difficulties. Thus restricted Boltzmann machine (RBM) was introduced to partly resolve the problem, which does not allow intra-layer connections as illustrated in Figure 2.8.

RBM can be defined through the framework of *energy-based* models. Let the visible units in the model be x and hidden units h, then each configuration x, h can be associated with a scalar value energy $E(x, h)$ that can also be translated into a joint probability:

$$P(x, h) = \frac{e^{-E(x,h)}}{Z}$$

where $Z = \sum_{x,h} e^{-E(x,h)}$ is called partition function. In a RBM, energy function is defined as:

$$E(x, h) = -b'x - c'h - h'Wx$$

where W is the weight governing the connectivities between visible and hidden units and b, c are the offsets of the visible units. Given a training dataset $X = x_1, ..., x_n$, we can therefore define the log-likelihood over the data:

$$L(X, W, b, c) = \sum_{i=1}^{n} \log P(x_i)$$

One can then express the maximum likelihood estimator in terms of the conditional probabilities $P(h_i|v), P(v_i|h)$ which can be estimated via sampling methods. Contrastive divergence [20] has been further proposed to speed up the sampling process.

Example Application When applied to computer vision, the learned weights W can also be treated as the dictionary [8]. Further, RBM can be stacked into the so-called Deep Belief Network and has been applied to recognition task [21] where units in the higher layer are the actual learned features that are put into the final classifier as shown in Figure 2.9.

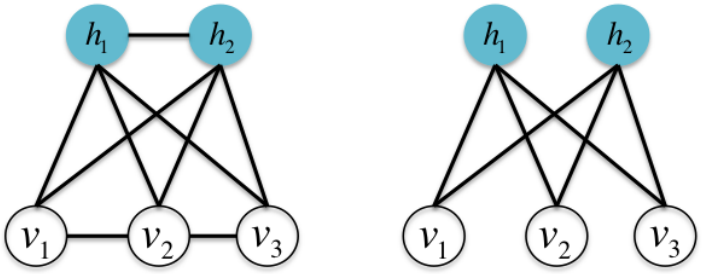


FIGURE 2.8: Illustration for Boltzmann machine (left) and restricted Boltzmann machine (right). Shadowed nodes h_1, h_2 denote hidden units and hallow nodes denote visible units.

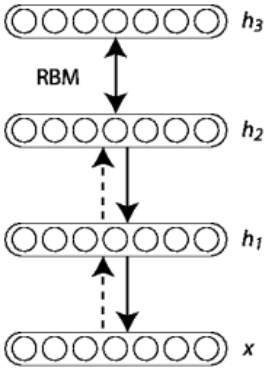


FIGURE 2.9: Illustration for Stacked restricted Boltzmann machine. h_i denote hidden units and x denote visible units.

2.2.4 Spike-and-Slab Sparse Coding

Spike-and-Slab Sparse Coding (S3C) model [17] is firstly used in the NIPS 2011 Workshop on Challenges In Learning Hierarchical Models' Transfer Learning Challenge and is the winning method for the challenge. As pointed out by the author, it is closely related to both the sparse coding model and the RBM: on one hand, it can be seen as a variant of sparse coding model with additional modification to the priors; on the other hand, it can also be reformulated into the framework of the RBM. The S3C model serves as the basic building block of this thesis work, we will discuss it in more details in later chapter.

2.3 Material Classification Pipeline

So far, we have discussed various ways to computer features for image data either with hand-crafted descritpor or learned feature representation. We now turn to the basic framework used for material recognition task as shown in Figure 2.10 where we want

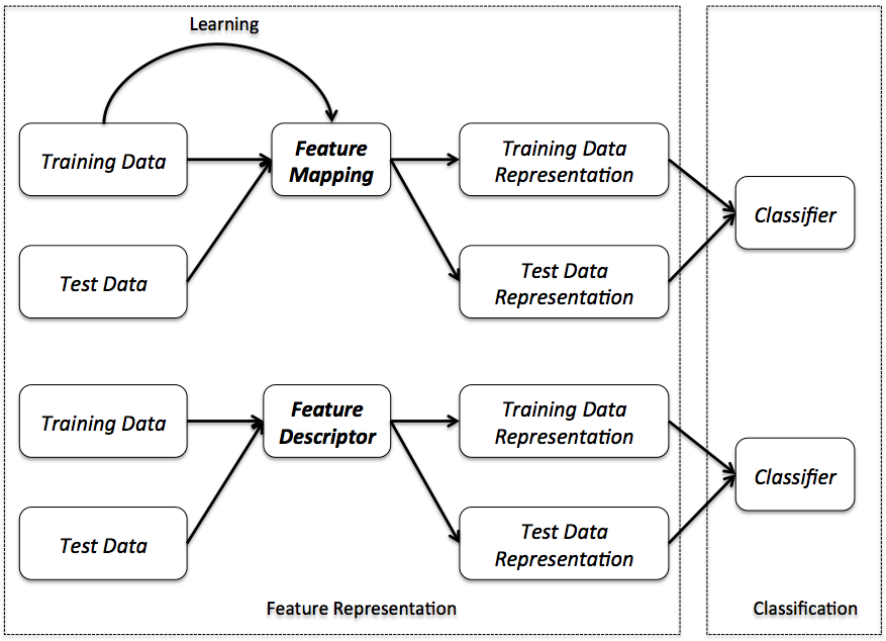


FIGURE 2.10: Illustration of feature classification pipeline. The upper part represents the pipeline for learned feature, and the bottom part represents the one for hand-crafted descriptor.

to break it into two stages, the feature representation in which we use the features as described in previous section to code image data and classification in which we usually train a classifier on given data and make prediction for new data.

2.3.1 Feature Representation

We have seen both hand-crafted descriptor and learned features can code image data, yet we should note there are some significant difference between these two approaches, and that's why we separate the pipelines for them in the general framework.

For hand-crafted descriptor, we directly obtain the representation by counting the histogram of the descriptor. For example, the basic LBP feature first applies the binary coding over all the pixels so that each pixel get a unique code, and then it counts the number of occurrence for these codes and eventually form the histogram for the whole image.

For learned feature, typically we first obtain a dictionary from the learning algorithm over sampled patches and then we shall densely extracted patches and code each patch with the dictionary. Afterwards, it goes similar to the hand-crafted descriptor to form the final histogram.

2.3.2 Classification

So far we have shown various ways to compute feature and how to represent the data. We now start to discuss how to use these obtained data for classification. Given a dataset $\{X, y\}$ where X represent the features and y represent the labels, we want to predict y based on X. In most cases, we are supposed to given part of the data known as *training data* $\{X, y\}_{train}$, and we train a classifier from the training data and give predictions on new data known as *test data* based on the output of the classifier. This is known as *supervised learning* in field of machine learning.

One of the most popular classifier is the Support Vector Machine (SVM) [10] which has shown superior performance in wide range of applications. In this section, we focus on SVM and in particular we also cover the *kernel trick* which extend the original SVM formulation. Yet one should note there are different models for classification, one related example would be the aLDA model used in [29] where a hierarchical probabilistic model is build on the feature data and is used for classification.

2.3.2.1 Basics

Starting from the simplest binary classification case, suppose we are given a set of training data $(x_1, y_1), ..., (x_m, y_m)$ with $x_i \in R^N$ and $y_i \in \{-1, +1\}$, where N is the dimension for input data, linear classifier can be defined as a mapping $f : R^N \rightarrow \{-1, +1\}$:

$$f(x) = \text{sgn}(w \cdot x + b) \begin{cases} +1 & \text{if } w \cdot x + b \geq 0 \\ -1 & \text{if } w \cdot x + b \leq 0 \end{cases}$$

Geometrically, the *hyperplane* (also known as *decision boundary*) defined by $w \cdot x + b = 0$ separate the input space into two *decision regions*, where *margin* is defined as the perpendicular distance between the decision boundary and the closest of the data points as illustrated in Figure 2.11.

2.3.2.2 SVM: Maximum Margin Classifier

The support vector machine is a linear classifier that seeks the decision boundary to maximize the margin, and the points on the margin are called *support vectors*. For linear separable case, this can be formulated as solving the following constrained minimization

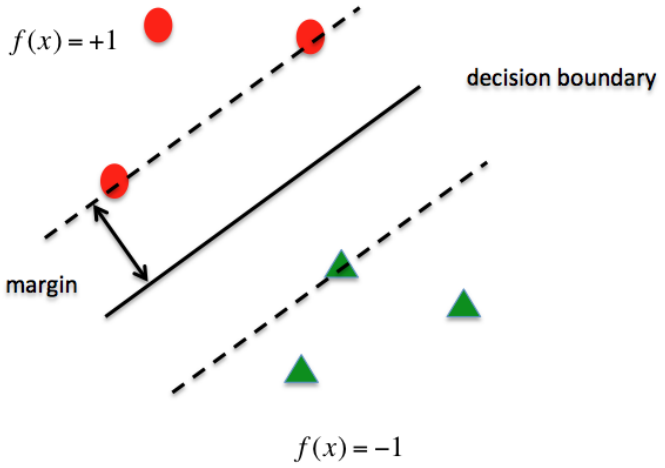


FIGURE 2.11: Illustration of binary linear classification.

problem:

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(w \cdot x_i + b) \geq 1 \end{aligned}$$

For linear inseparable case, *slack variables* ξ_i are introduced to allow the data points to violate the margin with proper penalty, which leads to the following formulation:

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad \forall i = 1...n \end{aligned}$$

In both cases, the problem can be abstracted as quadratic programming with linear inequality constraint which is a convex optimization problem [6] and the solution can be described with Lagrange multipliers. Without loss of generality, we study the case of linear inseparable case where its Lagrange function can be expressed as:

$$L(w, b, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(w \cdot x_i + b) - (1 - \xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i$$

By setting the derivatives to zero with respect to w, b, ξ, we get the minimizer to the problem as:

$$w = \sum_{i=1}^{n}\alpha_i y_i x_i, \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \quad \alpha_i = C - \mu_i$$

Plugging the minimizer into the original Lagrange function, we reach the so-called *Lagrange dual objective function*:

$$D(\alpha, \mu) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

This serves as a lower bound on the original objective function. Since we already know it's a convex optimization problem and we can further valid the so-called *Slater's condition* [6] for it, we can compute the the maximizer from the dual objective function which equals to the minimizer and is exactly the optimal solution to both primal and dual problem.

In particular, the optimal solution follows the **Karush-Kuhn-Tucker (KKT) condition**, among which we can get:

$$\alpha_i [y_i (w \cdot x_i + b) - (1 - \xi_i)] = 0$$

which yields

$$\begin{aligned} \alpha_i > 0 \quad if \quad y_i (w \cdot x_i + b) - (1 - \xi_i) = 0 \\ \alpha_i = 0 \quad if \quad y_i (w \cdot x_i + b) - (1 - \xi_i) < 0 \end{aligned}$$

Note the solution to $y_i (w \cdot x_i + b) - (1 - \xi_i) = 0$ represents the points on the margin and only these points have non-zero weights which characterize the decision boundary, and thus are called support vector.

2.3.2.3 Kernels

Note the original SVM only deals with linear separable case, and in order to extend its capability, kernel method was introduced to tackle nonlinear case. The basic idea behind kernel method can be described as to embed the data into a space where the patterns can be discovered as linear relations [39] as illustrated in Figure 2.12. Mathematically, we can easily replace the $x_i \cdot x_j$ with proper kernel $k(x_i, x_j)$ as in the dual formulation.

Linear Kernel The simplest kernel function is linear kernel as $k(x_i, x_j) = x_i' x_j$ where the feature mapping function is just the identity function $\phi(x) = x$.

Radius Basis Function Kernel One of the most commonly used kernel is Radius Basis Function (RBF) kernel (also known as Gaussian kernel) as $k(x_i, x_j) = exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$. The kernel function is related to the radial basis function network where basis function

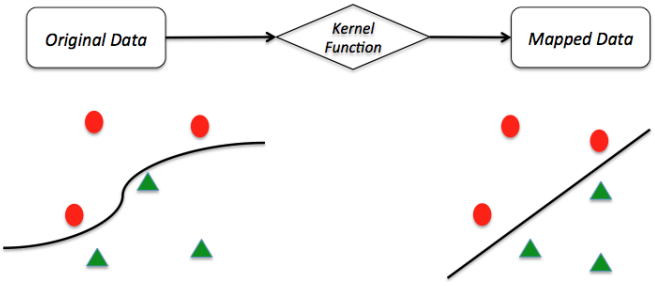


FIGURE 2.12: Illustration of kernel methods where data that is linear inseparable in original space has been mapped into a new space in which new representation is linear separable.

depends only on the radial distance from a center and therefore also referred to as the RBF kernel.

χ^2- Kernel Another kernel which has shown superior performance to the RBF kernel on material recognition task [18] [7] is χ^2- kernel as

$$k(x_i, x_j) = exp\{-\gamma\chi^2(x, y)\}$$
$$\chi^2 = \sum_j \frac{\|x_{ij} - y_{ij}\|^2}{x_{ij} + y_{ij}}$$

This kernel is also known as the *generalized RBF kernel* as it combines the benefits of both the homogeneous additive kernels (the χ^2- kernel) and the RBF kernel [44]. In particular, the χ^2- distance measure is designed to compare histograms which are heavily used for most feature descriptors like SIFT.

2.4 Discussion

In this chapter, we reviewed both manually designed feature descriptors and feature learning techniques. While the former method has been used a lot in previous studies, the learned feature has gained popularity among computer vision researchers. From theoretical side, learned feature is more adaptive and can be obtained in an automatic way. From practical side, as we will show in the later chapter, our proposed learned feature can outperform the manually designed descriptor on standard material recognition benchmarks.

Chapter 3

Datasets

In this chapter, we will first give an overview of standard material and texture databases. Then we will discuss the differences between these databases and the motivations behind them.

3.1 Datasets

3.1.1 CUReT

Columbia-Utrecht Reflectance and Texture Database (CUReT) was originally proposed to investigate the visual appearance of real-world surfaces and the dependence of appearance on imaging conditions [13]. It consists of 61 texture classes and each has been imaged under 205 viewing and illumination conditions. Hence the main challenges come from the effects of specularities, inter-reflections, shadowing and other surface normal variations [41]. Figure 3.2 shows the variations with respect to pose and illumination change for one sample material. The original image size is 640×480, Varma et al [41] later on crop the image into 200×200 and use 92 of the 205 views in their experiment. This cropped database is also used in the later experiment in [18]. This is illustrated in Figure 3.1 where the original image also include background but the cropped image does not.

3.1.2 KTH-TIPS

The major limitations of the CUReT database are a lack of significant scale change and limited in-plane rotation. The KTH-TIPS database (KTH is the abbreviation of the university, and TIPS stands for Textures under varying Illumination, Pose and Scale)

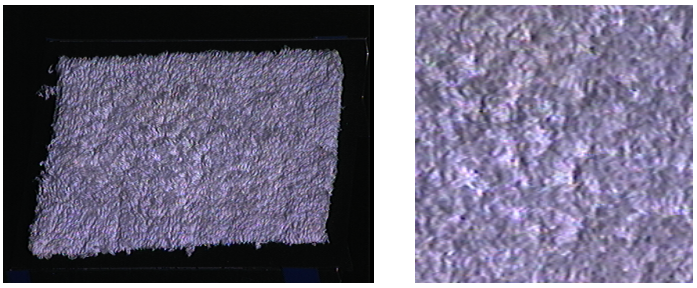


FIGURE 3.1: Example image from the original CUReT database (left) and corresponded cropped image used in [41] (right).

FIGURE 3.2: Examples taken from [18] for CUReT database. Images of white bread demonstrate the variation of appearance of a 3D texture as the pose and illumination conditions change.

[18] is just designed to further incorporate scale and rotation information to supplement CUReT database. It contains 10 materials also present in the CUReT database, and imaged at 9 distances from the camera to give equidistant log-scales over two octaves, at which images were captured using three different directions of illumination (front, side and top) and three different poses (central, 22.5° turned left, 22.5° turned right)t giving $3 \times 3 = 9$ images per scale, and $9 \times 9 = 81$ images per material.

Based on the KTH-TIPS database, the same group proposed KTH-TIPS2 database [7] to study material recognition with a special focus on generalization to novel instance of materials by imaging multiple different samples of different materials. It includes more than 4000 images from 11 material categories, and each category has 4 physical, planar samples. All the samples are imaged in the similar way as in the original KTH-TIPS database.

In particular, when removing the background from the original image, images are manually cropped into 200×200 to be consistent with the experiments in [41]. However, there are still some exceptions when either height or width of the original image does not fit the requirement, in this case, it is cropped in a way to keep the same area.

FIGURE 3.3: Examples of images in KTH-TIPS2 database.

3.1.3 MPI-VIPS

Even though the KTH-TIPS2 database introduce more samples per category, it is still far away from real world scenario which usually have many different instances, various settings, large intra-class variation and many more material classes, yet acquisition of such dataset for learning is rather tedious. While grabbing large databases from the Internet has been a promising direction pursued in recent years, it often comes with a bias and is less appropriate fro domains for which there is an underlying parametric structure that should be learned. Li and Fritz [28] approach this problem via rendering more training data from 3D models and proposed the MPI-VIPS (MPI is the abbreviation of the institute, and VIPS denotes for Virtual texture under varying Illumination, Pose and Scales) database.

The work is motivated by the availability of material shader from commercial suppliers to the computer graphics community as well as Internet resources. Therefore the authors collect a set of shader that match the material classes from the KTH-TIPS2 database in order to facilitate related comparison. They also used Autodesk 3ds Max to the rendering and followed the scene setting in the KTH-TIPS2 database.

In particular, the author proposed the so-called *Manifold Alignment* to match the appearance between rendered and real images which corresponds to picking proper rendering parameter. The experimental results indicated that such alignment step is crucial for successfully utilizing rendered data.

3.1.4 Flickr Material Database

The study on MPI-VIPS shows the feasibility of getting more training data by rendering, yet current limited number of shaders still restricted the scale of this application. As said in previous section, the alternative would be collecting data from the Internet. One such attempt is the recent proposed Flickr Material Database (FMD) by Liu et al [29] where material images are manually picked photos from Flickr.com under unknown real-world conditions, which also makes it very challenging for current systems [29][22][28][36]. There are 10 common material categories with 100 images per category, 1000 images in total as shown in Figure 3.5. For the 100 images for each category, 50 of them are close-up meaning textured objects are closed to the camera and the rest of them are of regular views. Images are photoed at resolution of 512×384.

FIGURE 3.4: Examples of images in MPI-VIPS database.

FIGURE 3.5: Examples of images in FMD database.

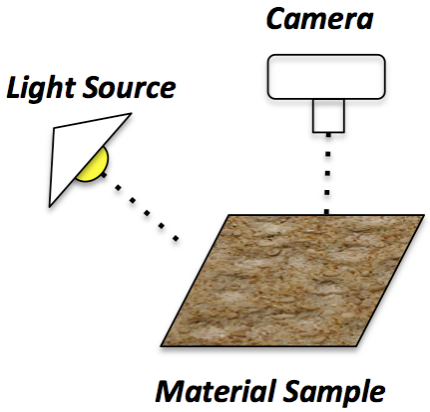


FIGURE 3.6: Example of labotory condition for imaging material samples.

3.1.5 Discussion

We have briefly reviewed on several standard material databases in previous section. Apparently, every newer database tries to improve the former database in one way or the other. In this part, we want to briefly discuss these places and discuss about some general trends about research on material recognition task.

3.1.5.1 Instance Identification to Category Classification

Early CUReT databases is used for instance identification task, meaning for each category there is only one material or texture sample although the conditions under which the images are taken are varied. In contrast, KTH-TIPS, KTH-TIPS2 and MPI-VIPS have included more samples per category, which introduces larger inter class variations and significantly adds difficulties to recognition task.

3.1.5.2 Planar-based Image to Real World Scene Image

Typical material or texture databases like CUReT and KTH-TIPS databases are planar-based image which means material images are usually taken under controlled conditions and material samples are mostly placed closely to the camera as illustrated in Figure 3.6. These conditions help to reduce the noise to the final classification task, yet make the task unrealistic. The more recent FMD database came to just address this issue where images collected are taken under unknown conditions, and there are lots of images that depict complete object characterized by certain material instead of only pictures of very restrictive local surface.

Chapter 4

Feature Learning for Material Recognition

In chapter 2 and 3, we reviewed related work and introduced some standard datasets on material and texture classification. In this chapter, we will focus on one specific model for feature learning, namely the Spike-and-Slab Sparse Coding (S3C). We begin by briefly recapping some basics in probabilistic graphical model. In particular, we discuss EM algorithm and variational methods which have been used for the later inference for the S3C model. Next we present description for the S3C model and how to do learning for the model. Afterwards, we also presents details on the experimental results for single scale experiment. Finally, we discuss results and conclusion.

4.1 Probabilistic Graphical Model: a review

In graph theory, a graph generally comprises vertices (also called nodes) connected by edges (also known as arcs or links). In a probabilistic graphical model, each node represents a random variable (or group of random variables), and the links express probabilistic relationships between these variables [2]. Two major classes are commonly used, namely directed graphical models (also known as Bayesian networks), where the edges have particular directionality indicated by arrows and undirected graphical models (also known as Markov random fields). In this thesis, we will focus on the directed graphical model.

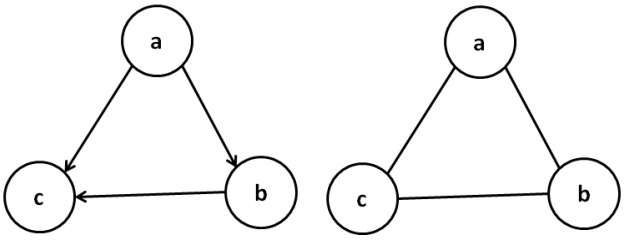


FIGURE 4.1: Illustration of probabilistic graphical models.

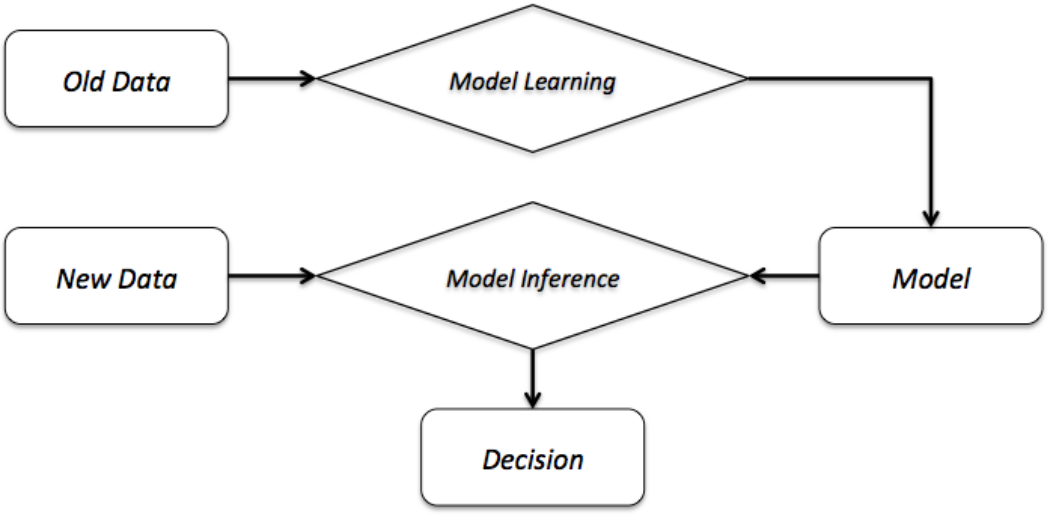


FIGURE 4.2: Pipeline of model learning and inference in probabilistic graphical model where shaded circles represent hidden variables.

4.1.1 Model Learning and Inference

Applying a graphical model to a real world problem can be broken into two major steps as illustrated in Figure 4.2, namely model learning and inference. By model learning, we acquire a model to represent the problem and estimate the parameters in the model from the data at hand. Model inference usually comes after the model learning by computing the probabilistic estimation for new incoming data from the obtained model.

4.1.2 EM Algorithm and Variational Inference

In this section, we begin by briefly reviewing *expectation-maximization* algorithm which is commonly used for finding maximum likelihood estimators in latent variable models. We then start from solving Gaussian mixture mode and eventually generalize to general form of EM algorithm.

4.1.2.1 Gaussian Mixture Model and EM Algorithm

Given a dataset $\{X\}$, suppose there are K different sources which we would like to model with Gaussian distribution, then the whole data can be modeled with Gaussian mixtures. More formally put, each individual source can be characterized by the probability distribution

$$N(x|\mu_i, \Sigma_i)$$

where i denotes the index of the source, μ_i and Σ_i represents the mean and covariance for the very source. The Gaussian mixture distribution can be written as a linear superposition of Gaussian

$$p(x) = \sum_{i=1}^{K} \pi_i N(x|\mu_i, \Sigma_i)$$

where the weight π_i (also known as *mixing coefficient*) can be treated as the prior probability of picking the i^{th} source $\pi_i = p(z = i)$ (z denotes current index of source) and therefore follow the constraints

$$\sum_{i=1}^{K} \pi_i = 1, \quad 0 \leq \pi_i \leq 1$$

In general, the prior π_i and model parameters μ_i and Σ_i are unknown. When applying the model to a dataset, we would like to answer two questions:

- How to characterize these different sources of Gaussian? (model learning)
- What is the assignment for each data point? (inference)

which exactly corresponds to our two steps for learning and inference: during model learning, we estimate the parameters μ_i, Σ_i and the prior for each source; during inference, we figure out the posterior $p(z = i|x)$ for each data point (also known s the responsibility). Note the assignment is associated with a probability value instead of determined static assignment.

Expectation-Maximization (EM) algorithm can be applied to the model:

- **Initialization** Initialize the parameters $\theta = \{\mu_i, \Sigma_i, \pi_i\}$
- **E step** Evaluate the posterior $p(z = i|x, \theta)$ using current parameters with Bayes Rule $p(z|x, \theta) = \frac{p(x|z,\theta)p(z)}{p(x|\theta)}$

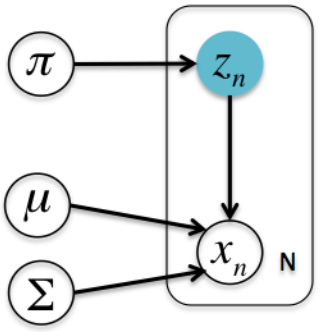


FIGURE 4.3: Graphical model for Gaussian Mixture model.

- **M step** Estimate the parameters using current posterior by maximize the log likelihood
- **Check Convergence** Evaluate the convergence of parameters and likelihood, if not return to E step.

The final output from the EM algorithm will be the estimated parameters, yet one should note the assignment i.e. the posterior can be obtained by re-run the E step with the estimated parameters.

4.1.2.2 From Standard EM to Variational EM

The EM algorithm is a powerful framework and can be applied to many different models with latent variables. Note in the formulation of posterior

$$p(z|x,\theta) = \frac{p(x,z,\theta)p(z)}{p(x|\theta)}$$

where the marginal likelihood can be written as

$$p(x|\theta) = \sum_z p(x,z|\theta)$$

which could be potentially intractable due to the structure of summation. One option would be to find some good approximate solution to speed up the computation which the *variational method* is used exactly for. The terminology origins from the techniques in calculus of variations where the central subject is *functional* that takes a function as input and outputs the value of functional.

Often the true posterior is replaced by variational distribution $q(z)$. Then the log likelihood function can be decomposed into

$$\begin{aligned}\ln p(x|\theta) = & \ln \tfrac{p(x,z|\theta)}{p(z|x,\theta)} \\ = & \ln[\tfrac{p(x,z|\theta)}{q(z)} \tfrac{q(z)}{p(z|x,\theta)}] \\ = & \textstyle\sum_z q(z) \ln[\tfrac{p(x,z|\theta)}{q(z)}] - \sum_z q(z)[\tfrac{p(z|x,\theta)}{q(z)}]\end{aligned}$$

where we can define these two parts as

$$\begin{aligned}L(q,\theta) &= \sum_z q(z) \ln[\frac{p(x,z|\theta)}{q(z)}] \\ KL(q(z)||p(z|x,\theta)) &= -\sum_z q(z)[\frac{p(z|x,\theta)}{q(z)}]\end{aligned}$$

where $KL(*)$ represents Kullback-Leibler divergence (KL divergence) measuring the difference between two probability distributions. Note KL divergence is semi-definite, i.e. $KL(I) \geq 0$, therefore the $L(*)$ forms a lower bound for the true likelihood

$$\ln p(x|\theta) = L(q,\theta) + KL(q(z)||p(z|x,\theta)) \geq L(q,\theta)$$

with equality if and only if the variational probability $q(z)$ equals to the true posterior $p(z|x,\theta)$. The lower bound L is then used in the EM algorithm for evaluation which also leads the name *variational EM* [38] for it. One related example can be found in [3], where variational EM is applied to solve the *Latent Dirichlet Allocation* (LDA) model.

In particular, there are different strategies for picking the approximate distribution $q(z)$, and among them, we will discuss *structured variational approximations* [24] which is applied to the model in this thesis. The idea of structured variational approximations can be described as optimizing the functional L over a family of coherent distributions Q where the family is chosen to be computationally tractable. One typical example of this method is the *mean field approximation* where the variational distribution is decomposed into a fully factorized model

$$Q(X) = \Pi_i(x_i)$$

The S3C model also applies structured approximation at inference stage, but it uses richer representation than the mean field approximation which we will discussed in more details in the following section.

4.2 Spike-and-Slab Sparse Coding Model

While we have seen broad application and success of feature learning techniques in object recognition, material recognition still relies on hand-crafted features. The appearance of material classes seem special in many ways. First of all, the samples seem to obey a stronger manifold assumption, as the appearance varies rather smoothly w.r.t. changes in lighting direction, orientation and scale. For objects, more drastic changes can occur due to the more pronounced 3d structure.

Spike-and-Slab Sparse Coding (S3C) by Goodfellow at el [17] has been recently proposed to combine the merits of feature learning methods like sparse coding and RBMs. It has shown to perform superior to previous feature learning technique as well as turned as best performer on a recent transfer learning challenge where learning data manifolds is key. Therefore, we use it in our investigation to feature learning for material recognition. As we will show qualitatively as well as quantitatively, it seems particularly well suited to capture the data distribution of natural materials.

In this section, we first introduce the S3C model as our basic method for feature discovery, Then we extend the model for multi-scale feature learning with two different strategies.

4.2.1 Model Description

The Spike-and-Slab Sparse Coding (S3C) model was proposed by Goodfellow at el [17]. It's a two-layer generative process: the first layer is a real-valued D-dimensional visible vector $v \in R^D$, where v_d corresponding to the pixel value at position d; the second layer consists of two different kinds of latent variables, the binary *spike* variables $h \in \{0,1\}^N$, the real-valued *slab* variables $s \in R^N$. The spike variable h_i gates the slab variable s_i, and those two jointly defines i^{th} hidden unit as $h_i s_i$. The process can be more formally described as follows:

$$\forall i \in \{1,...,N\}, d \in \{1,...,D\}$$
$$p(h_i = 1) = \sigma(b_i)$$
$$p(s_i|h_i) = N(s_i|hi\mu_i, \alpha_{ii}^{-1})$$
$$p(v_d|s,h) = N(v_d|W_{d:}(h \circ s), \beta_{dd}^{-1})$$

where σ is the logistic sigmoid function, b is a set of biases on the spike variables, μ and W govern the linear dependence of s and h and v on s respectively, α and β are diagonal

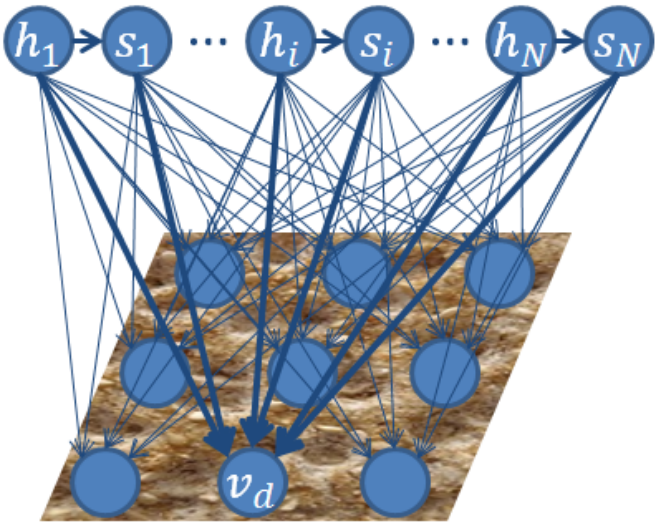


FIGURE 4.4: Graphical illustration of S3C model, where v_d denotes visible units, h_i and s_i represent hidden variables (spike and slack variables respectively), directed arrows stand for dependency.

precision matrices of their respective conditionals, and $h \circ s$ denotes the element-wise product of h and s. Columns of W is constrained to have unit norm, α is restricted to be a diagonal matrix and β to be a diagonal matrix or a scalar. In particular, W can be interpreted as a series of filters which can be used sparsely to represent the data.

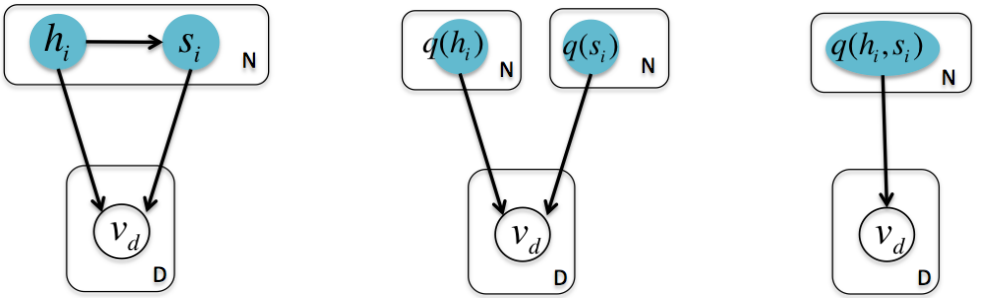
4.2.2 Model Learning

Variational EM algorithm is used for model learning. In the E-step, we only compute a variational approximation to the posterior rather than the posterior itself. Then in the variational E-step maximize the energy functional with respect to a distribution Q over the unobserved by minimize the Kullback-Leibler divergence:

$$D_{KL}(Q(h, s||P(h, s|v)))$$

where $Q(h, s)$ is drawn from a restricted family of distribution to ensure that Q is tractable. In particular, in the original paper, the S3C's author apply a structured family for Q, i.e.

$$Q(h, s) = \prod_i Q(h_i, s_i)$$



(a) Original S3C model.

(b) Mean field approximation to S3C model.

(c) Structured approximation to S3C model.

FIGURE 4.5: Graphical representation for the approximation scheme

instead of the fully factorized mean field approximation

$$Q(h, s) = \prod_i Q_h(h_i) Q_s(s_i)$$

This is illustrated in Figure 4.5 where variational distributions for h_i and s_i are decoupled in the mean field approximation (left) whereas they are modeled jointly with another variational distribution (right).

4.3 Experiments

In this section, we present detailed information for the experiments on single-scale S3C model, together with quantization-based color patches, LBP and its several variants. In particular, we shall discuss implementation details on how we compute these features in practice. Afterwards we present the experimental results and draw some conclusions.

4.3.1 Implementation Details

We divide the implementation details for computing features in two parts: the first part will cover the contents for S3C feature which is mainly implemented in Python; the second part will discuss the details for other descriptors.

FIGURE 4.6: Demonstration of selection of data structure. Tensor or multi-dimensional array is only possible for images of same sizes (Left), while it is hard to fit for the other case (Right).

4.3.1.1 S3C Feature

Although the author of S3C model kindly provided the source code for the original experiment for transfer learning on object recognition task, there are still many places we need to adapt for our specific settings.

Implementation Framework We start with the framework from Goodfellow [17] which is built on a Python library that supports define, optimize and evaluate mathematical expressions involving multi-dimensional arrays efficiently called Theano [1]. One major advantage coming from Theano is that it make use of Graphical Processing Unit (GPU) which speeds up to 140 times faster than CPU for data-intensive calculations.

Data Structure The original data interface in the package is hard-coded with tensor (also called multi-dimensional array) for object recognition task on small size image like CIFAR-10 and CIFAR-100 [25], in which all the images has the same size of 32. Although this could be adapted straightforward for FMD dataset which has similar settings, it does not work with KTH-TIPS2 database, where images are only in roughly equal area but with different width and length for lots of instances as illustrated in Figure 4.6. Thus we choose list structure in Python to replace the tensor in our experiment, where each entry is a tensor to store single color image.

Runtime Memory Consumption As the framework makes use of the GPU and GPU usually possesses very limited amount of memory compared with CPU-RAM (Random Access Memory), we have to be careful when setting up the configuration of the pipeline. For the S3C model, a rough estimate of runtime memory consumption can be the total place taken by the visible units and the two types of hidden units, i.e. the spike variables and slab variables. Suppose the image size is $c \times r$, patch size is fixed at $p \times q$, patch stride is s and number of both types of hidden units is N, if we perform all the

GPU Type (GeForce GTX)	460	480	560	580	660	680
Memory Capacity (MB)	1024	1536	1024	1536	2048	2048

TABLE 4.1: Memory capacity for some commonly used GPU.

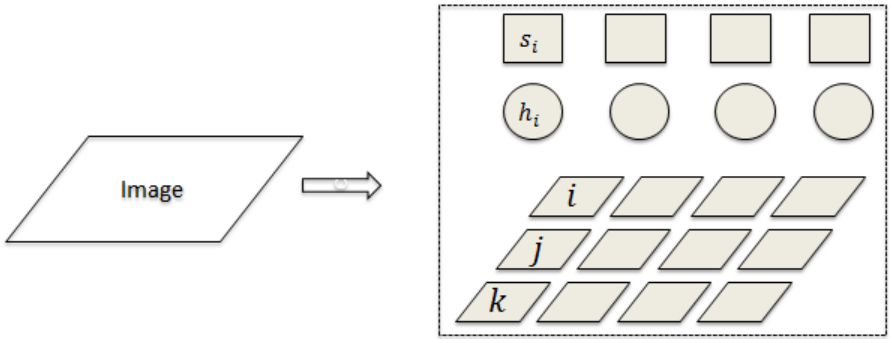
computation for patches densely extracted from a single *color* image in a batch, then the total runtime memory can be estimated as: $c/s \times r/s \times (3pq + 2N)$. An initial configuration on FMD database with float32 precision (4 bytes per number), where image size is 512×384, patch size is 6×6, patch stride is 1, N is 1600, makes 2.4 gigabytes which already exceeds the maximum amount of memory in GTX 480 used in our experiment. Thus to fit the model into the framework, we have to make some improvements to the basic model, one alternative kindly pointed out by Goodfellow is to split the patches into several batches, and then perform the operation for one batch at a time. By doing so, we only store visible variables and corresponded hidden variables for each batch and we also gain the benefits to be able to vary the batch size flexibly to fit a reasonably large model.

4.3.1.2 Other Features

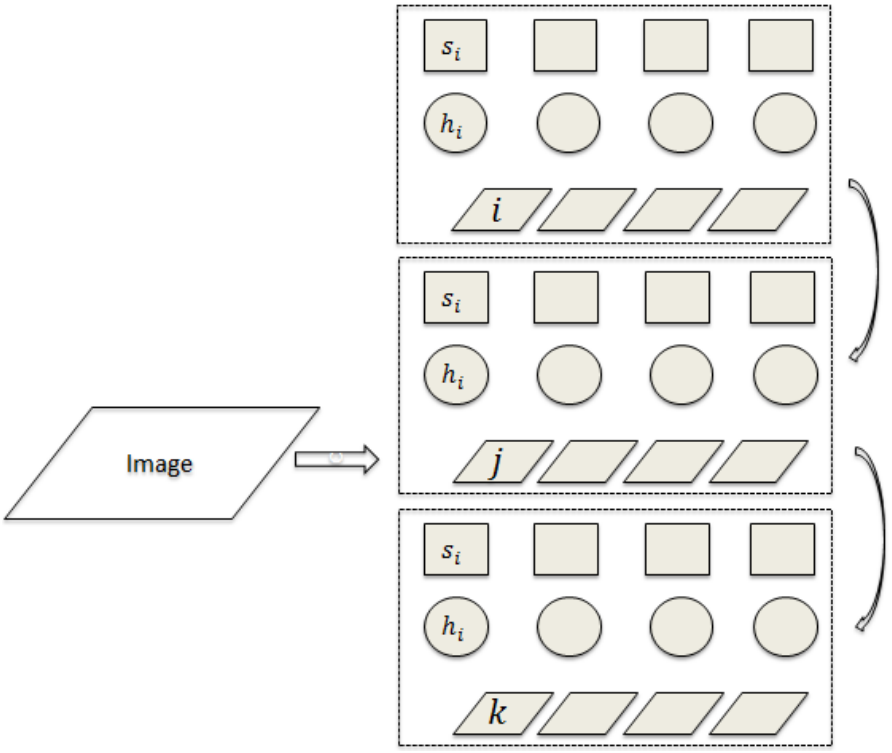
Besides the python code for S3C model, we also use MATLAB for computing LBP and quantization method for color patches.

LBP feature we use the code from [19]. The code supports computing the basic LBP and its several variants including the u-LBP, ri-LBP and ri,u-LBP with specified radius R and number of sampled points N. In practice, one needs to first define the mapping with (R, N) and then calls the *lbp.m* function to compute the descriptor. Since we iterate the computation over all the images, it is faster to predefine the mapping outside the loop.

Color Patches For color patches, we use a lot of the utility functions from VLFeat [43]. In particular, the library provides implementation of Elkan's algorithm [15] for k-means which is typically faster than the standard k-means algorithm at the cost of potential larger storage. In addition, as the operations of feature extraction for each image are independent between each other, we use parfor (execute code loop in parallel) in MATLAB to speed up the computation.



(a) Basic runtime memory model



(b) Improved runtime memory model

FIGURE 4.7: Runtime memory model. Objects in the block denote variables stored during runtime. In details, the lines of rectangles in the top row represent hidden slack variable, circles represent hidden spike variables, tilted rectangles represents densely extracted patches placed at each grid position, and each block represent the total runtime memory consumption. The basic runtime memory model (shown on the top) simply store all the units in a batch while the improved runtime memory model split patches into small batches and only perform batch operation on one batch at a time which greatly reduce the runtime memory consumption.

4.3.2 Experimental Setup

In order to compare the learned features with hand-crafted features, we provide recognition rates using standard SVM classifiers on the coded data from the S3C model with Color, single-scale LBP, local quantization pattern (LQP)[23]-a recently introduced variant of LBP descriptor and kernel descriptor which has been shown of state-of-the-art performance on FMD database, for single scale experiment. For multi-scale approaches we consider: MLBP, Stacked S3C (S-S3C), and Joint S3C model (J-S3C) for multi-scale experiment. In all our experiments we use 1600 hidden units as proposed by the authors of S3C.

4.3.3 Experimental Results

For this group of experiment, we apply S3C model directly on the data with single scale only. In details, we vary the patch size of $\{6 \times 6, 12 \times 12, 24 \times 24\}$ and we compare its performance with color-patch, LBP and several variants of LBP, including uniform-LBP (u-LBP), rotation invariant-LBP (ri-LBP) and rotation invariant, uniform-LBP (ri,u-LBP) as described in the related work. The final experimental results are shown in Table 4.8.

From the results on KTH-TIPS2 database: first, we see that when using the S3C for feature representation, different patch sizes lead to different results for both linear and the $exp - \chi^2$ kernel, and patch-size at 12×12 gives the best number for both kernel; second, when comparing the results from the S3C model with those from the descriptors, the S3C works better (for all the 3 patch sizes) than all the descriptors when only using linear kernel while for $exp - \chi^2$ kernel, it becomes more complex as the S3C does a better job than the color-patch, ri-LBP and ri,u-LBP but u-LBP produces the best number 67.15% in this competition; next, if we combine the results in both columns, the S3C is the actual winner - the result from the model at patch-size of 12×12 get the best number with accuracy 71.25% which is superior to any descriptors with any kernels by a fairly significant margin.

From the results on FMD database: alone with results from S3C model, we note again different patch sizes produce different results, but unlike the trend in KTH-TIPS2, there is no overall winner at the same patch size, while 6×6 arrives at 42.4% gives the best performance for linear kernel, 24×24 is the winner for $exp - \chi^2$ kernel with accuracy 43.2%; when comparing the results with those from the descriptors, although S3C model cannot beat the competitors at all different patch sizes, it gets the best numbers for both kernel.

Overall, we see S3C outperforms all the descriptors for different kernels on both datasets.

Feature	ClassificationRate(%)	
	Linear Kernel	$exp - \chi^2$ Kernel
Color	50.19	54.00
LBP	58.72	64.71
u-LBP	60.33	**67.15**
ri-LBP	50.34	52.29
ri,u-LBP	50.93	51.02
S3C(6×6)	63.79	57.46
S3C(12×12)	**71.25**	65.98
S3C(24×24)	55.85	60.67

(a) Results on KTH-TIPS2.

Feature	ClassificationRate(%)	
	Linear Kernel	$exp - \chi^2$ Kernel
Color	28.2	40.2
LBP	38.4	40.4
u-LBP	41.4	37.4
ri-LBP	37.8	37.8
ri,u-LBP	31.4	27.4
S3C(6×6)	**42.4**	35.8
S3C(12×12)	41.4	41.6
S3C(24×24)	35.2	**43.2**

(b) Results on FMD database.

FIGURE 4.8: Classification Rates on KTH-TIPS2 and FMD databases with different descriptors, including LBP, uniform-LBP (u-LBP), rotation invariant-LBP (ri-LBP), rotation invariant, uniform-LBP (ri,u-LBP), Multi-scale LBP (MLBP), single scale S3C feature at patch size {6,12,24} and two multi-scale S3C models, Joint-S3C (J-S3C) feature and Stacked-S3C feature (S-S3C).

4.4 Discussion

4.4.1 Visualization of Models

To provide further insight in our model we visualize the filters learnt by the S3C models that form the basis of the code. All filters for all databases are shown in Figure 4.10 (at patch size 6×6), 4.11 (at patch size 12×12), 4.12 (at patch size 24×24). We can see how color and structure is encoded jointly in those filters. While the filters for the FMD database seem to show significantly more structure, the KTH versions show higher frequency responses. This is consistent with the nature of the databases. The FMD shows objects made out of the particular material which includes stronger edges

from the object boundaries. In addition, the filters for the patch size 24 look a lot more noisy on both datasets which we attribute to a lack of data to fit this many parameters.

Furthermore, we are interested if we really achieved a better embedding for the data manifold formed by material patches. While a direct assessment is very difficult, we provide additional visualization here. Figure 4.9 show 2 dimensional projections by PCA of the representations by raw patches, LBP and our learned representation. We see that the MLBP seems already to do a much better job than the raw patches. Further improvements in terms of class separations can be seen for our learned features, which we find very encouraging for further studies of learned material descriptors

4.4.2 Patch Size

The results of single scale feature descriptor indicate that picking a proper patch size is important for making the model to work. The size of patch determines the locality of the descriptor and therefore affects how the descriptor can be generalized to more different instances, and from our experience, there seems not be any overall optimal patch size, which suggests we may need to try several candidates for a specific dataset and select the best one for use. On the other hand, this also reveals that automatically learned features can depict certain characteristics for dataset other than the traditionally manually designed descriptor with fixed pattern.

Furthermore, the impact of patch size also reminds us of the scale information. When fixing the number of model parameters and increasing the patch size, we are in fact trying to encode a larger neighborhood with fewer parameter which leads to a coarser representation in this sense and this is very similar to the behavior within the scale pyramids. These thoughts directly lead to our efforts to extend the model to multi-scale feature coding as will be described in the next chapter.

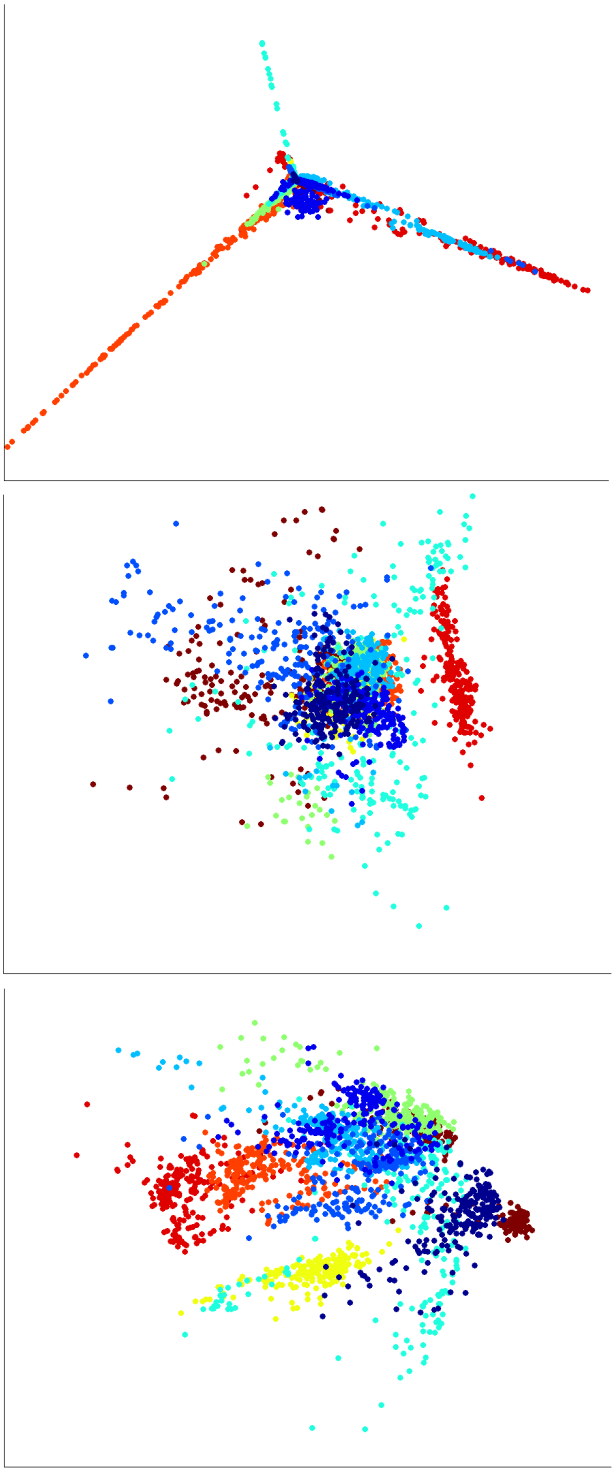


FIGURE 4.9: One example of visualization of low dimension projection via PCA for data in KTH-TIPS2a under different descriptors: Color-Patch (left), LBP(middle), single-scale S3C(right). Each dot represents the low dimension projection of the representation for one material image with each color denoting one different material category. This suggests that features learned by S3C represent the manifold structure within the data better.

FIGURE 4.10: Illustrations of learned filters at patch size = 6 on KTH-TIPS2 database (left) and FMD database (right)

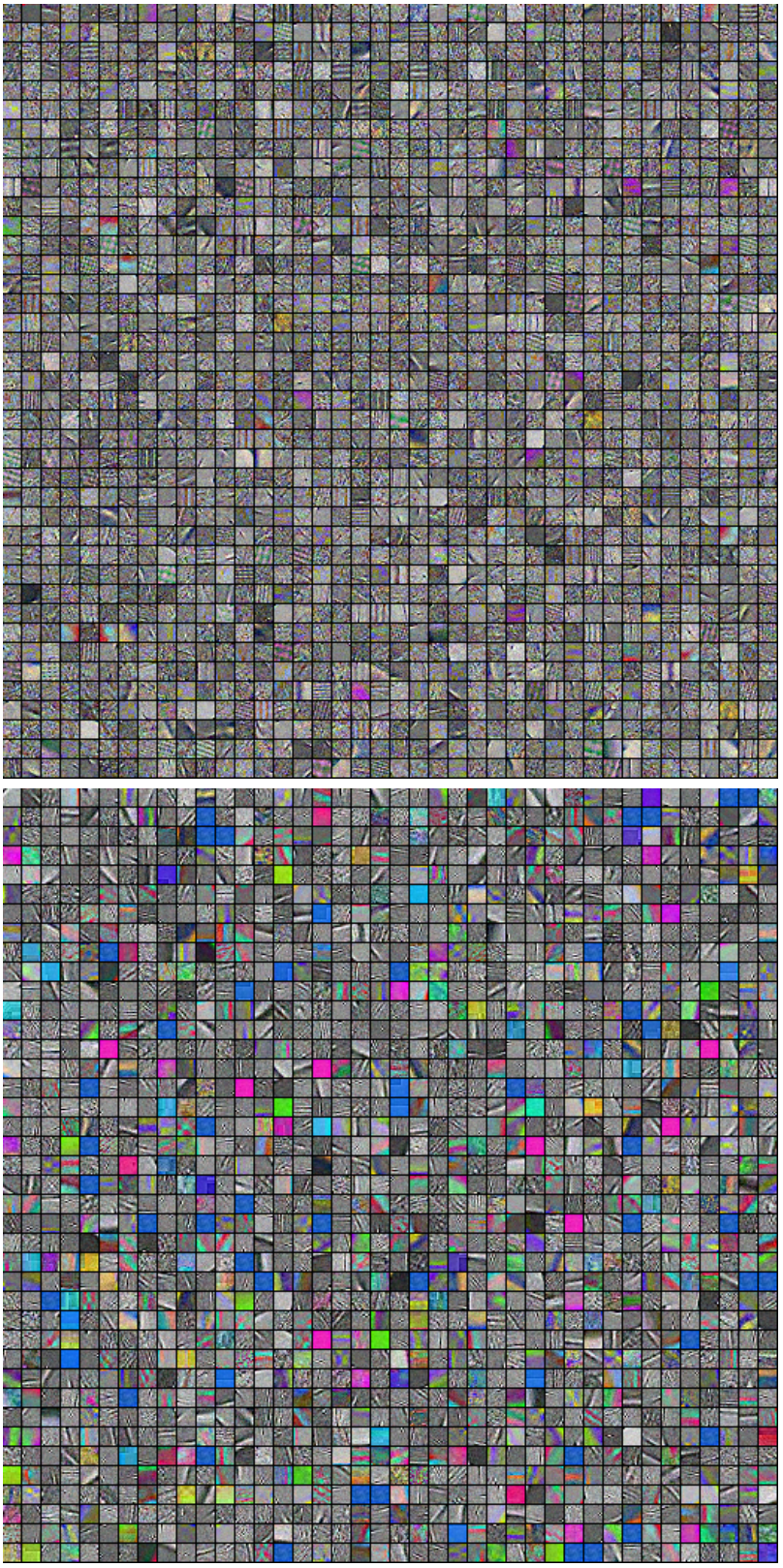


FIGURE 4.11: Illustrations of learned filters at patch size = 12 on KTH-TIPS2 database (left) and FMD database (right)

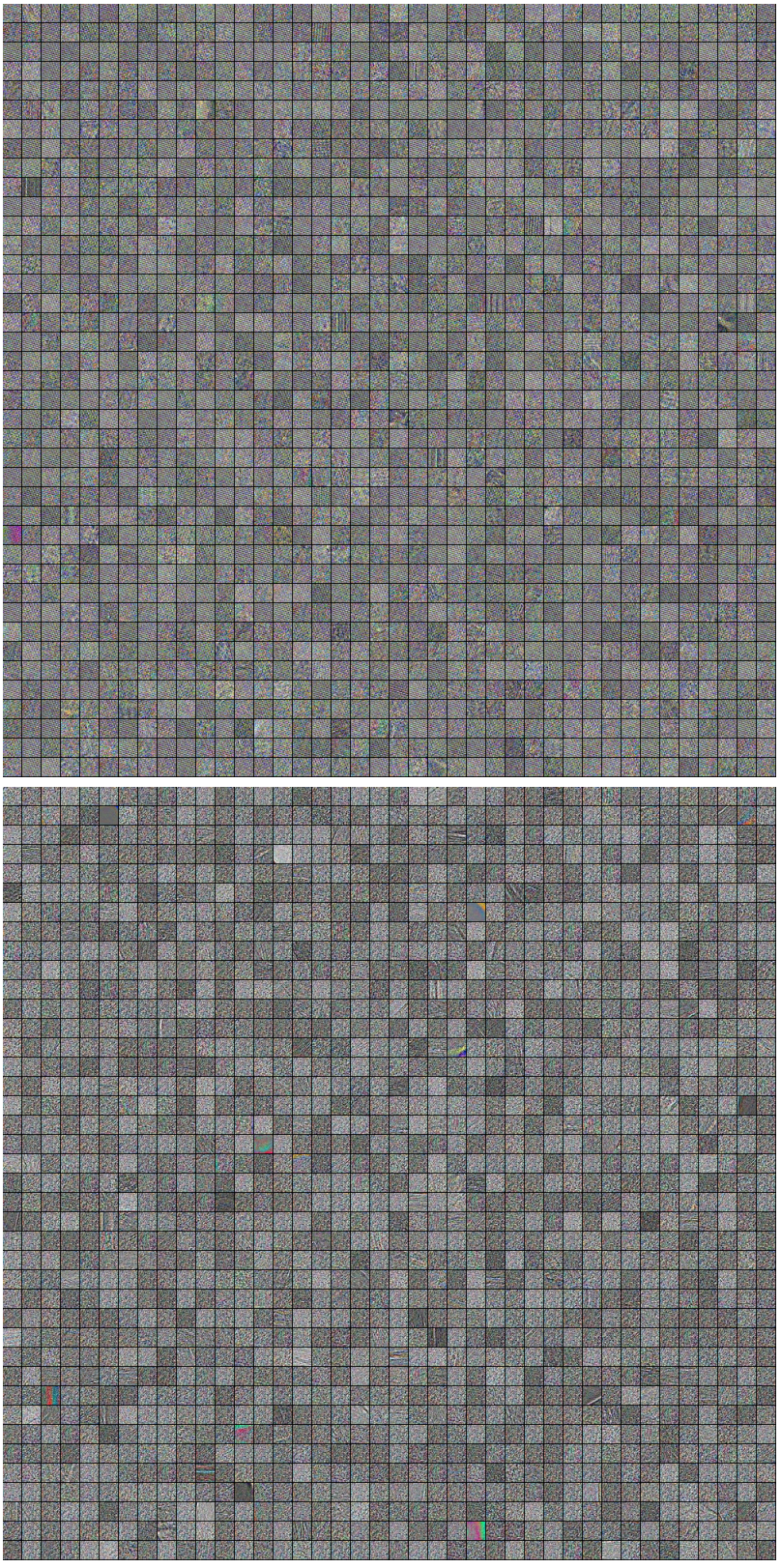


FIGURE 4.12: Illustrations of learned filters at patch size = 24 on KTH-TIPS2 database (left) and FMD database (right)

Chapter 5

Extension to Multi-scale Feature Coding

In previous chapter, we presented the content for single-scale feature learning, serving as starting point of this work. At the beginning of this chapter we shall motivate ourselves for the importance of scale information. Then based on the work in chapter 4, we make an effort to extend the learning framework for multi-scale and we propose two different strategies for the extension including the multi-scale stacked feature learning and multi-scale joint feature learning. Afterwards, we also present experimental results on multi-scale feature learning experiment and some conclusions.

5.1 Multi-scale Feature Learning

The scale information is a critical element for material and texture recognition problem. [7] showed that explicit treatment of scale is necessary for material recognition in realistic settings. In [28], Li et al performed a manifold alignment with respect to scale between real and synthesized data, which turned to be crucial for using the generated data to improve recognition rate. Similarly, local descriptor like LBP is usually limited by its small spatial support area, several extensions [33], [31] for multi-scale descriptor have also been shown to overcome the limitation to some extent. Therefore we propose different strategies to employ multi-scale information in feature learning (also as depicted on Figure 5.1):

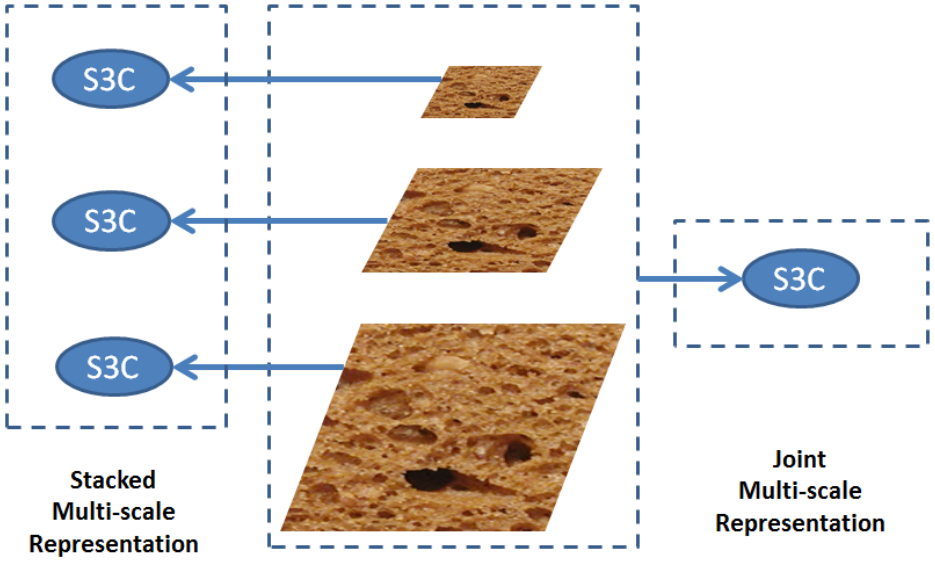


FIGURE 5.1: Multi-scale feature learning with: multi-scale collaborated representation (left), multi-scale joint representation (right).

5.1.1 Multi-scale Stacked Feature Learning (S-S3C)

In the first strategy, we perform the encoding at multiple scales and stack the obtained codes as a new piece of code, then use this code for classification. In practice, we alter the patch size during encoding for each scale to realize features for multiple spatial resolutions in order to represent the scale information.

5.1.2 Multi-scale Joint Feature Learning (J-S3C)

In the second strategy, we first construct a multi-scale pyramid for each image, apply the feature learning directly on the pyramid and then use the obtained codes for classification. In particular, for color images, we first convert them into gray-scale representation, and then stack its Gaussian pyramid for feature learning. Simultaneously, we perform an additional encoding process on the original image to account for the color information, and together with codes from the gray-scale pyramid representation, we get a complete multi-scale joint representation.

5.2 Experiments

5.2.1 Implementation Details

MLBP MLBP bases on single scale LBP descriptors, we can vary the building block to produce a variety of MLBP. For example, 4-scale MLBP $LBP^{riu2}_{8,1+8,2.4+16,4.2+16,6.2}$ consists of four single-scale LBP descriptors $LBP^{riu2}_{8,1}$, $LBP^{riu2}_{8,2.4}$, $LBP^{riu2}_{16,4.2}$ and $LBP^{riu2}_{16,6.2}$. In practice, we simultaneously compute the LBP features centered at the position but with different R, N and then concatenate these codes into a new feature vector. Similar to single-scale LBP, we define the mappings for each single-scale LBP before the loop to speed up the computation.

S-S3C To implement the S-S3C, we perform single-scale training at 3 scales which correspond to 3 different patch sizes at $6 \times 6, 12 \times 12, 24 \times 24$ and then code them into 3 pieces of codes x_1, x_2, x_3separately. Afterwards, we concatenate them into a new code $x_1 + x_2 + x_3$. In practice, we also tried different normalization schemes for the combination, include individual normalization $\hat{x_1} + \hat{x_2} + \hat{x_3}$, overall normalization $x_1 + \hat{x_2} + x_3$ and double normalization $x_1 + \widehat{\hat{x_2} + x_3}$ where $\hat{}$ denotes the operation of normalization, and we found out the overall normalization gives the best performance.

J-S3C We use the method described in previous section to obtain the code. Compare with the S-S3C implementation, this is much faster as we perform single-scale S3C coding on the concatenated image with only extra computation when creating the multi-scale pyramid.

5.2.2 Experimental Setup

For this group of experiment, we introduce scale information with two different strategies, stacked multi-scale representation and joint multi-scale representation, as described previous section. In particular, we separate the gray-scale multi-scale representation and color version.

5.2.3 Experimental Results

The final results for multi-scale experiment are shown in Table 5.2. In particular, we are interested in how the learned multi-scale representation works compared with MLP as the hand-crafted multi-scale descriptor.

From the results on KTH-TIPS2: first, we see the color scale J-S3C coding works better than the gray scale version for both linear and $exp-\chi^2$ kernels; second, both versions of J-S3C coding do a better job than the S-S3C coding scheme; when compared with the numbers from MLBP, while the S-S3C is worse than MLBP, both S-S3C representations consistently outperform it with accuracies 70.47% and 69.30%; further, we investigate the differences between single-scale S3C (shown in Table 4.8) and the multi-scale extension, we note that both versions of multi-scale representations are superior to single-scale LBP and color-patch, however, we do get surprisingly good result from single scale S3C feature that is even slightly better than any of the multi-scale descriptors.

From the results on FMD database: the two versions of multi-scale representation perform differently with different kernels, the gray-scale J-S3C model gives better result with linear kernel whereas the color-scale J-S3C model leads to better performance with $exp-\chi^2$ kernel; the J-S3C model consistently outperforms S-S3C model except for the case where gray scale J-S3C model is slightly worse than the S-S3C using the linear kernel; when compared with MLBP descriptor, the learned feature is the overall winner on this dataset; further, we see the multi-scale feature learning works better than all the single-scale representation for both learned feature and the descriptors.

Feature	ClassificationRate(%)	
	Linear Kernel	$exp-\chi^2$ Kernel
MLBP	66.71	66.08
S-S3C	63.65	58.63
J-S3C-gray	68.08	66.57
J-S3C-color	**70.47**	**69.30**

(a) Results on KTH-TIPS2.

Feature	ClassificationRate(%)	
	Linear Kernel	$exp-\chi^2$ Kernel
MLBP	41.4	42.0
S-S3C	49.2	42.2
J-S3C-gray	**50.0**	41
J-S3C-color	48.8	**43.2**

(b) Results on FMD database.

FIGURE 5.2: Classification Rates on KTH-TIPS2 and FMD databases with different descriptors, including LBP, uniform-LBP (u-LBP), rotation invariant-LBP (ri-LBP), rotation invariant, uniform-LBP (ri,u-LBP), Multi-scale LBP (MLBP), single scale S3C feature at patch size {6,12,24} and two multi-scale S3C models, Joint-S3C (J-S3C) feature and Stacked-S3C feature (S-S3C).

5.3 Discussion

5.3.1 Scale Information

Most of time, we see improvements when incorporating scale information, however on the KTH-TIPS2 database, we find that a descriptor learned at single scale performs the best. This may be related to the properties of the specific dataset and also the nature of our designed multi-scale descriptor. Both strategies for our multi-scale descriptors involve some redundancy between every scales that may degrades the classification performance, in return, this redundancy also encodes the scale information by itself that could improves performance, and final performance will be affected by these two factors jointly:

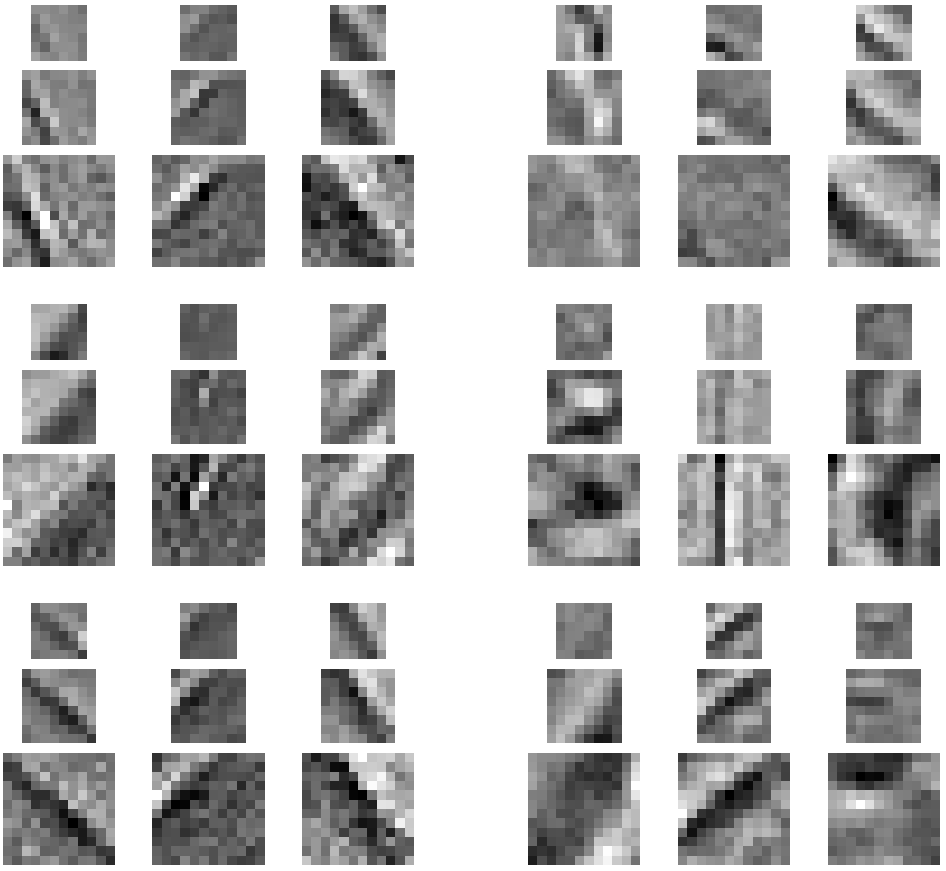
For the KTH-TIPS2 databases, material images were taken under strictly controlled conditions, in particular, only 9 different scales for all the instances, so the improvement via scale information is very limited in this case while the redundancy still affect the classification rate negatively, this in particular explains why the two multi-scale descriptor which already incorporate the information in model with patch size of 12 get worse results than the single-scale descriptor.

In contrast, for the case of FMD database, images were collected from Flickr photos in arbitrary conditions, scale information become significantly more important and surpass the influence from the redundancy, which makes the multi-scale descriptor beat any of its components at single scale. In real world application, it is always closer to the latter situation, thus the multi-scale descriptor is preferable in this sense.

Figure 5.3 shows the visualization of our proposed Multi-Scale Spike-and-Slab Sparse Coding model. We see how each filter has a multi-scale response. We looked at a larger range of such filters, which reveals some more interesting properties. Some of these filters have a very similar structure across scales, while other do vary strongly. This observation and the strong performance numbers in our experiments lead us to believe that a multi-scale code indeed captures additional information about how edge structures propagate through scale.

5.3.2 Color Information

While color information serves as an important cue for visual recognition task, it could also leads to confusion, so we should be careful to incorporate color information. It is interesting to compare the results for the two multi-scale joint representation, with one in gray-scale and the other in color: on the KTH-TIPS2 database, the color information



(a) Multi-scale filters learned on KTH-TIPS2 database.

(b) Multi-scale filters learned on FMD database.

FIGURE 5.3: Example of multi-scale filters learnt on the KTH-TIPS2a (left) and FMD (right) datasets. The represent a multi-scale code that is learnt jointly based on the proposed J-S3C model

led to an improvement over the gray-scale representation while the gray-scale version achieved the best performance on FMD database. It could be explained by the large variation of color information in FMD which naturally caused the confusion, whereas color cue is more simple and informative for classification on KTH-TIPS2.

5.3.3 Further Comparison to State-of-the-Art Descriptors

Together with the results from single scale, we also made comparison to State-of-the-Art descriptors to see the real power of our model. As not all papers follow the same experimental protocol, we reproduced two additional settings in order to provide more points of comparison. We follow the protocol in [23] and take 3 samples of each class for training and the fourth for testing, and then report averages over 4 random partitions via a simple 3-NN classifier, feature learned by single scale S3C at patch size of 12x12 achieved 70.2%, which is significantly better than the reported results of 64.2% for

LQP. Further we did additional experiments on FMD, following the settings in [22], i.e. performing 5 trials and computing the average, and with multi-scale collaborated representation, we got average recognition rate of 48.3% and standard deviation of 1.8%, which is comparable to the best single kernel descriptor with 49%.

Chapter 6

Representation Transfer

The S3C model has been proposed for the transfer learning challenge for objects recognition task, yet we adapt it as a pure feature learning technique for material recognition task. One might be also interested in how does the transfer learning work for material recognition, namely if the model learned on a particular material dataset can be used for some different datasets. In this chapter we shall cover this part of content.

6.1 Representation Transfer

One inspiring idea is the self-taught learning proposed by [37] by using unlabeled data in supervised classification tasks. In particular, one should distinguish it from other common learning paradigms, like semi-supervised learning and transfer learning. In semi-supervised learning, classifier uses additional unlabeled data with same set of labels; in transfer learning, classifier uses additional labeled data with possibly different set of labels; in self-taught learning, it only requires additional unlabeled images with almost random labels. The basic rational behind self-taught learning is that image data shares some low-level information like edges, and by learning a representation on an easy-to-get data like Internet images, it can still be applied to arbitrary image classification tasks. For material recognition task, we would like also to explore similar idea, and so we did additional experiments to investigate transferring presentations across databases.

Feature	ClassificationRate(%)	
	Linear Kernel	$exp - \chi^2$ Kernel
Code learned on FMD, and represent KTH-TIPS2		
S3C(12×12)	65.98	61.11
Code learned on KTH-TIPS2, and represent FMD		
S3C(12×12)	44.4	43.2

TABLE 6.1: Results for Transfer Representation.

6.2 Experiments

6.2.1 Experimental Setup

For this purpose, we conducted two experiments. From one direction, we first train a S3C model on FMD database, and use the model to encode the data in KTH-TIPS2 database, afterwards, we use the obtained representation to perform classification; for the other direction, we follow the similar fashion but train the model on KTH-TIPS2 this time, and then use coded FMD data for classification. In particular, we fix the patch size at 12 and only use single scale model.

6.2.2 Experimental Results

The results are shown in Table 6.1. In order to compare the transfer representation with the standard representation where codes are learned on the same dataset, we present again the numbers in Table 6.2 for both datasets using single S3C model at patch size of 12×12.

From these results: we can see that transfer representation works better with linear kernel on both datasets; when encoding the image data in KTH-TIPS2 with the model learned on FMD, the performance is worse than the standard representation; when representing the FMD data with the model learned on KTH, the performance even improved any of the single scale descriptor. This indicates that the features learned through the S3C model on specific dataset are actually eligible to capture some common characteristics which could generalize to more different data within similar context (various material categories in our case).

Feature	ClassificationRate(%)	
	Linear Kernel	$exp - \chi^2$ Kernel
Code learned on KTH-TIPS2, represent KTH-TIPS2		
S3C(12×12)	71.25	65.98
Code learned on FMD, and represent FMD		
S3C(12×12)	41.4	41.6

TABLE 6.2: Results for Standard Representation.

6.3 Discussion

The results on representation transfer is encouraging and validate it is possible to learn a model on one dataset and apply the model to get a reasonable representation.

Yet there are still questions like how to explain the different influences brought by the models trained on different datasets. One may expect the models trained on FMD database can give improvement other than slightly degrading on the recognition rate as the material images in it contain more variations and hence feature learned on it is more general in this sense. In contrast, images in the KTH-TIPS2 database are all photoed under very restricted conditions and there are only a few instances per category. All these factors seem to suggest the model trained on it may not work so well on the more complex FMD images, yet we see improvement in this case. Though the final answer may need further deeper analysis, here we may provide one possible explanation. Indeed, the variations in FMD can be good for the model to generalize on new data, yet the model also incorporates a lot of information on shape and environment which is unnecessary for description on KTH-TIPS2 which only capture the surface looks on material sample, thus can decrease the power of the model when applied to KTH-TIPS2. Nevertheless, the model trained on KTH-TIPS2 only encode very descriptive information on the material surface which can be generalized to real world scene.

Chapter 7

Conclusion and Future Work

In this thesis, we have presented a general framework for feature learning on material recognition task. Based on the previous works, we successfully applied novel unsupervised feature learning technique to material images and made further extension to it by incorporating scale information. Our experimental results indicate this is a promising approach to material recognition task.

7.1 Discussion of Contributions

From the early CUReT texture database, to later proposed KTH-TIPS and KTH-TIPS2 databases, till recent Flickr Material Database, the evolution of texture and material datasets indicates researchers are getting closer to face real-world challenge. Hence we spare no effort to push ourselves towards a better recognition solution. As feature representation is crucial to general recognition task, we considered unsupervised learning techniques over manually designed descriptors to obtain features in a more automatic and adaptive way. Our contributions can be summarized as follows:

Unsupervised Feature Learning for Material Recognition We have applied a recently proposed generative model S3C model to feature learning for material recognition task and compared the performance with hand-crafted feature descriptors. Our results shows consistent improvement over the descriptors. We also made visualization of the obtained models to provide further insight and managed to show the learned model is capable to achieve a better embedding for the data manifold formed by material patches. Moreover, the observation on how the patch size influence the recognition rate gave rise to our further interest in multi-scale extension to the model.

Multi-scale Feature Learning To explore how we can encode scale information in our framework, we proposed two different strategies, e.g. the multi-scale stacked feature learning and multi-scale joint feature learning. The results are encouraging, we compared the results with state-of-the-art descriptors and showed improved performance on standard material recognition task. In addition, we analyzed the obtained model and investigated how multi-scale code can capture information through scales.

Transfer Representation We discussed if and how the learned model can be applied to a new database and showed the feasibility of this attempt. This indicates the feature learned via the models is capable to capture certain characteristic structures shared between different datasets.

Summary We believe this thesis contributes to both feature learning and general material recognition. Starting from the basic model, we carefully adapted to our specific task and developed extensions tailed to our needs. For each group of experiments, we not only made analysis between the numbers but also tried to visualize the model and provide insight to explain the results.

7.2 Future Work

While this thesis has shown the feasibility of feature learning on material recognition task, we identify three main directions that seem promising to be explored:

Inference Mechanism To enable parallel updates during inference, the author for S3C model used heuristic approximation which no longer guaranteed convergence though it usually works well in practice. In consequence, the stopping criterion for the learning process is not very clear at the moment. One can further investigate if it is possible to maintain parallel updates while preserving the property of convergence.

In addition, our experimental results especially the plots for filters indicates it's getting more difficult for the model to converge with increasing patch size within current number of iterations. This may be partly explained by the fact that inference is growing with number of visible units as larger patch consists of more pixels. To further address this problem, one may carefully distribute the work among several GPUs to greatly speedup the process so that the inference reach convergence in shorter time.

Model Extension In this thesis, we made one extension to the S3C model by introduce multi-scale feature learning. Another interesting exploration is to stack the S3C module into hierarchical model. Similar structure has been proposed by [46] for sparse coding and has been shown to produce strong performance. One can further consider using deep learning framework which gained large popularity in machine learning and computer vision communities recently due to its strong performances in strong performance on several image classification benchmarks [21] [26].

Relationship between Different Features One can further investigate the deeper relationships between learned features and hand-crafted features. Previous attempt has been made in [4] where it is possible to establish a kernel view of orientation histogram. It would be interesting to see how to connect our learned feature to existing descriptors and therefore form a unified view on them.

Bibliography

[1] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

[2] C.M. Bishop et al. *Pattern recognition and machine learning*. springer New York, 2006.

[3] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[4] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *NIPS*, 2010.

[5] Y.L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[6] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[7] Barbara Caputo, Eric Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *ICCV*, 2005.

[8] A. Coates and A.Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, 2011.

[9] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 1995.

[11] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, 2004.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[13] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 1999.

[14] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[15] Charles Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, 2003.

[16] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer Series in Statistics, 2001.

[17] Ian Goodfellow, Aaron Couville, and Yoshua Bengio. Large-scale feature learning with spike-and-slab sparse coding. In *ICML*, 2012.

[18] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh. On the significance of real-world conditions for material classification. In *ECCV*, 2004.

[19] Marko Heikkila and B. Timo Ahonen. Matlab code for lbp. http://www.cse.oulu.fi/CMV/Downloads/LBPMatlab, 2009.

[20] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002.

[21] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[22] D. Hu, L. Bo, and X. Ren. Toward robust material recognition for everyday objects. In *BMVC*, 2011.

[23] Sibt Ul Hussain and B. Triggs. Visual recognition using local quantized patterns. In *ECCV*, 2012.

[24] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[25] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[26] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[27] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *ICCV*, 1999.

[28] Wenbin Li and Mario Fritz. Recognizing materials from virtual examples. In *ECCV*, 2012.

[29] Ce Liu, Lavanya Sharan, Edward H. Adelson, and Ruth Rosenholtz. Exploring features in a bayesian framework for material recognition. In *CVPR*, 2010.

[30] D.G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[31] T. Mäenpää and M. Pietikäinen. Multi-scale binary patterns for texture analysis. *Image Analysis*, 2003.

[32] K. Nigam, A.K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 2000.

[33] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *TPAMI*, 2002.

[34] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 1996.

[35] B.A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 1996.

[36] X. Qi, R. Xiao, J. Guo, and L. Zhang. Pairwise rotation invariant co-occurrence local binary pattern. In *ECCV*, 2012.

[37] R. Raina, A. Battle, H. Lee, B. Packer, and A.Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, 2007.

[38] L.K. Saul and M.I. Jordan. Exploiting tractable substructures in intractable networks. In *NIPS*, 1996.

[39] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[40] M. Topi, O. Timo, P. Matti, and S. Maricor. Robust texture classification by subsets of local binary patterns. In *Pattern Recognition*, 2000.

[41] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *CVPR*, 2003.

[42] M. Varma and A. Zisserman. A statistical approach to material classification using image patch exemplars. *TPAMI*, 2009.

[43] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[44] S. Vempati, A. Vedaldi, A. Zisserman, and CV Jawahar. Generalized rbf feature maps for efficient detection. In *BMVC*, 2010.

[45] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[46] K. Yu, Y. Lin, and J. Lafferty. Learning image representations from the pixel level via hierarchical sparse coding. In *CVPR*, 2011.