Hierarchical feature learning for image categorization

Zuo, Zhen

2015

Zuo, Z. (2015). Hierarchical feature learning for image categorization. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65659>

<https://doi.org/10.32657/10356/65659>

HIERARCHICAL FEATURE LEARNING FOR IMAGE CATEGORIZATION

ZHEN ZUO

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

2015

Acknowledgments

First and foremost, I would like to express my sincerest gratitude to my supervisor Professor Gang Wang, who has supported me throughout my Ph.D. study with his patience and knowledge, and encouraged me to explore my own way of research. During these years of studying under his supervision, I have learned a lot. I have been greatly impressed by his professional attitude towards high quality research.

I would like to thank all the members of our team for being great colleagues to work with, and in particular thanks to Mr. Bing Shuai, Mrs. Abrar Hamzeh Saad Abdul Nabi, Mr. Xingxing Wang, Dr. Jiwen Lu, Mr. Ting Liu, Mr. Bing Wang, Dr. Li Wang, Dr. Zhenhua Wang, Dr. Jinghua Wang, Mr. Amir Shahroudy, and Mr. Rahul Rama Varuir.

I would also thank all my friends for their generous help and suggestions: Ms. Yan Wang, Dr. Yifan Li, Mr. Lifan Zhao, Mr. Renliang Weng, Ms. Keyan Liu, Dr. Xiang Li, Dr. Yusong Hu, Mr. Yi Yu, Mr. Jie Chen, Mr. Peng Hu, Dr. Qi Chen, Mr. Dajiang Zhang, Ms. Xue Wang, Mr. Jingpeng Wang, Mr. Chengde Wan, Mr. Yangtao Wang, Mr. Yuan Wang, Ms. Anran Wang.

Last but not least, I would like to thank my parents and Mr. Xiao Liu for their constant love and encouragement.

Contents

Acknowledgments	i
Summary	v
List of Figures	vii
List of Tables	x
List of Abbreviations	xi
List of Notations	xii
1 Introduction	1
1.1 Background	1
1.2 Motivations	4
1.3 Contributions of the Thesis	6
1.4 Organization of the Thesis	8
2 Related Works	10
2.1 Feature Engineering & Feature Learning	10

2.1.1	Feature Engineering	11
2.1.2	Feature Learning	12
2.2	Image Categorization	19
3	Discriminative Hierarchical Feature Learning	24
3.1	Introduction	25
3.2	Naive Bayes Nearest Neighbour	26
3.3	Discriminative Hierarchical Feature Learning	28
3.3.1	Single Layer Discriminative Feature Learning	28
3.3.2	Hierarchical Learning Structure	31
3.3.3	Approximation	32
3.4	Experiments and Analysis	32
3.4.1	Results	33
3.4.2	Parametric Analysis	39
3.5	Conclusion	41
4	Deep Discriminative and Shareable Feature Learning	43
4.1	Introduction	44
4.2	Discriminative Training	46
4.3	Deep Discriminative and Shareable Feature Learning	49
4.3.1	DDSFL Hierarchical Structure	49
4.3.2	Single Layer DDSFL Learning Components	50
4.3.3	DDSFL Objective Function and Optimization	55
4.4	Exemplar Selection	57

4.4.1	NN Based Exemplar Selection	57
4.4.2	SVM Based Exemplar Selection	60
4.4.3	NN Based vs SVM Based Exemplar Selection	62
4.5	Experiments and Analysis	64
4.5.1	Datasets and Experiment Settings	64
4.5.2	Comparison with Other Features	67
4.5.3	Comparison with Other Methods on Scene Classification	70
4.5.4	Analysis of the Effects of Different Components	72
4.6	Conclusion	76
5	Convolutional and Hierarchical Recurrent Neural Network	78
5.1	Introduction	79
5.2	Recurrent Neural Networks on Vision	82
5.3	Convolutional and Hierarchical Recurrent Neural Networks	84
5.3.1	Convolutional Layers	84
5.3.2	Review of General RNNs	85
5.3.3	Hierarchical Recurrent Layers	88
5.3.4	Fully Connected Layers	94
5.4	Experiments	95
5.4.1	Experimental Settings	96
5.4.2	Experimental Results	98
5.4.3	Analysis of C-HRNNs	105
5.5	Conclusions	113

6	Conclusions & Future Works	114
6.1	Conclusions	114
6.2	Future Works	115
	Author's Publications	117
	Bibliography	119

Summary

Extracting informative, robust, and compact data representation (feature) has been considered as one of the key factors for good performance in computer vision, and image categorization is one of the most fundamental computer vision problems. Traditionally, hand-crafted features like SIFT and HOG have been widely used, however, these features cannot adapt to data and need to be well designed. In contrast, feature learning methods have been proposed to encode data-adaptive information in data representation, which have outperformed hand-crafted features with big gaps, and brought a rapid progress in image categorization. The goal of this thesis is to present various feature learning architectures for the problem of object/scene image categorization.

In the first part of the thesis, a discriminative hierarchical feature learning framework will be presented for object image categorization. This work aims to learn non-linear transformation matrices to transform image patches to local features. Current features learned by unsupervised learning methods can hardly capture the differences between different classes, which are crucial for object categorization. To capture such information, a discriminative constraint is proposed to force the local feature patches extracted from the same categories to be locally similar, while local feature patches from different classes to be separable.

In the second part, discriminative and shareable information will be encoded in

features for scene image categorization. Different from object images, scene images do not have clear foreground/background. Some patterns are shared among several classes, some patterns are class-specific. While some patterns represent noisy data, which is not helpful and should be excluded. In order to encode such information, the exemplar based deep discriminative and shareable feature learning framework will be proposed to learn compact filter banks, and hierarchically transfer local image patches to features.

In the third part, a class of end-to-end neural networks, called convolutional and hierarchical recurrent neural networks (C-HRNNs), will be presented for large-scale object/scene image categorization. In existing convolutional neural networks (CNNs), both convolution and pooling are locally performed for image regions separately, no contextual dependencies between different image regions have been taken into consideration. Such dependencies represent useful spatial structure information in images. In contrast, recurrent neural networks (RNNs) are well known for their capability of learning contextual dependencies of sequential data by using the recurrent (feed-back) connections. In this work, C-HRNNs aim to encode both spatial and scale dependencies among different image regions to enhance the global discriminative power of image representation. Where CNN layers are firstly processed to generate middle level features. HRNN layers are then processed to learn spatial and scale dependencies.

List of Figures

1.1	Challenge in image categorization: large intra-class variance, and small inter-class variance.	2
1.2	Image categorization framework.	3
2.1	The basic single layer feature learning module structure of RICA. .	15
2.2	Convolutional neural network structure.	17
2.3	Structure of RNN.	20
2.4	The “BoW + SPM” image categorization framework.	20
3.1	Schematic diagram of our discriminative information encoding scheme.	26
3.2	Illustration of our discriminative feature learning and object recognition framework.	27
3.3	Examples from Caltech-101 dataset.	34
3.4	(a) Examples from STL-10 dataset. (b) Examples from four different tree species of NTU Tree-51 dataset.	35
3.5	Per-category recognition accuracy on STL-10 for comparison between our hierarchical feature learning method and the original hierarchical RICA.	36

3.6	False positive image examples in STL-10, hierarchical feature learning failed cases.	37
3.7	The NTU tree-51 dataset.	38
3.8	Per-category recognition accuracy on NTU tree-51 for comparison between our hierarchical feature learning method and the original hierarchical RICA.	40
3.9	Accuracy of our hierarchical features on Caltech-101 versus the weight of the discriminative term.	41
4.1	Illustration of single layer discriminative and shareable feature learning scheme.	47
4.2	The hierarchical structure of DDSFL.	47
4.3	DDSFL image classification training and testing framework.	48
4.4	Comparison results of DDSFL vs Caffe on MIT Indoor.	69
4.5	Visualization of the filters learned by RICA and DDSFL on the UIUC Sports.	72
4.6	Results of varying number of DDSFL filters and layers on Scene 15.	76
5.1	The overall framework of C-HRNNs. It's combined with three types of layers: CNN layer, hierarchical RNN layer, and FC layer.	80
5.2	General SRN and LSTM structures.	87
5.3	Spatial contextual dependencies modeling in a single scale HRNN layer.	88
5.4	Visualization of the hrnn6 (C-HLSTM)/conv5 (SPP-net) layer features.	106

5.5	Same as Figure 5.4 for more sample images.	107
5.6	Selected per class accuracies tested on Places 205: SPP-net vs C- HLSTM.	110
5.7	C-HLSTM vs SPP-net, success and failed cases.	112

List of Tables

3.1	Results on Caltech-101.	39
3.2	Results on STL-10.	39
3.3	Results on NTU tree-51.	39
4.1	Comparison results between DDSFL feature and other features. . .	66
4.2	Class-wise classification comparison results between DDSFL and other features on PASCAL VOC 2012.	67
4.3	Comparison results of DDSFL and other popular methods on Scene 15, UIUC sports, and MIT Indoor.	71
4.4	Analysis of the effect of each component in DDSFL (two layers). . .	74
4.5	Analysis of the effect of number of layers.	75
5.1	Network structures. C-HSRN and C-HLSTM share the same struc- ture, which are indicated as C-HRNNs.	97
5.2	Comparison of error rates on the ILSVRC 2012 validation set. . . .	99
5.3	Comparison of accuracy on Places 205.	101
5.4	Comparison of accuracy on SUN 397.	102
5.5	Comparison of accuracy on MIT indoor.	104
5.6	Error rates of applying C-HRNNs with different spatial contextual directions on ILSVRC 2012.	109

List of Abbreviations

SIFT	Scale-invariant feature transform
HOG	Histogram of oriented gradients
LBP	Local binary patterns
SVM	Support vector machine
BoW	Bag of words
SPM	Spatial pyramid matching
PCA	Principal component analysis
RICA	Reconstruction independent component analysis
L-BFGS	Limited memory Broyden-Fletcher-Goldfarb-Shanno
NBNN	Naive Bayes nearest neighbour
P2CD	Patch to class distance
LLC	Locality constrained linear coding
DDSFL	Deep discriminative and shareable feature learning
CNN	Convolutional neural network
RNN	Recurrent neural network
GPU	Graphics processor unit
DBN	Deep belief network
RBM	Restricted boltzmann machine
C-HRNN	Convolutional and hierarchical recurrent neural network
SRN	Simple recurrent neural network
LSTM	Long short term memory recurrent network
BPTT	Back propagation through time

List of Notations

$P(a b)$	The conditional probability of a given b
$x \in \mathcal{R}^D$	A D -dimensional vector x with real elements
$X \in \mathcal{R}^{D_1 \times D_2}$	A matrix X with size of $D_1 \times D_2$
$NN(x)$	The nearest neighbour of x
$\ x\ _0$	l_0 -norm of x which counts the number of nonzero elements in x
$\ x\ _1$	l_1 -norm of x defined as $\sum_d^D x_d $
$\ x\ _2$	l_2 -norm of x defined as $\sqrt{\sum_d^D x_d^2}$
$\ x - y\ ^2$	Euclidian distance between vectors x and y
$\min f(\cdot)$	Minimizing the loss function $f(\cdot)$
$\exp(\cdot)$	Exponential function
$\partial x / \partial y$	The partial derivative of x with respect to y
$Tr(X)$	Trace, sum of the elements on the main diagonal of X
$\text{diag}(x)$	Diagonal matrix corresponds to vector x
$\text{abs}(\cdot)$	Absolute value
$(\cdot)^T$	Matrix or vector transpose
$\odot$	Elementary multiplication

Chapter 1

Introduction

1.1 Background

“A picture is worth a thousand words”. Images are able to convey complex information. Human visual system is amazingly powerful. We are able to recognize objects/scenes images in various conditions, such as multi-viewpoint, different illumination conditions, various scales, affine transformation, rotation, occlusion, etc. However, we interact with numerous images everyday, analyzing images by humans are very labor intensive and low efficient. Therefore, utilizing computers to automatically and efficiently do large scale visual understanding becomes more and more important. To understand an image, the very first step is recognizing classes of the objects/scenes. This procedure is called image categorization, and it is one of the most fundamental and widely demanded topics in computer vision area.

Nowadays, image categorization research is being performed on large-scale natural image datasets. In these datasets, images with the same object/scene labels can be very variant (e.g. in Figure 1.1, the appearance of different cats/dogs are

Figure 1.1: Challenge in image categorization: large intra-class variance, and small inter-class variance. There are two categories: cats are indicated with red boxes, and dogs are indicated with green boxes. (Best viewed in color)

very diverse), while images from different semantic categories can be highly similar (e.g. Figure 1.1, the appearance of cat and dog categories are highly similar). In image data domain, each image can have an infinite number of pixel combinations, the information is usually redundant and highly variant. For human brains, such large intra-class variance and small inter-class variance problem can be easily solved. However, it is extremely hard for computers to extract the foreground signal from the background noise, and distinguish one class of objects from another. As shown in Figure 1.2, in general image categorization framework, the input is the raw pixel image, the target output is the corresponding image label. To get the image labels, feature vectors will be extracted to represent images, based on which, classifiers will be trained on training image features, and used to classify testing image features. There are mainly two challenges:

- 1 How to efficiently and effectively extract useful information from images, and

Figure 1.2: Image categorization framework.

represent it as compact features.

- 2 How to train appropriate classifiers to discriminatively classify features from different image categories, and give semantic meaningful labels that humans can understand.

This thesis mainly focus on solving the first challenge. Feature engineering/learning is one of the most fundamental problems in a wide range of computer vision topics, such as object recognition and scene classification. High quality features should be able to reveal the most significant information in the input raw data, while also be mathematically and computationally convenient to process. There are mainly two lines of works to solve this problem: 1) feature engineering, and 2) feature learning.

Feature engineering (hand-crafted features) had ruled the computer vision society in the previous decade. Famous works include SIFT [1], HOG [2], Shape context [3], Gabor features [4], etc. Feature engineering takes advantage of human ingenuity, and utilizes expert knowledge to extract discriminative information. Although such feature descriptors can lead to good performance, they have several obvious drawbacks: the feature extraction procedures need to be well designed step by step, which are labor intensive; the parameters are fixed, they cannot adapt to different input training data; the feature extraction procedure is usually com-

plicated and time consuming, which makes hand-crafted feature hard to perform very well on challenging real-world image categorization tasks.

In contrast, feature learning has shown great advantages in automatically and efficiently discovering useful information based on the provided training data. Especially in recently few years, deep learning techniques have achieved great success on many challenging computer vision tasks. For example, the deep convolutional neural networks [5] won the ImageNet 2012 image classification challenge, and outperformed the rank 2 team (which used hand-crafted features) with 10% in error rate. This thesis focuses on designing feature learning algorithms for image categorization.

The second challenge is training appropriate classifiers. In image categorization, supervised training datasets are usually provided with image level class labels. Based on such training data, classifiers are trained and stored for future testing data label prediction. The most popular classifiers are the support vector machine (SVM) [6] and Softmax classifier [5]. Classifier is an important component in image categorization, but it is not our primary focus. In the rest of this thesis, SVM or Softmax will be directly employed.

1.2 Motivations

The goal of this thesis is developing hierarchical feature learning algorithms for generating effective and efficient image representations, which are applied to image categorization tasks.

For image categorization, the intra-class variance is usually huge while the inter-class variance is relatively small. Sometimes, images from totally different classes can share the same components. For example, both “street” and “high-

way” have large portion of areas corresponding to “road”, “car” and “sky”. But recognizing “road”, “car” and “sky” cannot help to distinguish “street” and “highway”, since they are the common patterns in this situation. Therefore, the key of distinguishing one class of images from another is capturing the most discriminative information of each class. However, in existing feature learning methods, especially unsupervised feature learning methods, every class is equally treated in the training procedure. The systems are usually set to automatically learn image representations from the given training dataset without paying any special attention to discriminative information mining. To capture such discriminative patterns, we introduce image-level class constraints in local patch-level feature learning procedure, and aim to encode class-wise discriminative information in local image representations. The idea is to force the local feature patches extracted from the same category to be locally similar, while local feature patches extracted from different classes to be separable.

On the other hand, the compactness of image representation is also a key concern in feature learning, which directly influences the training/testing speed and cost of computational resources. Besides discriminative patterns, there also exist shareable patterns among similar categories, which can be used to roughly separate groups of similar classes from other dissimilar classes. Such patterns do not need to be duplicatedly learned by each category, which will not only consume unnecessary computational resources, but also lead to redundant feature transformation filters and feature dimensions. For example, “street” and “highway” are outdoor scenes, and “sky” exists in both categories, while “office” is an indoor scene. Consequently, knowing the existence of “sky” can help the system to distinguish outdoor from indoor scenes. The “sky” in “street” images and “highway” images are shareable patterns, learning “street sky” and “highway sky” separately

is meaningless. However, existing feature learning methods have rarely taken such shareable correlations into consideration, which makes the learned feature transformation filters highly redundant in most cases. Thus, we aim to encode shareable information in features, and help to learn compact image representations.

Moreover, contextual information is another important clue for image categorization. For images, there exist spatial contextual dependencies among local image regions at the same scale but different locations, and there also exist scale contextual dependencies among image regions at the same position but from different scales. For example, in an image of “car”, knowing the existence of “car window” can help to increase the representation preference of “car door” in the surrounded regions. What’s more, knowing the global-level scale pattern “car” can also help to increase the preference of representing “car window” and “car door” in the low-level scale local regions. However, in most general feature learning frameworks, such as the most popular deep convolutional neural networks [5], feature transformations are locally performed without considering the contextual correlations among different image regions. In contrast, we aim to model such spatial and scale contextual dependencies in image representation for better image classification.

1.3 Contributions of the Thesis

Motivated by the aforementioned ideas (refer to Section 1.2), three branches of works have been proposed to encode different types of visual clues in image feature representation. The main contributions of this thesis are summarized as below.

Firstly, in order to encode class-wise discriminative information in local feature representation, we propose a discriminative hierarchical feature learning algorithm.

Based on the hierarchical reconstruction independent component analysis [7, 8] structure, we encode the discriminative information by forcing the learned feature descriptors extracted from the same class to be locally similar, while descriptors from different classes to be separable. We also collect a fine-grained tree image classification dataset, called NTU Tree-51, which aims to recognizing fine-grained tree species at a distance. Testing our discriminative features on two popular object image classification benchmarks (Caltech 101, STL-10), and our NTU Tree-51, 3% to 7% accuracy improvements can be observed, which prove the effectiveness of encoding discriminative information in features.

Secondly, different from object categorization, there is no clear separation of foreground and background in scene images, different components combine together to form a specific scene scenario. Thus, we need to strengthen the discriminative power of image representation for distinguishing different scenes, while also maintain the shareable information among similar classes to keep the compactness of feature vectors. Our proposed deep discriminative and shareable feature learning (DDSFL) algorithm is designed for scene image classification. Particularly, for each category, a subset of the filters are activated in a data-adaptive manner, meanwhile sharing of filters among different categories is also allowed. Discriminative power of the filter bank is encoded by enforcing the features from the same category to be close to each other, while features from different categories to be far away from each other. To further enhance the discriminative power and make the training procedure more efficient and effective, we also propose two exemplar selection methods to iteratively select high-quality training data. Furthermore, our DDSFL features also show great complementary effect to the Caffe feature [5] pre-trained on ImageNet [9]. By combining DDSFL with Caffe feature, we can update the state-of-the-arts on three challenging scene image classification datasets:

Scene 15, UIUC Sports, MIT Indoor, and achieve very promising results on Pascal VOC 2012.

Thirdly, in order to make use of the spatial and scale contextual dependencies among different image regions, we build a class of end-to-end deep neural networks, which are called convolutional and hierarchical recurrent neural networks (C-HRNNs) for large-scale image categorization. Different from the existing popular deep convolutional neural networks (CNNs), where different image regions are locally processed layer by layer without taken the contextual correlations into consideration, we utilize our proposed hierarchical recurrent neural networks (HRNNs) to model spatial dependencies among image regions from the same scale but at different positions, and scale dependencies among image regions at the same positions but from different visual scales. By integrating CNNs with HRNNs, C-HRNNs are able to keep the discriminative representation power of CNNs, while also efficiently and effectively model the contextual dependencies in image representation. Testing on four most challenging large-scale object/scene image classification benchmarks, C-HRNNs can update the state-of-the-arts on Places 205, SUN 397, MIT indoor, and achieve very competitive results on ILSVRC 2012. In this work, we show that besides going deeper and wider, utilizing recurrent neural networks is another very promising direction to further strengthen the representation power of CNNs.

1.4 Organization of the Thesis

This thesis is organized as follows:

In Chapter 2, a general overview of relevant works on feature engineering/learning, and image categorization will be provided.

In Chapter 3, our proposed discriminative hierarchical feature learning algo-

rithm will be introduced, and applied to object image categorization benchmarks.

In Chapter 4, our proposed deep discriminative and shareable feature learning method will be introduced, and applied to scene image classification.

In Chapter 5, we propose a new class of end-to-end neural networks, which are called convolutional and hierarchical recurrent neural networks. They aim at modeling spatial and scale dependencies in image representation, and target on solving large-scale image categorization problems.

Chapter 6 concludes the thesis, and presents several future research directions.

Chapter 2

Related Works

This chapter includes a survey of important works related to this thesis. In Section 2.1, a general review of feature engineering/learning works will be introduced in detail. In Section 2.2, relevant works of the key components in image categorization will be reviewed.

2.1 Feature Engineering & Feature Learning

The performance of computer vision methods is heavily dependent on the applied features (data representation). High quality features are able to capture the most intrinsic patterns from the original raw data, and keep the feature representation compact and easy to process. There are mainly two lines of works: feature engineering (extracting features by preprocessing pipelines and data transformation designed with expert knowledge), and feature learning (automatically learn data-adaptive representations). In the rest of this section, a review of popular feature engineering and feature learning methods will be provided.

2.1.1 Feature Engineering

In the feature engineering area, hand-crafted features including SIFT [1], HOG [2], LBP[10] and GIST[11] are very popular, and they ruled the computer vision in the last decade. These hand-crafted features are manually designed by integrating human ingenuity and prior knowledge. Feature engineering methods can generally lead to good low-level image descriptors, and specific features can be elaboratively designed for specific data domain. For example, the most famous hand-crafted descriptor SIFT [1] has a sophisticated extraction procedure: Firstly, a 16x16 (can be other size) image patch should be taken; Secondly, the patch is evenly divided into 16 sub-blocks of 4x4 size; Thirdly, an 8 bin orientation histogram is created for each sub-block; Fourthly, by concatenating histograms from all the 16 sub-blocks, a 128-dimensional vector is achieved; Finally, after normalizing the vector to unit length, and applying a threshold of 0.2, a SIFT descriptor can be extracted. Thus, even though hand-crafted features are very powerful, designing such features is labor intensive, and they can hardly capture any information other than what have been defined by the designers. Therefore, they can hardly extract high visual level patterns or capture data-adaptive information.

Although very powerful, different hand-crafted features focus on different aspects, and one can only encode its pre-defined prior knowledge. There are numerous hand-crafted features, different features usually encode complementary information. Therefore, feature combination has been widely applied to computer vision tasks to improve the performance with the price of much higher dimensional representations. Nevertheless, not all the features are appropriate for every task, and some features might encode duplicated information. Thus, feature selection methods can be further employed to identify and remove irrelevant, redundant or

useless attributes which are not helpful or even harmful for the performance.

2.1.2 Feature Learning

Feature learning aims at automatically training models to transform raw data inputs to representations that can lead to better performance on tasks like classification, regression, etc. Feature learning expands the scope of traditional machine learning built based on feature engineering representations, and it is more towards artificial intelligence. The machines replace the human to “see” real-world data (training data), “learn” abstract concepts and knowledge (models) in more effective ways, and “describe” the real-world data (feature representation). However, real-world data (e.g. image, video, audio, etc.) is very diverse and noisy. Thus, it is important to design effective and efficient feature learning techniques, which can learn useful information from noise, and distinguish the confusions among different concepts.

Human brain is excellent in efficiently and robustly representing critical information. In artificial intelligence, inspired by the human visual system, hierarchy of representations with increasing level of abstraction are proposed and widely used. Although theoretically any hierarchical non-linear transformations can be approximated as close as possible with shallow structures, hierarchical structures are much more efficient than the shallow ones by learning representations with increasing levels of abstractions. Among numerous ways of feature learning, this review focuses on hierarchical feature learning method, which is formed by multiple layers of non-linear transformations constructed by a single algorithm, and target on extracting more abstract, compact, and ultimately informative representations.

Hierarchical feature learning methods can be roughly divided into two lines of

works: unsupervised learning (training data has no labels) and supervised learning (training data has labels).

The popular unsupervised deep learning methods include Deep Belief Network (DBN) [12–14], restricted boltzmann machines (RBMs) [15, 16], auto-encoders [17–22], and hierarchical reconstructed independent components analysis network [7, 19, 23], etc. To guarantee good performance, greedy layer-wise unsupervised pre-training is usually processed based on the previously learned lower-level layers. Such pre-training focuses on training one layer at a time while keep the previous layers fixed. Afterwards, new end-to-end unsupervised feature learning models would be initialized with the combination of the pre-trained network parameters/transformation matrices. Finally, a supervised classifier will be trained on the top with the other layers fixed. Furthermore, the whole network can also be fine-tuned/retrained based on the classification results to achieve better models. These unsupervised learning methods are very powerful when data labels are hard to collect, and they can get good performance on modeling general patterns. However, when class labels are available, unsupervised methods can hardly capture the class-specific discriminative patterns, which are very important for tasks like image categorization.

Supervised deep learning is another popular direction. There are lots of success frameworks, such as convolutional neural networks (CNNs) [5, 24–34], recurrent neural networks (RNNs) [35–41], sum-product networks [42]. Generally, the supervised hierarchical representation learning methods directly initialize network parameters/transformation matrices randomly, and typically utilize stochastic gradient descent and back-propagation to do optimization. The supervised algorithms are widely used in many practical systems in natural language processing area (RNN models have been widely used) and image recognition area (CNN models

have achieved great success). These methods can automatically learn discriminative information hidden in different classes. However, large-scale labelled training data need to be provided for the training procedure.

Among all these hierarchical feature learning frameworks, RICA, CNNs, and RNNs will be introduced in detail in the rest of this sub-section as three of the most representative feature learning schemes. Our proposed hierarchical feature learning methods in Chapter 3, Chapter 4, and Chapter 5 are designed based on these three learning schemes.

RICA Learning Scheme Unsupervised learning is one of the dominant directions in feature learning, among these unsupervised learning methods, the hierarchical reconstruction independent component analysis (RICA) learning structure proposed by Le and Zou *et al.* [7, 8] has shown its power in several challenging image and video recognition tasks. The learned features are shown to be robust and invariant to local transformations. The hierarchical network kept the advantages of the single layer learning module, while further improve its performance by capturing hierarchical visual level feature representations.

As shown in Figure 2.1, the basic single layer RICA feature learning module [23] consists of a linear auto-encoder term and a non-linear term. This combination has been proven to be very effective, besides, the resulting objective function can be easily optimized by unconstrained solvers like L-BFGS.

The non-linear function is set in an “energy pooling” manner. Given $x \in \mathbb{R}^D$ as a raw-pixel value training patch with dimensionality D , the RICA 1) uses the matrix $W \in \mathbb{R}^{D \times D}$ to linearly transform the input data into Wx ; 2) applies energy pooling [43] to represent the subspace structure of Wx , and get the non-linear

Figure 2.1: The basic single layer feature learning module structure of RICA. The blue units are input data, the green units are the square linear transformed data, while the red units are the output data after applying energy pooling. The output data corresponds to the learned features. (Best viewed in color)

transformed feature vector $q \in \mathbb{R}^{D/2}$:

$$q = \sqrt{H(Wx)^2} \quad (2.1)$$

where the square of Wx and the square root of $H(Wx)^2$ are processed element-wisely. $H \in \mathbb{R}^{D/2 \times D}$ is the subspace pooling matrix used to reduce feature dimension, each row of H selects and sums two adjacent feature dimensions without overlapping. While W can be learned by minimizing the following equation:

$$\sum_{j=1}^N \left(\|x^j - W^T W x^j\|_2^2 + \gamma \sum_{r=1}^{D/2} q_r^j \right) \quad (2.2)$$

in which, x^j and q^j denote the j -th training patch in the image space and feature space respectively, and N denotes the number of training patches. The first “auto-encoder” term is used to prevent the bases of W from degenerating. The second “sparse” term is used to ensure the sparsity of the learned feature descriptors.

Based on the unsupervised RICA learning scheme, discriminative and shareable information will be further encoded in Chapter 3 (integrate discriminative information only) and Chapter 4 (integrate enhanced discriminative information and shareable information).

Convolutional Neural Networks Since the success of Alex-net on ImageNet competition [5], CNNs are currently the most developed networks for solving large-scale image classification problems among all different deep feature learning structures. The core idea of CNNs is progressively learning more abstract (higher visual level) and more complex patterns: the first few layers focus on learning “Gabor like” low level local features (e.g. edges and lines); based on which, the middle layers target on learning parts of objects (e.g. “tires” and “windows” in the images with label “car”); the higher layers connect to the final image-level labels, and aim to learn representations of the whole images.

As shown in Figure 2.2, a typical end-to-end CNN consist of convolutional layer, pooling layer (e.g. max, average), normalization layer (e.g. response normalization), non-linear activation layer (e.g. rectified linear units), fully connected layer, regularization layer (e.g. dropout), and loss layer (e.g. softmax). Among all these layers, convolution, pooling and fully connected layers are the key layers:

- 1 Convolutional layers are proposed to extract more and more abstract/compact patterns layer by layer. To form a rich representation of input data, there are multiple local filters in each convolutional layer. While each filter is

Figure 2.2: Convolutional neural network structure.

replicated across the entire visual field, with such shared filters, only limited number of network parameters need to be learned and stored.

- 2 Pooling layers are used to summarize the values of feature responses over a sub local image region. It can reduce the feature dimension, and make the feature response maps more robust to input variance.
- 3 Fully connected layers are utilized to combine the outputs of the topmost convolutional layer into a high dimensional feature vector, and further learn hidden representations of the global image. Compared with other layers, fully connected layer has many more trainable parameters, thus, dropout layers are followed to prevent overfitting with the price of more training iterations to converge.

After feed-forwardly processing all the hierarchically constructed layers, the global image representation vector will directly connect to a loss layer (generally use softmax), and calculate the loss. Then back-propagation will be processed,

and loss will be backwardly propagated layer by layer in conjunction with stochastic gradient descent. Such forward and backward procedures will be iteratively processed for many iterations on batches of training data to optimize the network parameters.

Because of the great representation power of CNN features, in Chapter 4 and Chapter 5, CNNs will be combined/integrated with our proposed deep discriminative and shareable feature learning (DDSFL) method, and hierarchical recurrent neural networks (HRNNs) respectively. According to the experimental results, CNNs show great complementary effect to the DDSFL, and it can lead to good performance of HRNNs.

Recurrent Neural Networks In recent years, recurrent neural networks (RNNs) have achieved great success in natural language processing [36–38, 44–47]. RNNs [35, 48, 49] are neural networks for modeling sequential dependencies. Different from the CNNs, which are purely combined with “feed-forward” network layers, RNNs [35, 48] are “feed-back” neural networks designed for modeling contextual dependencies. Because of the connections from the previous states to the current ones, RNNs are networks with “memory”. Through such “feed-back” connections, the model can retain information about the past inputs, and it is able to discover correlations among the input data that might be far away from each other in the sequence. Because of the reuse of hidden layers, only a limited number of neurons need to be kept in the model, and testing is fast. Another advantage of RNN is the possibility to represent more advanced patterns of contextual dependencies in the sequential data.

In a typical recurrent network as shown in Figure 2.3, when processing a sequence with length T , its hidden layer feature $h^{(t)}$, and the predicted output label

$y^{(t)}$ at the state $t \in [1, \dots, T]$ can be represented as:

$$h^{(t)} = \psi_h (W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h) \quad (2.3)$$

$$y^{(t)} = \psi_y (W_{hy}h^{(t)} + b_y) \quad (2.4)$$

where $x^{(t)}$ is the t -th input data, $h^{(t)}$ denotes the hidden layer units, $y^{(t)}$ represents the output, W_{xh} , W_{hh} , W_{hy} are the transformation matrices between $x^{(t)}$ and $h^{(t)}$, $h^{(t-1)}$ and $h^{(t)}$, $h^{(t)}$ and $y^{(t)}$ respectively. b_h and b_o are the constant bias terms, while ψ_h and ψ_y are the non-linear activation functions.

The “memory” of RNNs is progressively increased by updating W_{xh} , W_{hh} , W_{hy} , and each state can be unfolded to a function of all the previous states. Ideally, the RNNs can remember whatever they have “seen”. Such historical information can be used to find meaningful connections between the current data and its context. Based on the basic RNNs, in Chapter 5, hierarchical recurrent neural networks will be proposed to model the spatial and scale contextual dependencies among local image regions for better image representation.

2.2 Image Categorization

Image categorization is one of the most fundamental problems in computer vision area. To understand the visual world, the very first step is to recognize the object/scene, which is the main application of this thesis. In this section, a brief review of current popular object/scene image categorization problems and techniques are given.

From almost a decade ago, image categorization methods start to recognize real-world image data from dozens and hundreds of different categories. For ex-

Figure 2.3: Structure of RNN. In a sequence of length T , for each state t , $x^{(t)}$ and $y^{(t)}$ are the input and output respectively. The hidden layer $h^{(t)}$ depends on the current state input $x^{(t)}$, and the previous state hidden layer output $h^{(t-1)}$. The input-to-hidden W_{xh} , hidden-to-hidden W_{hh} , and hidden-to-output W_{hy} transformation matrices are shared across the whole sequence.

Figure 2.4: The “BoW + SPM” image categorization framework.

ample, the Caltech 101 [50], Scene 15 [51], and MIT indoor [52]. Based on these datasets, lots of traditional image classification methods have been developed. The most popular “bag of words + spatial pyramid matching” (“BoW + SPM”) framework is shown in Figure 2.4. For a given input image, there are mainly five procedures: 1) Hand-crafted/learned local features are densely extracted to represent edges and corners; 2) Dictionary (codebook) is learned based on unsupervised clustering methods (e.g. K-means clustering); 3) Each local feature is encoded (e.g. vector quantization coding [51], sparse coding [53]) with the dictionary, and achieve histograms of counts or frequencies; 4) Applying SPM to pool (e.g. average pooling [51], max pooling [53]) histograms of local features over pre-defined coarse spatial bins, which aims to capture statistical feature data and rough spatial structures; 5) Finally, linear/non-linear classifiers (e.g. SVM [6] with kernel tricks) should be trained for classification.

In Chapter 3 and Chapter 4, our proposed feature learning algorithms learn local features, and follow the above traditional image classification framework.

In recent years, deep neural networks, especially convolutional neural networks(CNNs) have brought a new revolution in computer vision. Figure 2.2 shows a typical CNN structure. Generally, feed-forward structured deep neural networks (e.g. CNNs) are built in end-to-end way from the input raw data to the classifier. Usually, there are forward and backward paths : 1) Forward path: input training raw pixel images will be progressively processed to extract more and more abstract feature responses, and finally connect to classifier to calculate classification loss; 2) Backward path: classification loss is back propagated through stochastic gradient descent, and all the network weights are updated.

Their success mainly depends on two factors: 1) Powerful GPUs hardware, which can support parallel computing and greatly speed up the learning proce-

ture of large deep neural nets; 2) Large-scale datasets with ground truth labels, such as ImageNet [54] (1.2 million images), Places 205 [55] (2.5 million images) and SUN 397 [56] (0.1 million images). For the large-scale image classification tasks, the traditional hand-crafted feature based methods work much less efficient than deep neural networks. Take the Alex-net [5] as an example, with a NVIDIA K40 GPU, only 2ms will be taken to test an image, which is orders of magnitudes faster than that of hand-crafted feature based methods. Based on current published results, deep neural networks also perform much better than hand-crafted features for large-scale problems. Since the hand-crafted features can hardly model the tremendous divergence exist in large-scale dataset, in contrast, automatically learning from data is more practical for such application. Another reason is that the good performance of traditional hand-crafted features heavily depends on kernel SVM, however, because of the calculation procedures of kernels are very slow, it is unfeasible for large-scale image categorization. Thus, deep neural networks are more appropriate for manipulating large-scale data.

In Chapter 5, our proposed convolutional and hierarchical recurrent neural networks follow the above deep neural network categorization framework.

Besides the above approaches, there are also some other directions to solve image categorization problem. One important branch is the non-parametric methods, in which the most popular methods are the k-nearest neighbors (KNN) based methods [57–59]. KNN is instance-based learning (lazy learning) method, which means the function is only locally processed based on the descriptive and inferential statistics, and all computation is deferred until classification. Non-parametric methods are usually very efficient, and the structures are more flexible to the data. However, their performances are generally unsatisfactory compared with state-of-the-art frameworks, such as the deep neural networks. Another important branch

is the semi-supervised methods [60–63], which aims at exploring the underlying geometric structure of the data distribution and generate additional unlabeled training data. These methods are very useful when training data samples are not enough to support effective recognition.

Image categorization is also highly correlated to some other computer vision applications, such as object recognition and detection, in which the position of object instance is a key factor. One of the most famous works is the part-based models [64], which describes images based on mixtures of multiscale deformable part models. Another success framework is R-CNN [30], which utilizes CNNs to bottom-up region proposals.

Chapter 3

Discriminative Hierarchical Feature Learning

Recently, deep feature learning has shown great success in automatic feature learning. There are many deep feature learning methods focused on generic feature learning with unsupervised or supervised methods. However, discriminative feature learning still haven't been paid much attention. We argue that it is important to investigate discriminative information. Pure generic feature learning may learn general patterns, they are not helpful for distinguishing the subtle differences among different categories, which is the key of classification tasks. Thus, in this chapter, we propose a discriminative hierarchical feature learning method, which learns non-linear transformations to encode discriminative information in feature representations.

3.1 Introduction

We assume that the local image patches contain the image-level class-specific information. Based on this assumption, we assign all the local image patches with the same class labels as the images they extracted from. To get more discriminative local image representation, we aim to learn such a feature space, in which, the feature patches (transformed image patches) from the same class are close together, while the feature patches from different classes are separable from each other.

However, local patches from the same class can be highly diverse. Simply forcing all the feature patches from the same class to be close will bring too much noise. Instead, a feature patch only needs to be close to a small subset of the patches from the same class. Thus, we introduce the “Patch-to-Class” distance (P2CD) (inspired by the “Image-to-Class” distance proposed in naive bayes nearest neighbour [58]) to directly measure the distance between each feature patch and its nearest neighbour patches from difference classes. As shown in Figure 3.1, this framework forces the training feature patches to be close to their corresponding classes (positive), while to be far away from other classes (negative), which means shortening $P2CD(q, NN_p)$ while elongating $P2CD(q, NN_n)$.

Our discriminative hierarchical feature learning framework is built based on the hierarchical reconstruction independent component analysis (RICA) structure [7, 8, 23] (our method can also be applied in other feature learning frameworks involving learning transformation matrix). As shown in the orange box on the left of Figure 3.2, the first layer features are learned through small input image patches (yellow boxes), then they are convolved with a larger region (red box) to generate the inputs to the second layer. The final features are the combination of outputs of both layers. Since we focus on learning discriminative multi-layer local features, we

Figure 3.1: Schematic diagram of our discriminative information encoding scheme. The square in green denotes the training patch; the circles in red denote patches from the same category (positive); the triangles in blue denote patches from other categories (negative). This framework aims to reduce $P2CD(q, NN_p)$, while elongate $P2CD(q, NN_n)$.

simply follow the Bag-of-Words (BoW) to get global image representation, and use linear SVM to do classification. Our overall object recognition pipeline is shown in Figure 3.2.

3.2 Naive Bayes Nearest Neighbour

Before we get into the details of our discriminative feature learning method, we will firstly give a brief review of the Naive Bayes nearest neighbour (NBNN) algorithm.

In our discriminative feature learning, we expect the learned features from the same class to be more similar while features from unrelated classes to be separable. To reach this goal, a proper local distance measurement is required. The first local distance learning function was proposed by Forme *et al.* in [65, 66], they learnt a weighted “Image-to-Image” (I2I) distance function for each training image, which is calculated as a combination of distances between local feature

Figure 3.2: Illustration of our discriminative feature learning and object recognition framework. Firstly, the discriminative hierarchical feature learning module extracts multi-layers of local features. Then these features will be processed BoW to get global image representation. Finally, linear SVM will be applied for classification. This chapter focus on the discriminative hierarchical feature learning module. In its first layer, features are learned from small image patches (yellow boxes). In its second layer, features are learned from larger image patches (red box), whose inputs can be achieved by concatenating the over-complete first layer features extracted within a larger image area (red box), and then applying PCA for dimension reduction and data whitening. In each layer, the blue units are input data, the green units are the squared linear transformed data, while the red units are the output learned features after applying energy pooling. (Best viewed in color.)

patches. After that, Boiman [58] proposed NBNN based “Image-to-Class” (I2C) local distance measurement, and it has been proven to outperform I2I methods in visual recognition area.

NBNN is a feature-based image classifier, it employs nearest neighbour distance in feature space instead of image space. It achieved impressive degree of accuracy. Because of its simplicity and high-efficiency, a lot of subsequent works based on

this idea have been performed, such as local NBNN [67], I2C large margin metric learning [68], NBNN kernel [69], etc.

However, NBNN has only been used as a classifier, we extend it to our feature learning scheme. We can measure the distances between local features and object categories, and learn discriminative feature representation afterwards.

3.3 Discriminative Hierarchical Feature Learning

We aim to learn a transformation matrix to transform the local image patches from the original image space to the discriminative feature space. Recently, RICA has shown its power in several challenging image and video recognition tasks. Thus, we build our discriminative feature learning framework based on the hierarchical structure developed in [7] and the single layer feature learning method RICA [23]. This combination has been proven to be very effective, besides, the resulting objective function is easy to optimize. The detailed of RICA has been introduced in Section 2.1.2, interested readers may refer to [7, 23] for more details.

3.3.1 Single Layer Discriminative Feature Learning

The basic RICA learning method described in Section 2.1.2 is totally unsupervised, and it can hardly capture discriminative information hidden in different classes. Ideally, for classification, we expect the learned features to be close to the features from the same class, while to be far away from the features from other classes. Thus, we propose the following framework to encode discriminative information.

In our discriminative learning scheme, we aim at maximize the following func-

tion for a training feature patch q :

$$\frac{P(c|q)}{P(\bar{c}|q)} = \frac{P(q|c) \cdot P(c)}{P(q|\bar{c}) \cdot P(\bar{c})} \quad (3.1)$$

where $\bar{c}$ denotes all the classes except class c , and q is from c .

Assuming the class priors are equal, then the posteriors are equal to the likelihoods, which can be approximated by applying the Parzen window estimator as described in [58]:

$$\hat{P}(q|c) = \exp\left(-\frac{1}{2\sigma^2}\|q - NN_c(q)\|_2^2\right) \quad (3.2)$$

in which, $NN_c(q)$ is the nearest neighbour belonging to class c of the training patch q in the feature space. $\|\cdot\|_2$ is the l_2 norm. If we further take the log probability and ignore the constant, we can rewrite the right-hand side of Equation 3.2 as $-\|q - NN_c(q)\|_2^2$, which can be considered as the negative ‘‘Patch-to-Class’’ distance (P2CD). Then Equation 3.1 can be written in a simplified form:

$$\log \frac{P(q|c)}{P(q|\bar{c})} = -\|q - NN_c(q)\|_2^2 + \|q - NN_{\bar{c}}(q)\|_2^2 \quad (3.3)$$

where $NN_{\bar{c}}(q)$ is the nearest neighbour of q in the feature space, and it is from classes other than c . Based on our discriminative learning method, we get the representation of the single layer learning module:

$$\begin{aligned} & \min_W E_u + \eta E_s \\ \text{where } E_u &= \sum_{j=1}^N \left(\|x^j - W^T W x^j\|_2^2 + \gamma \sum_{r=1}^{D/2} q_r^j \right) \\ E_s &= \sum_{j=1}^N \left(\|q^j - NN_c(q^j)\|_2^2 - \|q^j - NN_{\bar{c}}(q^j)\|_2^2 \right) \\ q &= \sqrt{H(Wx)^2} \end{aligned} \quad (3.4)$$

where E_u represents the unsupervised RICA term used to enforce low reconstruction error and sparsity, E_s represents the supervised discriminative constraint. $x^j \in \mathbb{R}^D$ and $q^j \in \mathbb{R}^D$ (D is the vector dimension) denote the j -th training patch in the image space and feature space respectively. $W \in \mathbb{R}^{D \times D}$ is the trained transformation matrix, $H \in \mathbb{R}^{D/2 \times D}$ is a subspace pooling matrix to summerize every two adjacent feature dimensions together and reduce feature dimension, both the square of Wx and the square root of $H(Wx)^2$ are processed element-wisely. N denotes the number of training patches. γ and η are the tradeoff parameters used to control the level of sparsity and discriminative power.

We adopt the gradient descent to optimize the object function 3.4. The partial derivatives of E_u and E_s with respect to W_{mn} (an element at of W) are:

$$\begin{aligned}
 \frac{\partial E_u}{\partial W_{mn}} &= \sum_{j=1}^N \frac{\partial \|x^j - W^T W x^j\|_2^2}{\partial W_{mn}} + \gamma \sum_{j=1}^N \sum_{r=1}^{D/2} \frac{\partial q_r^j}{\partial W_{mn}} \\
 &= \sum_{j=1}^N \frac{\partial E_{ae}}{\partial W_{mn}} + \gamma \sum_{j=1}^N \sum_{r=1}^{D/2} \frac{\partial E_{sparse}}{\partial W_{mn}} \\
 \frac{\partial E_s}{\partial W_{mn}} &= \sum_{j=1}^N \frac{\partial \|q^j - N N_c(q^j)\|_2^2}{\partial W_{mn}} - \sum_{j=1}^N \frac{\partial \|q^j - N N_{\bar{c}}(q^j)\|_2^2}{\partial W_{mn}} \\
 &= \sum_{j=1}^N \frac{\partial E_{pos}}{\partial W_{mn}} - \sum_{j=1}^N \frac{\partial E_{neg}}{\partial W_{mn}} \\
 \frac{\partial E_{sparse}}{\partial W_{mn}} &= H_{rm} (W_m x^j) x_n^j / \sqrt{H_r (W x^j)^2} \\
 \frac{\partial E_{ae}}{\partial W_{mn}} &= -4 W_m x^j x_n^j + Tr \left[\left[2 W^T W \left(x^j (x^j)^T \right) \right]^T (W^T J^{mn} + J^{nm} W) \right] \\
 \frac{\partial E_{pos}}{\partial W_{mn}} &= 2 (q^j - N N_c(q^j))^T \left(\frac{\partial q^j}{\partial W_{mn}} - \frac{\partial N N_c(q^j)}{\partial W_{mn}} \right) \\
 \frac{\partial q^j}{\partial W_{mn}} &= H_m (W_m x^j) x_n^j / \sqrt{H(W x^j)^2}, \quad (J^{mn})_{kl} = \delta_{mk} \delta_{nl}
 \end{aligned} \tag{3.5}$$

Where E_{sparse} and E_{ae} correspond to the sparse term and auto-encoder term. E_{pos} and E_{neg} are intermediate terms which correspond to the positive and negative “Patch-to-Class” distance respectively. $\partial E_{neg}/\partial W_{mn}$ has the same form as $\partial E_{pos}/\partial W_{mn}$, $\partial NN_c(q^j)/\partial W_{mn}$ and $\partial NN_{\bar{c}}(q^j)/\partial W_{mn}$ have the same form as $\partial q^j/\partial W_{mn}$. The transformation matrix W can be updated with step size α until convergence: $W = W - \alpha \times (\partial E_u/\partial W + \eta \partial E_s/\partial W)$.

3.3.2 Hierarchical Learning Structure

Though the single layer feature learning module can achieve good performance, it still has some limitations. For example, it is not able to share statistical information among different local features, and it cannot extract information from multiple visual levels. To get higher level visual representations that can not only tolerate non-trivial transformations in small local areas, but also capture contextual information of the first layer features, we leverage a multi-layer scheme to learn hierarchical features. Here we adopt a two-layer framework: in the first layer, feature learning is performed on small image areas (16x16 image patches) to extract the first layer discriminative features; while in the second layer, higher level image representation is learned from bigger image areas (32x32 image patches) to get the second layer discriminative features.

To get the second layer inputs, convolution is applied to the first layer outputs at multiple grid locations to get a highly over-complete set of first layer features in the 32x32 image areas. Concatenating these first layer features will generate a high-dimensional representation, which cannot be efficiently processed. Thus, PCA is applied for dimension reduction and data whitening, the output of which is the input to the second layer. We get the final feature descriptors by concatenating

the output features of both layers as shown in Figure 3.2.

3.3.3 Approximation

In each layer, it's time consuming to search nearest neighbour from a large collection of image patches. Especially in our case, the membership of nearest neighbours $NN(q)$ of each training patch x change when the feature transformation matrix W updates. For simplification, firstly, we fix $NN(q)$ and update W until W converges to a suboptimal value. Secondly, we search nearest neighbours in the learned sub-optimal feature space, and renew the membership of nearest neighbours $NN(q)$. We iterate these two steps for several times. In our experiments, when the number of iterations increases, the performance slightly increases. For efficiency consideration, we merely apply one iteration. We initialize W as the matrix learned by the basic learning module without discriminative term, and then search for $NN(q)$ based on this W , and update W afterwards. Furthermore, to speed up the procedure of nearest neighbour search, we use FLANN [70], which is a library making use of multiple randomized k-d trees to achieve fast NN approximation.

3.4 Experiments and Analysis

We test our discriminative hierarchical feature learning algorithm on two general image classification benchmarks: Caltech-101, STL-10, and one new fine-grained image classification dataset we have collected: NTU Tree-51. To make a better comparison with other methods, we only use gray-scale images. In layer 1, we learn on 16x16 patches (256 dimensional input data), and get 128 dimensional first layer features. In layer 2, based on the learned first layer features, we convolve them within the larger 32x32 patches with a stride of 2, thus each second layer patch

covers 81 first layer patches. We concatenate the 81 first layer feature responses and get 10,368 dimensional data. Afterwards, following the settings in [8], we apply PCA to produce 300¹ dimensional input data for the second layer. After processing layer 2, we get 150 dimensional second layer features. Since this chapter focuses on local feature learning, we simply employ the most general settings in the BoW framework. We use spatial pyramid matching [51] to get the global image representation, and linear SVM as classifier.

To ensure the accuracy of nearest neighbour search, we densely extract patches from the training images to build the positive and negative patch pool. Additionally, we discard patches extracted from the low contrast areas for denoising. Specifically, for each class, we densely extract 1000-2000 patches per training image for the first layer, and 200-400 patches for the second layer to build the positive patch pool. Meanwhile, we randomly select patches from images belonging to negative categories, and build the negative patch pool with 10 times the size of the positive pool. Finally, we randomly select 10% of the patches from each class-specific positive patch pool as the training patches, and learn features based on 100,000-500,000 training patches in the first layer, and 30,000-100,000 training patches in the second layer.

3.4.1 Results

Caltech-101 [50] has images from 101 different object classes with high intra-class variance. Some image examples are shown in Figure 3.3. There are 31 to 800 images in each class. Following the most general settings, we randomly select 30 images per class for training, and 50 images for testing. We resize all the images

¹According to the cross validation results, higher dimensional data will lead to slightly better performance with the cost of much more computation time, and the model is easy to get overfeat. While lower dimensional data will lead to loss of information, and the performance will drop.

Figure 3.3: Examples from Caltech-101 dataset.

to 150x150 pixels. The numerical results of our discriminative learning method and other algorithms are reported in Table 3.1. We test the hierarchical RICA with training patches randomly extracted from training images, and get 66.7% in accuracy as the baseline. With our discriminative learning method, the accuracy can be improved to 73.0%, which is 6.3% higher than the baseline. Comparing with the results of SPM [51] (which utilize exactly the same framework except different features), 8.4% performance gain can be achieved by just replacing the traditional hand-crafted feature SIFT to our discriminative features. Moreover, if we further apply more sophisticated coding methods, such as LLC [53] instead of SPM, the results can be further improved (around 8%). While since this work focus on feature learning instead of feature coding, we just provide the numerical results based on the simple SPM framework.

STL-10 [71] is a newly proposed challenging dataset for deep learning networks

a Examples of STL-10

b Examples of NTU Tree-51

Figure 3.4: (a) Examples from STL-10 dataset. (b) Examples from four different tree species of NTU Tree-51 dataset. It is a challenging fine-grained image classification dataset with large intra-class variance and small inter-class variance.

Figure 3.5: Per-category recognition accuracy on STL-10 for comparison between our hierarchical feature learning method and the original hierarchical RICA.

and unsupervised learning, which contains 96x96 pixel images from 10 classes as shown in Figure 3.4a. We only use the provided labelled data: 5,000 for training and 8,000 for testing. The training datasets are predefined in 10 folders, where each folder contain 1,000 training images. According to the testing protocol, we train on the pre-defined folders, and use the average results as the final testing accuracy. The results are shown in Table 3.2. As this dataset is very challenging, the accuracy is relatively low, but we can still get 2.3% compared with the baseline, and achieve 56.7% in accuracy. The per-category recognition accuracy can be checked in Figure 3.5. Our method outperforms the original hierarchical RICA on most of the categories, except “cat” and “dog”, this is mainly because the images in these two classes are very diverse, which makes our discriminative learning not accurate. Some false positive images are shown in Figure 3.6.

NTU Tree-51 is a fine-grained image dataset we have collected, which is shown in Figure 3.7. NTU Tree-51 is collected for recognizing fine-grained tree species at a distance. All the images were cropped from Google Street View images, which were captured continuously from a distance on a moving vehicle. This dataset

Figure 3.6: False positive image examples in STL-10, hierarchical feature learning failed cases. In each column, the text in red shown on the top row denotes the true label, while the texts under each images denote their false labels.

contains 2613 street view tree images in total, which is composed of 51 common tree species in Singapore, and each species contains 30-70 samples. This dataset is challenging because of its large intra-class variance, and relatively small inter-class variance. As shown in Figure 3.4b, the images from each class has very huge intra-class variances and relative small inter-class variances. We resize all the images to 150x150 pixels, for each species, we use 20 images for training, and the rest for testing. The numerical results are shown in Table 3.3. With our discriminative term, we can improve the baseline with 7.2% in accuracy. The per-category accuracy of our discriminative method compared with the original hierarchical RICA method is shown in Figure 3.8. Our method outperforms the original deep RICA algorithm in 34 categories and equal performs in 8 categories. Our method get worse than deep RICA in 9 categories, most of these species are highly confusing ones, which make our discriminative feature learning not accurate.

Furthermore, according to the comparison results of using the first layer only and two layers shown in Table 3.1, 3.2, and 3.3, with the stacked second layer,

Figure 3.7: The NTU tree-51 dataset. Each image is an instance of a tree species.

the performance can be significantly improved, thus, the hierarchical structure is crucial.

Algorithm	Acc.
SPM [51]	64.6%
Hierarchical RICA [7, 23]	66.7%
NBNN [58]	70.4%
local NBNN [67]	71.9%
LLC [53]	73.4%
Hierarchical SC [72]	74.0%
CRBM [73]	77.8%
Ours (first layer)	66.3%
Ours (two layers)	73.0%

Table 3.1: Results on Caltech-101.

Algorithm	Acc.
K-means (Triangle) [71]	51.5%
RICA [23]	52.9%
Hierarchical RICA [7, 23]	54.4%
Sum-Product networks[74]	62.3%
Ours (first layer)	53.3%
Ours (two layers)	56.7%

Table 3.2: Results on STL-10.

Algorithm	Acc.
SPM [51]	69.6%
Hierarchical RICA [7]	70.9%
LLC [53]	75.3%
Ours (first layer)	74.9%
Ours (two layers)	78.1%

Table 3.3: Results on NTU tree-51.

3.4.2 Parametric Analysis

We observe the sparse term γ does not bring much influence to the performance, hence we fix it as a constant for all the datasets (50 in the first layer, and 1 in the second layer), and focus on η : weight of the discriminative term in layer 1 and layer 2. In this section, we experimentally investigate how they may affect the performance.

Figure 3.8: Per-category recognition accuracy of NTU tree-51 for comparison between our hierarchical feature learning method and the original hierarchical RICA. Our method vs hierarchical RICA: on 34 categories better, 8 equal, and 9 worse.

We vary the value of the regularization parameter η in layer 1 and layer 2 separately. By applying cross validation, we get the results as shown in Figure 3.9. The performance numbers do not change very dramatically. With layer 1 only, as shown in Figure 3.9a, when $\eta = 10$, our hierarchical discriminative method can bring 5% improvement compared to the single layer RICA. Figure 3.9b shows the comparison result of our hierarchical discriminative method versus the hierarchical RICA. Here we fix the η in layer 1 as 10. In layer 2, when $\eta = 0$, it corresponds to the result of only applying discriminative learning on layer 1, and using the basic RICA in layer 2. As the accuracy steadily increase when the value of η increase, we can get 3% improvement when $\eta = 20$. This indicates that our discriminative learning method not only improves the performance of the first layer, but also further improves the performance of the second layer. Generally, setting η to 10 for both layers will lead to good performance on all the datasets.

 a Influence of η in layer 1

 b Influence of η in layer 2

Figure 3.9: Accuracy of our hierarchical features on Caltech-101 versus the weight of the discriminative term. The bottom straight lines represent the accuracy results of applying the hierarchical RICA, while the curves indicate the accuracy of our discriminative hierarchical method.

3.5 Conclusion

In this chapter, we proposed a discriminative hierarchical feature learning algorithm, which aims to force the features from the same class to be close, while features from different classes to be separated. We propose P2CD to measure the distance between a feature descriptor and a class. By applying a two-layer discriminative learning method, we obtain a hierarchical feature representation

that can not only represent local discriminative features, but also express multiple visual level features with larger receptive fields by applying convolution and stacking. On two general object recognition benchmarks and a new fine-grained image classification dataset, we experimentally show that learning discriminative features significantly improve the performance.

However, there also exist shareable patterns among similar classes, making use of such information can help to learn more compact feature representations. Furthermore, the discriminative information can also be more efficiently learned if the training data are pre-selected high quality data. The discriminative feature learning algorithm introduced in this chapter serves as a basis of our subsequent works, and extension of this work will be presented in Chapter 4.

Chapter 4

Deep Discriminative and Shareable Feature Learning

Different from object images, there is no clear foreground/background in scene images. In order to enhance the discriminative power of the discriminative features (refer to Chapter 3), while also making use of the shareable patterns existing among different scene image categories to learn compact image representations, in this chapter, we propose a new local feature learning approach named Deep Discriminative and Shareable Feature Learning (DDSFL). DDSFL aims to hierarchically learn feature transformation filter banks to transform raw pixel image patches to features. The learned filter banks are expected to: (1) encode shareable visual patterns of a flexible number of categories; (2) encode discriminative information; and (3) hierarchically extract patterns at multiple visual levels.

4.1 Introduction

As shown in Figure 4.1, in each feature learning layer, we aim to learn an over-complete filter bank, which is able to cover the variances of patches from different classes, meanwhile keeping the shareable correlation among similar classes and discriminative power of each category. Intuitively, this goal can be reached by randomly selecting training patches, and learning filter banks for each class independently, then concatenating them together afterwards. However, there are several problems: (1) some of the patterns are shared among some classes, repeatedly learning filters corresponding to the similar patterns are neither compact nor computationally efficient, meanwhile the feature dimension will increase linearly with the number of classes, which limit the learning methods to be only applicable to small datasets; (2) discriminative power can hardly be fully exploited, since the class specific characteristics are generally subtle and not obvious without comparing with other classes; (3) In most cases, images are dominated by noisy or meaningless patches, learning filters from randomly sampled image patches will increase the learning cost and depress the performance.

To learn compact and effective filter banks, each category is forced to only activate a subset of the global filters during the learning procedure. Beyond reducing feature dimension, sharing filters can also lead to more robust features (especially effective for scene images). Images belonging to different classes do share some information (e.g. in scene images, both “computer room” and “office” contain “computer” and “desk”). The amount of information sharing depends on the similarity between different categories. Hence, we allow filters to be shareable, meaning that the same filters can be activated by a number of categories. We introduce a binary selection vector to adaptively select which filters to share, and

among which categories.

To improve the discrimination power, similar to Chapter 3, we force the features from the same category to be close and the features from different categories to be far away. (e.g. patches corresponding to bookshelf in “office” can hardly be found in “computer room”). However, the local patches from the same categories are very diverse. Therefore, we propose to measure the similarity by forcing a patch to be similar to a subgroup of the training samples from the same category instead. Furthermore, not all the local patches from different classes need to be separable. Thus, we relax the discriminative term to allow sharing similar patches across different classes, and focus on separating the less similar patches.

To improve the quality of the filters and efficiency of the learning procedures, we propose two exemplar selection schemes to select high-quality training data. The proposed methods aim to remove the noisy training patches that commonly exist in many different classes, and select the patches that contain both shareable and discriminative patterns as the training data to learn the filter banks.

Furthermore, supported by lots of previous deep feature learning works, hierarchically extracting increasing visual level features can help to get more abstractive and useful information. In this chapter, we build a three layer learning framework, as shown in Figure 4.2. Specifically, we firstly learn the first layer features from small (16x16) raw pixel value image patches. Then for the higher layers, we convolve the previous layer features within a larger region (32x32 and 64x64 for the second and third layer respectively) as the input to PCA, and use the reduced dimensional data (we set the dimension to 300 for all the layers) as the input to train the current layer filter bank (we set to learn 400 filters for each layer). Finally, we combine the features learned by all the three layers as DDSFL feature. The complete feature learning and testing frameworks are shown in Figure 4.3.

In Section 4.5, we test DDSFL features on three widely used scene image classification datasets: Scene 15, UIUC Sports, MIT Indoor, and we also test on PASCAL VOC 2012. The experimental results show that DDSFL can outperform most of the existing methods, and it also has significant complementary effect with the Caffe [75] features (Alex-net [5] pre-trained on ImageNet [9]).

4.2 Discriminative Training

There are some previous works take discriminative information into consideration. For example, [76–79] learn discriminative dictionaries to encode local image features. Another line of works are usually known as mid-level feature searching or exemplar based representation [80–85]. They represent images in terms of weakly-supervise mined discriminative regions of interest (multi-scale patches, usually correspond to middle or high level patterns) or images. Afterwards, images can be represented with the max pooled responses of such mid-level patterns. In contrast, DDSFL not only focus on learning class-specific discriminative filters, but also learning shareable filters which encode common patterns exist among different categories. Combining both discriminative and shareable filters together, the complete global filter bank can be obtained to do the feature transformation. To the best of our knowledge, this hasn’t been done before. One of the most relevant works is [86], which aims at learning category-specific dictionaries for each category and a shared dictionary for common patterns among all the categories. While our DDSFL focuses on encoding discriminative and shareable information in local feature representations. The definitions of “shared” are also different, we define shared patterns as those exist among groups of classes, and we remove the common patterns among all classes as noise via exemplar selection procedure. Moreover,

Figure 4.1: Illustration of single layer discriminative and shareable feature learning scheme. $w_1, \dots, w_D$ represent the filters in the global filter bank W . For each class, we force it to activate a small subset of filters to learn class-specific patterns, and different classes can share the same filters to learn shareable patterns. Finally, the feature of a image patch x_i can be represented as $f_i = \mathcal{F}(Wx_i)$. (Best viewed in color)

Figure 4.2: The hierarchical structure of DDSFL. In the input layer, 16x16 raw image patches are extracted, and be transformed to the first layer features with the first partially connected (in the testing step, fully connected) layer. Afterwards, combining these features correspond to 32x32 image areas, and applying PCA to prepare the reduced dimensional inputs to the second layer. Similarly, the third layer can be processed. Within each partially connected layer, the solid arrows indicate that the corresponding filters are activated for class c , while the gray dashed arrows means filters are not activated. (Best viewed in color)

Figure 4.3: DDSFL image classification training and testing framework. The red dashed box on the top describes the training procedure, while the blue dashed box below indicates the testing process. For the training procedure: (a) Input the original image or feature outputs of the previous layer (if there are any), and densely extract image patches or local features (yellow boxes); (b) Select exemplars (red boxes) for training; (c) Process DDSFL training module and learn filter bank. For the testing procedure: (d) Apply the learned filter bank W to the original input image or previous layer features, and densely extract the DDSFL feature for the current layer; (e) Process LLC [53] and SPM [51] afterwards to convert the local features to global image representation, and applying linear SVM to do final classification. In the DDSFL module, $w_1, \dots, w_D$ represent the filters in the global filter bank W . During training step, for each class, we force it to activate a small subset of filters (the activated filters have been highlighted with different colors), and different classes can share the same filters. Finally, the DDSFL feature of a image patch x_i can be represented as $f_i = \mathcal{F}(Wx_i)$. (Best viewed in color)

our features can be directly applied to the feature coding framework described in [86] to build a more powerful categorization system.

From another aspect, compared with the previous exemplar based methods, our proposed exemplar selection methods are used to get more informative local patch level training data for the feature learning modules, thus the previous methods (middle or image level exemplar selection) are not suitable in this scenario. Furthermore, in [87, 88], object part filters at the middle level are shared to represent a large number of object categories for object detection. While we only use weakly supervised image level labels, and develop a nearest neighbor based maximum margin method to learn discriminative feature transformation matrix.

4.3 Deep Discriminative and Shareable Feature Learning

In this section, the hierarchical structure of Deep Discriminative and Shareable Feature Learning (DDSFL) framework will be introduced. For each single layer DDSFL, its three learning components will be introduced in detail, and an alternating optimization strategy will be provided afterwards.

4.3.1 DDSFL Hierarchical Structure

we use $x_i \in \mathbb{R}^{D_o}$ to denote the vectorized raw pixel values of an image patch, and $X = \{x_1, x_2, \dots, x_N\}$ to denote the collection of all the N training patches densely extracted from the training images (the patches are weakly labeled by the image level class labels). As shown in Figure 4.2, there are $L = 3$ layers in DDSFL framework. In each layer of DDSFL, we aim to learn a feature transformation filter

bank $W^{(l)} \in \mathbb{R}^{D \times D_0}$, $l = \{1, \dots, L\}$, in which, l is the layer indicator, D_0 denotes the dimension of the input data, D is the number of filters, and each row of $W^{(l)}$ represents a filter. By multiplying $W^{(l)}$ with an input vector $x_i^{(l)}$, and applying an activation function $\mathcal{F}(\cdot)$, we expect to generate feature $f_i^{(l)} = \mathcal{F}(W^{(l)}x_i^{(l)})$, which is discriminative and as compact as possible. Afterwards, by convolving the $f_i^{(l)}$ in a larger image receptive field, and process PCA, we can get the input $x_i^{(l+1)}$ of the next layer.

4.3.2 Single Layer DDSFL Learning Components

In each single layer, DDSFL framework aims to learn features that are able to: 1) preserve the information of the original data; 2) compactly represent patterns shared among similar classes; and 3) effectively encode class specific discriminative patterns. To achieve these goals, three corresponding learning components are proposed: the global unsupervised term, shareable pattern learning term, and discriminative information encoding term.

For the compact consideration, we force each class to activate a subset of filters in $W^{(l)}$, and using the training patches from the similar classes (classes activate the same filters) to train the shared filters. To encode discriminative information, we force the features from the same class to be similar, while features from different classes to be as different as possible. In the following subsections, we will introduce the details of the three learning components, and for the sake of convenience, we ignore the l in all the variables, and focus on the learning procedure of a single layer DDSFL.

4.3.2.1 The Global Unsupervised Term

To ensure the learned filter bank W is able to preserve the information in the original data after processing feature transformation, meanwhile, also be robust to the small variance exist in neighborhood spatial areas, we firstly introduced a global reconstruction term and build an auto-encoder like learning scheme:

$$\begin{aligned}
 L_u &= \sum_{i=1}^N \mathcal{L}_r(x_i, W) + \xi \sum_{i=1}^N \mathcal{L}_{sp}(x_i, \Omega(x_i)) + \lambda_1 \sum_{i=1}^N \|f_i\|_1 \\
 \text{where } \mathcal{L}_r(x_i, W) &= \|x_i - W^T W x_i\|_2^2 \\
 \mathcal{L}_{sp}(x_i, \Omega(x_i)) &= \frac{1}{M} \sum_{m=1}^M \|f_i - \Omega_m(f_i)\|_1 \\
 f_i &= \mathcal{F}(W x_i), \quad \mathcal{F}(\cdot) = \text{abs}(\cdot)
 \end{aligned} \tag{4.1}$$

where $\mathcal{L}_r$ is the global reconstruction error with respect to global filter bank W and training data x_i . This auto-encoder [17, 23] style reconstruction cost penalization term (refer to basic RICA learning method described in Section 2.1.2) can not only prevent W from degeneration, but also allow W to be over-complete. The spatial denoising term $\mathcal{L}_{sp}$ is used to minimize the difference between the training patch and its spatial neighbor patches, and makes the learned features to be robust to small spatial variance¹. For each training data x_i , we randomly sample M patches (extract different size patches, and resize them to have the same size with x_i) from the spatial neighborhood image areas, which are denoted as $\Omega(x_i)$. $\|\cdot\|_1$ and $\|\cdot\|_2$ are the l -1 norm and l -2 norm respectively. The term $\|f_i\|_1$ is used to enforce the sparsity of the learned feature f_i . To make the optimization easier while also preserving the quality of the learned features, we use the non-negative activation function in [7, 8], and set $\mathcal{F}(\cdot) = \text{abs}(\cdot)$. Then the sparse term $\|f_i\|_1$ degenerates

to the summation of all the elements in vector f_i .

4.3.2.2 Shareable Pattern Learning Term

Although the above global unsupervised term can already lead to satisfactory results, class information has not been used, which is critical for classification problem. Furthermore, we aim to learn a compact filter bank, rather than a redundant one. Thus, we propose to encode the shareable information exists among similar classes, and utilize the training data from these classes to jointly learn the sharable filters.

To reach this goal, we introduce a sparse binary vector $\alpha^c \in \mathbb{R}^D$ for each class $c \in \{1, \dots, C\}$ (C is the number of classes) to indicate the selection status of rows of W . For example, if $\alpha_d^c = 1$, $d = 1, \dots, D$, then the d -th row of W is activated. For each class, only a small number of filters can be activated, while the same filter can be potentially activated by several classes. The cost of our shareable constraint term of class c is formulated as following:

$$\begin{aligned}
 L_{\text{sha}}^c &= \sum_{j=1}^{N_c} \mathcal{L}_{\text{sha}}^c(x_j^c, W^c) + \lambda_2 \|\alpha^c\|_0 \\
 \text{s.t. } \alpha_d^c &\in \{0, 1\}, d = 1, \dots, D \quad W^c = \text{diag}(\alpha^c) W \\
 \text{where } \mathcal{L}_{\text{sha}}^c(x_j^c, W^c) &= \left\| x_j^c - (W^c)^T W^c x_j^c \right\|_2^2
 \end{aligned} \tag{4.2}$$

where W^c is the selected filters of class c , and N_c is the number of training patches from class c . Similar to $\mathcal{L}_u$, $\mathcal{L}_{\text{sha}}^c$ is the reconstruction cost function with respect to W^c and training patch x_j^c from class c . The $\|\alpha^c\|_0$ term is used to force the sparsity, consequently, only a small number of rows of W will be activated. Here

¹According to our experiment results, adding the spatial denoising term will not bring much performance gain, but it can help the learning result be more robust to different initialization.

Algorithm 4.1: Deep Discriminative and Shareable Feature Learning

Input:

x_i : Unlabeled training patch

x_j^c : Image-level labeled training patch from class c

D : Number of filters in the global filter bank

L : Number of feature transformation layers

C : Number of the classes

$\xi, \gamma, \eta, \lambda_1, \lambda_2$: Trade off parameters for controlling weight of spatial denoising term, shareable term, discriminative term, and sparsity

Output:

$W^{(l)}$: Global filter bank (feature transformation matrix) of layer l

```

1. Initialize  $\alpha^c = \mathbf{0}^T$ 
2. Set  $W^{(l)}$  as a random number  $D \times D_0$  matrix
3. Learn  $W^{(l)}$  with  $L_u$  as the initialized  $W^{(l)}$  to the DDSFL
4. Select training exemplars for each class (Section 4.4)
5. Search the positive and negative nearest neighbor sets for each  $x_j^c$ 
for  $l = 1$  to  $L$  do
    while  $W^{(l)}$  and  $\alpha^c$  not converge do
        for  $c = 1$  to  $C$  do
            6. Fix  $W^{(l)}$  and all the  $\alpha^{\bar{c}}$ , solve Equation 4.6 by updating  $\alpha^c$ 
        end
        7. Fix  $\alpha^c, c = 1, \dots, C$  and solve Equation 4.5 by updating  $W^{(l)}$ 
    end
    return  $W^{(l)}$ 
end

```

we apply a greedy method to search for appropriate filters to activate. When α^c is updated in each iteration, the corresponding training data for each filter will also be updated.

4.3.2.3 Discriminative Information Encoding Term

To enhance the discriminative power of the features, similar to Section 3.3.1, we introduce a term based on the assumption that discriminative features should be close to the features from the same category, and be far away from the features from the different categories. However, patches from the same image class are inherently

diverse. If we get a triplet $\{x_i, x_i^+, x_i^-\}$, in which, x_i denotes the training patch, x_i^+ denotes a positive patch randomly select from the same class, and x_i^- represents a negative patch from the other classes, it is highly possible that x_i is similar to x_i^- rather than x_i^+ . To solve this problem, we adopt the nearest neighbor based “Patch-to-Class” distance (P2CD) metric [58, 67, 89] (refer to Figure 3.1). For a training patch x_j^c , its positive nearest neighbor patch set from the same category is denoted as $\Gamma(x_j^c)$; and its negative nearest neighbor patch set from the categories other than c is denoted as $\bar{\Gamma}(x_j^c)$. The k -th nearest neighbor in the two sets are represented as $\Gamma_k(x_j^c)$ and $\bar{\Gamma}_k(x_j^c)$ respectively.

In the class-specific feature space of class c (transformed by W^c), the feature representation of the k -th positive and negative nearest neighbor patches are denoted as $\Gamma_k(f_j^c) = \mathcal{F}(W^c \Gamma_k(x_j^c))$ and $\bar{\Gamma}_k(f_j^c) = \mathcal{F}(W^c \bar{\Gamma}_k(x_j^c))$ correspondingly. Then we get triplet $\{f_j^c, \Gamma_k(f_j^c), \bar{\Gamma}_k(f_j^c)\}$, and we aim to minimize the distance between f_j^c and $\Gamma_k(f_j^c)$, while maximize the distance between f_j^c and $\bar{\Gamma}_k(f_j^c)$. Furthermore, to make the learning procedure more efficient, we should focus on the “hard” training samples (based on the maximum margin theory in learning). Hence, we develop a “hinge-loss” like objective function:

$$\begin{aligned}
 L_{\text{dis}}^c &= \sum_{j=1}^{N_c} \max(\delta + \text{Dis}(x_j^c, \Gamma(x_j^c)) - \text{Dis}(x_j^c, \bar{\Gamma}(x_j^c)), 0) \\
 \text{where } \text{Dis}(x_j^c, \Gamma(x_j^c)) &= \frac{1}{K} \sum_{k=1}^K \|f_j^c - \Gamma_k(f_j^c)\|_2^2 \\
 \text{Dis}(x_j^c, \bar{\Gamma}(x_j^c)) &= \frac{1}{K} \sum_{k=1}^K \|f_j^c - \bar{\Gamma}_k(f_j^c)\|_2^2
 \end{aligned} \tag{4.3}$$

where K is the number of nearest neighbors in the nearest neighbor patch sets, we fixed it as 5, and δ is the margin, we set it to 1 in our experiments.

Furthermore, the data in the triplet $\{x_j^c, \Gamma_k(x_j^c), \bar{\Gamma}_k(x_j^c)\}$ depends on the values of W^c , which is updating during each iteration. Consequently, when W^c is iteratively being optimized and updated, the nearest neighbors will also be different. Since the nearest neighbor searching is time consuming, we assume the nearest neighbor members $\Gamma_k(x_j^c)$ and $\bar{\Gamma}_k(x_j^c)$ (the original image patches correspond to the triplet in the feature space) will keep unchanged when W^c does not change too much, and we only update the nearest neighbor members every 50 iterations of W^c updating.

Mention that this discriminative term is an enhanced version of the one introduced in Chapter 3 (where K is simply set as 1, and the “high-loss” is not employed). The L_{dis}^c here is able to encode the discriminative information in a more efficient and effective way.

4.3.3 DDSFL Objective Function and Optimization

Combing the above three learning components, the complete DDSFL objective function can be expressed as:

$$\begin{aligned}
 & \min_{W, \alpha^c} L_u + \gamma \sum_{c=1}^C L_{\text{sha}}^c + \eta \sum_{c=1}^C L_{\text{dis}}^c \\
 \text{where } L_u &= \sum_{i=1}^N \mathcal{L}_r(x_i, W) + \xi \sum_{i=1}^N \mathcal{L}_{sp}(x_i, \Omega(x_i)) + \lambda_1 \sum_{i=1}^N \|f_i\|_1 \\
 L_{\text{sha}}^c &= \sum_{j=1}^{N_c} \mathcal{L}_{\text{sha}}^c(x_j^c, W^c) + \lambda_2 \|\alpha^c\|_0 \\
 L_{\text{dis}}^c &= \sum_{j=1}^{N_c} \max(\delta + \text{Dis}(x_j^c, \Gamma(x_j^c)) - \text{Dis}(x_j^c, \bar{\Gamma}(x_j^c)), 0) \\
 \text{s.t. } & \alpha_d^c \in \{0, 1\}, d = 1, \dots, D \quad W^c = \text{diag}(\alpha^c) W
 \end{aligned} \tag{4.4}$$

In Equation 4.4, we need to simultaneously optimize the global filter transformation matrix W and the class specific filter selection vector α^c . This function cannot be jointly optimized with respect to W and α^c . However, when the α^c is fixed, the objective function turns out to be convex. When W is fixed, the objective function can be conveniently solved with greedy algorithm. Therefore, we adopt an alternating optimization strategy to iteratively update W and α^c .

We firstly fix α^c for each class, and focus on updating W :

$$\begin{aligned} \min_W \sum_{i=1}^N \mathcal{L}_r(x_i, W) + \xi \sum_{i=1}^N \mathcal{L}_{sp}(x_i, \Omega(x_i)) + \lambda_1 \sum_{i=1}^N \|f_i\|_1 \\ + \gamma \sum_{c=1}^C \sum_{j=1}^{N_c} \mathcal{L}_{\text{sha}}^c(x_j^c, W^c) + \eta \sum_{c=1}^C L_{\text{dis}}^c \end{aligned} \quad (4.5)$$

Because of the non-negative constraint applied to f_i (Section 4.3.2.1), $\|f_i\|_1$ degenerates to summation of different elements in f_i . Consequently, Equation 4.5 can be easily optimized by unconstrained solvers like L-BFGS.

Next we fix the W , and focus on updating α^c class by class:

$$\min_{\alpha^c} \sum_{j=1}^{N_c} \mathcal{L}_{\text{sha}}^c(x_j^c, W^c) + \lambda_2 \|\alpha^c\|_0 + \eta L_{\text{dis}}^c \quad (4.6)$$

We update one α^c each time for the c -th class, and keep $\alpha^{\bar{c}}$ ($\bar{c} \neq c$) unchanged. To get α^c , we apply a greedy optimization method. We first set all the elements in α^c as 0, then we search for the single best filter that can minimize Equation 4.6, and activate that filter by setting the corresponding element in α^c to 1. Afterwards, based on the previously activated filters, we search for next filter that can further minimize the cost function. The optimization terminates when the loss $\mathcal{L}_{\text{sha}}^c$ is smaller than a threshold, and the renewed α^c will be sent as the input to Equation

4.5 again to further optimize W .

The learning algorithm and initialization procedure are shown in Algorithm 4.1. The alternative optimization terminated until the values of both W and α^c converge, which takes about 5 rounds, and each round is considered as converged once the maximum number of iterations is achieved or the loss is under certain threshold.

4.4 Exemplar Selection

The quality of the learned features not only depends on the learning structure and parameters, but also relies on the quality of input training data. In order to select training patch set that carries both potential shareable and discriminative patterns, while also excludes common noisy patches, a training data selection procedure is required before processing feature learning.

For this purpose, one intuitive solution is applying k-means clustering, and using the cluster centroids as the exemplars [89] for training. However, as analyzed in [58], informative descriptors have low database frequency, and the conventional clustering methods prefer dominant patterns as inlier of clusters. Consequently, the non-informative patches (e.g. sky areas) are more likely to be selected rather than the discriminative ones. Thus, we propose two discriminative exemplar training data selection approaches: Nearest Neighbor (NN) based and Support Vector Machines (SVM) based exemplar selection.

4.4.1 NN Based Exemplar Selection

Inspired by the image-level exemplar selection method in [84], we propose a NN based exemplar selection scheme to remove the patches that are likely to commonly

exist in many classes.

In [84], the exemplar images are defined as the minimum set of “dominating images” which can cover all the intra-class variations, meanwhile as distinctive as possible to the exemplars from other classes. Different from this scenario, our exemplars are local image patches/features. Furthermore, we aim to keep shareable patterns, while also select class-specific discriminative patterns. To make the processing more applicable, we manipulate the selection procedure by removing the uninformative common patterns existing in most of the classes.

Firstly, we define the “coverage set” of a patch x_i . Given X as the original global patch set, which contains patches densely extracted from all the training images. For each patch $x_i \in X$, we search its M_{nn} nearest neighbors from X , and define these M_{nn} patches as the “coverage set” of x_i .

Secondly, we define the “reaching number” and “reaching distance” of a patch x_j^c . If x_j^c is included in the “coverage set” of a patch $x_l^{\bar{c}}$ from classes other than c , it is “reached” once, and the times of being reached over the patch set $X^{\bar{c}}$ (from class $\bar{c}$) is the “reaching number” of x_j^c . Meanwhile, we also record the distances between x_j^c and the patches which cover x_j^c . The summation of these distances is the “reaching distance” of x_j^c .

Intuitively, the larger the “reaching number” or the smaller the “reaching distance”, the more common the x_j^c . However, if we only consider “reaching distance”, for the common patterns (e.g. sky area patches), there are numerous similar patches with small variation. Consequently, because of the limited size of “coverage set”, the chance that these common patches be “reached” by the other similar common patches are not always high, and a common patch might get small “reaching number”, even though its similar patterns widely exist. On the other hand, if we only consider “reaching distance”, for a target discriminative patch, it

Algorithm 4.2: NN based Exemplar Selection

Input:

X : Global patch set

X_c : Patch set of class c

C : Number of the classes

M_{nn} : Number of patches in each coverage set

ε_{nn} : Threshold for selecting discriminative exemplars

Output:

E_c : Exemplars of class c

1. Calculate the coverage set of each patch from X

for $c = 1$ to C **do**

 2. For each patch from X_c , calculate its reaching score $R_S(x_j^c)$ based on Equation 4.8

 3. Descendingly sort the patches from X_c based on the reaching scores.

 4. Select the top ε_{nn} percent ranked patches as the exemplars E_c

end

return E_c

might be only covered once, while the corresponding patches are highly similar to it, then its “reaching distance” is unfairly small. Therefore, we make use of both “reaching number” and “reaching distance”, and define the “reaching score” of x_j^c to class $\bar{c}$ as following:

$$R_S^{\bar{c}}(x_j^c, \bar{c}) = \frac{1}{R_N(\bar{c})} \sum_{l=1}^{R_N(\bar{c})} R_d \quad (4.7)$$

$$\text{where } R_d = \|x_j^c - x_l^{\bar{c}}\|_2$$

where $R_S^{\bar{c}}$ is the class-wise reaching score, $R_N(\bar{c})$ is “reaching number” denotes the number of the patches from classes $\bar{c}$ whose coverage sets contain x_j^c , and summation of R_d is the “reaching distance”.

Furthermore, to keep the shareable patterns, we also prefer to penalize the patterns appearing in most of the classes rather than shared among few classes.

Thus, we define the final “reaching score” as:

$$R_S(x_j^c) = \frac{1}{C-1} \sum_{\bar{c} \neq c} R_S^{\bar{c}}(x_j^c, \bar{c}) \quad (4.8)$$

where C is the number of classes. If x_j^c has low “reaching score”, then it means that x_j^c represents a common pattern, and should be removed in the training procedure, otherwise, x_j^c should be kept an exemplar.

For each class, we sort the patches based on their $R_S(x_j^c)$ scores descendingly, and select the top ε_{nn} percent of them as discriminative exemplars. The selecting procedures are shown in Algorithm 4.2.

4.4.2 SVM Based Exemplar Selection

Inspired by the mid-level discriminative exemplar searching methods [81, 90], we also try to use learning based methods to search for local patch level exemplars. Compare to the NN based selection method, SVM based method is much faster and cost less memory with small performance drop.

The desired exemplars should be both representative and discriminative: similar patches of the exemplars should be able to be found from the same or similar classes, and the exemplars of a certain class should also contain class-specific patterns. To reach these goals, we model the problem as a discriminative clustering problem, and the centers of the effective clusters are considered as the exemplars. In this chapter, we use linear SVM to iteratively refine the selection planes for each cluster. In one iteration of each cluster, there are mainly two steps: SVM training and SVM testing (re-clustering). In the SVM training step, the patches from the same cluster are positive samples, and the patches from all the other classes are negative samples, and a linear SVM model is learned based on them. In the SVM

Algorithm 4.3: SVM based Exemplar Selection

Input:

X^{tr} : Training patch set

X_{cs}^{tr} : The s -th cluster of the training patch set from class c

$X_{\bar{c}}^{tr}$: Training patches from classes other than c

X^{val} : Validation patch set

S_c : Number of initial clusters for each class c

C : Number of the classes

M_{svm} : Maximum number of patches in each cluster

Output:

E_c : Exemplars of class c

1. Applying k-means to initialize S_c clusters for each class

while not converged **do**

for $c = 1$ to C **do**

for $s = 1$ to S_c **do**

 2. For each cluster s from class c , set the cluster members X_{cs}^{tr} as positive data and randomly sample patches from $X_{\bar{c}}^{tr}$ as negative data, and train a linear SVM

 3. Test the learned SVM on the validation set X^{val} . If the SVM meets any of the non-discriminative condition, remove the cluster.

end

end

 4. Swap training set X^{tr} and validation set X^{val} .

end

5. Use the remaining qualified cluster centers as the exemplars E_c

return E_c

testing step, the previous learned SVM model is used to test on all the validation data patches (not visible in the training procedure), the patches which get fired² are kept and be used to replace the data in the original cluster, and they will be used to learn the new SVM model in the next iteration. To ensure the exemplars are representative to the class, we do not consider clusters with less than ε_{svm} ($\varepsilon_{svm} = 3$) members. To ensure the purity of the clusters, we constrain that each effective cluster can have at most M_{svm} (we set M as 10) members.

We randomly pick a large subset of local patches from the training images as

exemplar candidates, and run k-means clustering for all of the candidate patches of each class to initialize the clusters. Since the SVM heavily prefers the training data during testing, thus we equally separate the candidate patch set into training set and validation set. For the learned SVMs, we test them on the validation set class by class. If either of the following two conditions is true, then the cluster is considered as not discriminative, and should be removed:

- The SVM fires² less than ε_{svm} (3 in our experiment) patches from the same class (considered as outliers or noisy data);
- The SVM fires² many patches from more than half of all the classes (considered as common patterns exist everywhere).

Afterwards, we swap validation set and training set, and repeat the above procedures until convergence. The iteration usually terminates in about 5 rounds. The overall procedure is shown in Algorithm 4.3.

4.4.3 NN Based vs SVM Based Exemplar Selection

We propose NN based and SVM based exemplar selection methods. According to the experiment results (Table 4.4), both of them bring significant improvements than DDSFL without exemplar selection procedure. Here we make a brief comparison between these two methods in three aspects: performance, computation cost, and memory cost.

In classification performance aspect, comparing with the result of randomly select training patch, NN based exemplar selection leads to around 1% to 3% improvements, while SVM based selection gets around 0.5% to 1.5% improvements. Thus, NN based method is a better choice for higher accuracy results.

²classification score higher than -1.

Next we compare the two methods in computation cost aspect. For the NN based selection, the most time consuming procedure is the step 1 in Algorithm 4.2: coverage set searching. To speed up the procedure, we use the FLANN [70] toolbox, which is a library utilize multiple randomized k-d trees to achieve fast NN approximation for high dimensional data. For the SVM based selection, step 2 and 3 in Algorithm 4.3 (SVM training and testing) are the most and second most time consuming steps. To speed up the SVM training procedure, we apply stochastic learning of SVM [91]. While in the SVM testing step, for each trained linear SVM model, only classification score of the validation data patches need to be calculated. This procedure can be easily parallel computed, and we use GPU to speed up. In our experiments, the SVM based methods cost much less hours, especially when the number of training patches is very large.

At last, the memory costs are compared. In the NN based selection, we need to keep all the training patches X in the memory to apply coverage set searching for each patch. To depress the memory cost, we separate the X into several subgroups, process NN searching for each subgroup, and resort the searching results from all the subgroups afterwards. However, the more subgroups we generate, the longer overall NN searching time will be cost. On the contrary, SVM based selection requires very limited memory, and only a ignorable number of data points need to be loaded into the memory.

Therefore, NN based method can be used to achieve higher accuracy, while SVM based method can be manipulated faster with much less memory cost with less performance gain.

4.5 Experiments and Analysis

4.5.1 Datasets and Experiment Settings

We tested DDSFL on three widely used scene image classification datasets: Scene 15 [51], UIUC Sports [92], and MIT Indoor [52]. We also tested on the challenging PASCAL VOC 2012 object classification dataset. To make a fair comparisons with other types of features, we only utilized gray scale information for all of the images in these datasets.

- **Scene 15:** This dataset includes 4,485 images from 15 outdoor and indoor scene categories, each category contains 200 to 400 gray scale images. Based on the standard setting, we use 100 images per category for training, and the rest images for testing.
- **UIUC Sports:** This dataset contains 1,579 from 8 sports event classes, each category contains 137 to 250 images. We use 70 images per class as training images, and 60 images per class as testing images.
- **MIT Indoor:** This dataset have 15,620 image in total, and they are from 67 indoor scene categories. Based on the splitting set in [52], there are around 80 training images and 20 testing images for each category.
- **PASCAL VOC 2012:** This challenge aims to recognize objects from a number of visual object classes in realistic scenes. In this dataset, 20 object classes are selected. There are 5,717 training images with 13,609 objects, 5,823 validation images with 13,841 objects. Thus, the total number of labeled images are 11,540, and we did not use any extra training data. In the testing procedure, 10,991 unlabeled testing images are provided, and the

classification results can be achieved by uploading the classification scores to the PASCAL VOC evaluation server.

Since the resolution of the original images in UIUC Sports, MIT Indoor and PASCAL VOC 2012 are too high for learning local features efficiently, we resize them to have maximum 300 pixels along the longer axis. For Scene 15 and UIUC sports, we randomly split the training and testing datasets for 5 times, and get the results of different methods based on the same splits. The average accuracy numbers over these 5 rounds are reported for comparison.

For all the local features, we densely extracted features from S scales (S equals to 6, 5, and 3 for the first, second, and third layer respectively) with rescaling factors $2^{-i/2}, i = 0, 1, \dots, S$. Specifically, Random³ feature, RICA [23] feature and DDSFL features were extracted with step size 3 for the first layer, and step size 6 for the second and the third layer. SIFT features [1] were extracted from 16x16 patches, the step size was 3. HOG2x2 features [56] were extracted based on cells of size 8x8 with stride of 1 cell. LBP features [10] were extracted from cells of size 8x8.

We randomly picked 4,000 raw pixel value image patches (2,000 for MIT Indoor and PASCAL VOC 2012) from each training image with different scales, and select 10% of them as training exemplars (refer to Section 4.4). The margin of the hinge loss function in the objective function Equation 4.4 was fixed as 1. In each DDSFL layer, the parameters ξ , λ_1 , λ_2 , γ and η were initialized as zeros, and they were sequentially learned one after another by applying cross validation (e.g., learn the best ξ first, and then fix the value of ξ and search for the best value of λ_1). The maximum number of iterations of updating W and α_c in Algorithm 4.1 was 5.

³DDSFL feature extraction scheme without training procedure, the filter bank W for each layer is randomly generated.

Methods	Scene 15	UIUC Sports	MIT Indoor
GIST[11]	73.28%	-	22.00%
CENTRIST [93]	83.10%	78.50%	36.90%
SIFT[1]	82.06%	85.12%	45.86%
HOG2x2[56]	81.58%	83.96%	43.76%
LBP[10]	82.95%	80.04%	39.25%
OTC[94]	84.37%	-	47.33%
Random ³	70.91%	67.92%	21.34%
RICA[23]	80.01%	82.45%	47.76%
DDSFL	84.42%	86.91%	52.26%
Caffe[75]	87.99%	93.96%	58.52%
SIFT+ Caffe	89.90%	95.05%	70.51%
DDSFL + Caffe	92.81%	96.78%	76.23%

Table 4.1: Comparison results between DDSFL feature and other features. (Caffe is the global feature learned by the deep Alex-net pre-trained on ImageNet)

To convert the local features to global image representation, we utilized the LLC [53] and SPM [51], the combination of which can generally lead to good classification results. LLC used locality-constrained linear coding to encode local features, and performed max-pooling and linear-SVM afterwards (refer to Figure 4.3). While SPM utilized rough spatial structure information, and concatenate all the pooled features from different spatial pooling regions. For all the local features, the size of the codebook was fixed as 2,000, and each image was divided into 1x1, 2x2, and 4x4 spatial pooling regions. We’ve also tested on other coding schemes (e.g. vector quantization) and pooling strategies (e.g. average pooling), DDSFL consistently outperform traditional local features.

In all the experiments of DDSFL, if not specified, we report the results of three layer learning framework with NN based exemplar selection method.

	aero	bike	bird	boat	btl	bus	car	cat	chair	cow	
SIFT[1]	77.8	44.4	42.4	54.0	16.0	74.7	52.1	53.0	45.6	21.4	
HOG2x2[56]	78.3	44.8	38.9	51.6	16.6	77.0	51.2	57.3	49.2	23.5	
LBP[10]	78.7	40.8	36.6	53.5	16.2	75.8	46.2	55.6	45.4	20.8	
Random ³	63.0	15.4	18.8	26.7	10.4	44.6	30.2	28.7	26.3	9.4	
RICA	76.1	37.2	39.1	49.6	13.8	70.5	46.2	51.2	41.4	15.6	
DDSFL	77.7	42.5	45.4	53.3	24.0	72.2	50.6	54.2	47.4	26.0	
Caffe[75]	90.7	67.9	79.9	77.0	32.7	86.0	59.7	81.6	51.4	56.1	
SIFT+Caffe	92.3	72.3	83.0	79.5	38.1	88.5	65.8	84.9	59.1	61.5	
DDSFL+Caffe	92.7	75.4	85.1	79.4	42.3	88.1	68.5	87.1	62.9	65.8	
	table	dog	horse	moto	pers	plant	sheep	sofa	train	tv	mAP
SIFT[1]	40.2	38.4	40.7	54.2	71.3	13.1	27.4	33.0	64.5	47.6	45.6
HOG2x2[56]	40.4	40.3	40.6	46.2	72.8	13.5	34.4	34.7	63.6	49.9	46.2
LBP[10]	38.4	39.6	37.4	43.4	69.8	12.8	37.8	29.3	59.6	42.1	44.0
Random ³	15.8	22.8	13.7	22.0	50.7	7.3	15.1	15.8	30.0	22.9	24.5
RICA[23]	29.8	35.6	33.3	46.6	67.0	14.1	28.9	29.6	58.9	44.3	41.4
DDSFL	36.6	42.6	38.9	50.7	70.3	23.4	36.2	36.4	61.6	49.0	46.9
Caffe[75]	55.2	72.0	69.5	75.1	88.0	36.3	63.2	43.0	87.7	64.6	66.9
SIFT+Caffe	61.6	76.0	72.7	79.0	90.7	41.8	69.2	52.7	88.9	70.1	71.4
DDSFL+Caffe	60.6	80.2	74.0	76.4	92.3	43.6	74.3	53.2	88.4	73.7	73.2

Table 4.2: Class-wise classification comparison results between DDSFL and other features on PASCAL VOC 2012.

4.5.2 Comparison with Other Features

We firstly compare DDSFL feature with the popular features which have shown good performance on scene images classification. There are three types of features: 1) Hand-crafted features, which include SIFT[1], HoG2x2[2, 56, 64], LBP[10], GIST[11], and CENTRIST[93]; 2) The baseline features learned in unsupervised way without encoding discriminative or shareable information, which include feature extracted with Random weights³, and the RICA [23] feature; 3) Caffe[75] feature.

As shown in Table 4.1 and Table 4.2, compared with the hand-crafted features and baseline unsupervised learned features, DDSFL consistently and significantly outperforms all of them. By comparing with the baseline Random³ features, the significant performance improvements demonstrate that the learning procedures

can help to capture data adaptive information, and help to do better recognition. While by comparing with RICA features, the accuracy increasing indicates the effectiveness of learning discriminative and shareable information from image level labels.

4.5.2.1 Complementary Effect with Caffe

The Caffe feature [75] shown in Table 4.1 is a global feature extracted from the Alex-net [5], which is pre-trained on ImageNet [9]. Alex-net has 7 layers (5 convolution layers + 2 fully connected layers). When applying the pre-trained model to extract features in datasets other than ImageNet, the 6-th layer feature generally lead to better results than the 7-th layer feature⁴ [75, 95]. Therefore, we used the 6-th layer Caffe feature for evaluation. Although the Caffe feature is very powerful, it is not fair to directly compare the performance of Caffe and the other features. Firstly, DDSFL did not utilize the huge amount of training data in ImageNet. Secondly, the color information has not been utilized⁵. Thirdly, DDSFL focuses on learning powerful local features rather than global feature representation, and we supposed these two frameworks should encode complementary information, which was proved to be correct in the last row of Table 4.1 and Table 4.2.

To reveal the complementary effect intuitively, we tested on MIT 67 Indoor and compare the class-wise performance. In total, there are 14 categories that DDSFL performed better than Caffe, and 53 categories that Caffe did better than

⁴We also tested with the 7-th layer features. We got 87.35%, 93.44%, and 58.27% on the three scene datasets, and got mean accuracy of 65.79% on PASCAL VOC 2012.

⁵We did a small experiment with single layer DDSFL on UIUC sports, the inputs were $L \times a \times b$ values image patches, around 1% performance gain can be achieved. However, this performance improvement was achieved with the price of three times of computation power, so we did not test on all the datasets.

auditorium
		DSFL: 66.7%	Caffe: 38.89%	
corridor
		DSFL: 57.14%	Caffe: 47.62%	
bowling
		DSFL: 75.00%	Caffe: 100.00%	
winecellar
		DSFL: 23.81%	Caffe: 76.19%	

Figure 4.4: Comparison results of DDSFL vs Caffe on MIT Indoor. The first two rows show the two categories on which DDSFL works better than Caffe, the last two rows show the classes that are better represented by Caffe. DDSFL and Caffe are complimentary.

DDSL. In Figure 4.4, the first two rows show the categories that DDSFL worked better than Caffe, and we show the testing images which were correctly classified by DDSFL, but wrongly classified by Caffe. The last two rows show the categories which Caffe outperformed DDSFL, and we show the testing images which DDSFL failed to recognize but Caffe could. To quantitatively analyze the complementation effect, we simply concatenate DDSFL feature (after processing LLC and SPM) with the Caffe feature to check the performance. As shown in the last row of Table 4.1, we were able to get a huge performance improvement than purely using the Caffe features and produced the state-of-the-art results. Alex-net apply backpropagation

to transmit the supervised information to all the deep layers, this is very helpful for the relatively high layer parameter updating, while it is extremely weak in training bottom layer parameters. In contrast, we explicitly use supervised information to train each layer. Moreover, DDSFL is more suitable for relatively small datasets, as evidenced by the experimental results, while previous attempts on directly training a CNN on small datasets usually failed. Thus, these two features are expected to be complimentary.

To make a fair comparison and exclude the influence brought by coding and pooling schemes, we also tested the combination of SIFT and Caffe feature. Although this combination also lead to satisfactory results, there is an obvious gap between the performance of SIFT+Caffe and DDSFL+Caffe. This observation indicates that DDSFL can learn more effective complementary information by encoding data adaptive information. This might because of hand-crafted features such as SIFT usually extracting “garbor-like” features, which are highly similar to the lower level filters learned by Alex-net and not as effectively complementary as DDSFL.

4.5.3 Comparison with Other Methods on Scene Classification

There are numerous methods applied to the scene image classification, most of them simply utilized the existing hand-crafted features, and focus on the dictionary learning, coding scheme, or mid-level elements mining areas. Most of these works do not conflict with our feature learning framework, DDSFL feature can easily replace the hand-crafted features, and combine with these methods to achieve better performance. (e.g. LScSPM [101] and IFV [83] focused on coding, which

Methods	Scene 15	UIUC Sports	MIT Indoor
ROI + GIST[52]	-	-	26.50%
DPM [96]	-	-	30.40%
Object Bank [97]	80.90%	76.30%	37.60%
Discriminative Patches[90]	-	-	38.10%
LDC [89]	80.30%	-	43.53%
Macrofeatures [98]	84.30%	-	-
Visual Concepts [80]	83.40%	84.80%	46.40%
SR-LSR [99]	85.70%	83.90%	-
MMDL [100]	86.35%	88.47%	50.15%
Discriminative Part Detector[82]	86.00%	86.40%	51.40%
LScSPM [101]	89.78%	85.27%	-
IFV [83]	-	-	60.77%
MLrep + IFV [81]	-	-	66.87%
ISPR + IFV [102]	91.06%	92.08%	68.50%
MOP-CNN (Caffe) [103]	-	-	68.88%
DDSFL + Caffe	92.81%	96.78%	76.23%

Table 4.3: Comparison results of DDSFL and other popular methods on Scene 15, UIUC sports, and MIT Indoor.

can be used to encode DDSFL features.)

As shown in Table 4.3, comparing with the other methods, DDSFL+Caffe feature achieved the highest accuracy results on all of the three scene classification datasets. Note that Visual Elements [81] utilized numerous patches extracted at scales ranging from 80x80 to the full image size, and the patches were represented by standard HOG [2] plus a 8x8 color image in L*a*b space, and very high dimensional IFV[83] features. While MMDL [100] combined 5 types of features on 3 scales. MOP-CNN (Caffe) [103]) enhanced the original Caffe with 10% by considering geometric invariance, which can also be used in DDSFL to further improve the performance.

Figure 4.5: Visualization of the filters learned by RICA and DDSFL on the UIUC Sports.

4.5.4 Analysis of the Effects of Different Components

In this section, we aim to compare our shareable and discriminative learning method to the baseline without encoding such information, which is equivalent to the modified RICA method in [23], and the baseline of using randomly generated feature transformation matrix without learning procedure at all. In the rest of this subsection, we report the results of recruiting different components of the 2 layer DDSFL, which can be efficiently tested and already contain the hierarchical information.

We first show the visualization of the first layer filters learned from UIUC Sports in Figure 4.5a and Figure 4.5b. We can see that our DDSFL is able to capture more sharply localized patterns, corresponding to more class-specific visual information.

4.5.4.1 Effect of Learning Shareable Filters

We tested the DDSFL with or without the feature sharing terms, and got the intermediate results in Table 4.4. The first row of the table shows the result of

plainly using randomly generated filters without learning, the poor results indicate that the representation power of the network structure itself is severely limited. The second row shows the modified unsupervised RICA features learned by solving Equation 4.1. In the third row, $L_u + L_{\text{sha}}$ corresponds to features learned with Equation 4.2. The improvement in accuracy shows that learning shareable features is effective for classification. However, if we removed the global reconstruction error term $\mathcal{L}_u$ and only kept the shareable terms, as shown in the fourth row, the performance dramatically dropped to a random filter bank level.

4.5.4.2 Effect of Discriminative Regularization and Exemplar Selection

When only integrating the discriminative term (without exemplar selection) with the unsupervised term, it is equal to an enhanced version of the method described in Chapter 3. Comparing the fifth row of Table 4.4 with the second row, the effectiveness of the discriminative term is obvious, which is consistent with the conclusions in Chapter 3.

Comparing the sixth to eighth rows of Table 4.4, we can observe that without applying exemplars selection step, we could not achieve much improvement because noisy training examples might overwhelm the useful discriminative patterns. However, once using the selected exemplars, significant improvement can be observed. This shows that discriminative exemplar selection is critical in our learning framework. Furthermore, as shown on the last two rows of Table 4.4 comparing the NN based and SVM based exemplar selection methods, the former method usually got better performance with higher computational and memory cost.

Furthermore, it's obvious that only using 10% of the whole patch set dramatically increased the efficiency of nearest neighbor search in the training step. Thus,

Methods	Scene 15	UIUC Sports	MIT Indoor
Random ³	73.52%	70.42%	23.28%
L_u (modified RICA[23])	79.85%	82.14%	47.89%
$L_u + L_{\text{sha}}$	82.01%	83.67%	49.70%
L_{sha}	72.69%	72.52%	24.12%
$L_u + L_{\text{dis}}$ (without Exemplar)	82.44%	82.89%	50.75%
$L_u + L_{\text{sha}} + L_{\text{dis}}$ (without Exemplar)	82.50%	83.43%	51.28%
$L_u + L_{\text{sha}} + L_{\text{dis}}$ (SVM Exemplar)	83.02%	84.98%	51.65%
$L_u + L_{\text{sha}} + L_{\text{dis}}$ (NN Exemplar)	84.19%	86.45%	52.24%

Table 4.4: Analysis of the effect of each component in DDSFL (two layers).

our exemplar selection method is both effective and efficient.

4.5.4.3 Effect of Hierarchical Structure

According to the comparison results shown on Table 4.5, we can observe three interesting phenomena of using different number of layers. Firstly, no matter using how many layers, learning data adaptive features can always achieve more powerful filter banks and lead to better recognition results, and considering shareable and discriminative information can consistently achieve better result than unsupervised learning. Secondly, the good performance of higher layer features heavily depends on the high quality lower level features, by using the Random filters, the more layers be processed, the worse results will be achieved. Thirdly, in the RICA and DDSFL cases, we observe that the performance gain of using two layer structure is much more obvious than adding the third layer, which might because of lacking of training samples to support lager learning framework (especially when the scale is very small, one image might contain only one 3rd layer patch).

Methods	Scene 15	UIUC Sports	MIT Indoor
Random ³ (1 layer)	74.12%	71.46%	23.53%
Random ³ (2 layers)	73.52%	70.42%	23.28%
Random ³ (3 layers)	70.91%	67.92%	21.34%
RICA[23] (1 layer)	77.54%	79.03%	41.36%
RICA[23] (2 layers)	79.85%	82.14%	47.89%
RICA[23] (3 layers)	80.01%	82.45%	47.76%
DDSFL (1 layer)	82.61%	83.92%	47.16%
DDSFL (2 layers)	84.19%	86.45%	52.24%
DDSFL (3 layers)	84.42%	86.91%	52.26%

Table 4.5: Analysis of the effect of number of layers.

4.5.4.4 Effect of The Size of Filter Bank

To further analyze the influence caused by the size of filters, we tested on Scene 15 dataset with 128, 256, 512, 1024, and 2048 filters by using different number of learning layers. The results are shown in Figure 4.6. The accuracy results consistently increase when the number of filters increase. When the size of the filter bank is small, especially when the feature transformation matrix is under-complete, the learned features are relatively weak. When the number of filters increases, and W becomes over-complete, the performance is substantially improved. Thus, learning over-complete filter banks do help to obtain better feature representation because the resulting filter banks captures more information. However, when the number of filters further increases, the performance does not increase much, while the learning process will be extremely slow. In our experiment, we learned 400 filters for all the layers as a compromise between efficiency and accuracy. Furthermore, with different number of filters, increasing the number of DDSFL layers also consistently increase the performance, which indicates that higher level DDSFL features can learn complementary patterns.

Figure 4.6: Results of varying number of DDSFL filters and layers on Scene 15.

4.6 Conclusion

In this chapter, we propose a hierarchical weakly supervised local feature learning method, called DDSFL, to learn discriminative and shareable filter banks to transform local image patches into different visual level features. In DDSFL method, we learn a flexible number of shared filters to represent shareable patterns exist among similar categories. To enhance the discriminative power, we force the features from the same class to be similar, while features from different classes to be separable. We tested DDSFL on three scene image classification benchmark datasets and PASCAL VOC 2012, the results consistently show that our learned features can outperform most of the existing features. By combining with the Caffe features pre-trained on ImageNet, we can greatly enhance the representation power, and achieve state-of-the-art classification results on all the three scene dataset, and also get promising results on PASCAL.

DDSFL is able to learn powerful image descriptions for normal scale (thousands/ten thousands of images) image categorization. However, DDSFL or any

traditional BoW based image classification frameworks can hardly be efficiently applied to large-scale image (millions of images from thousands of categories) datasets, where there are too many local patches to extract. In contrast, deep neural networks like convolutional neural networks (CNNs) and their extensions can process more efficient batch training on GPU, and solve large-scale recognition problems with very good performance. Thus, in Chapter 5, we will explore new deep feature learning algorithms based on CNN structures.

Chapter 5

Convolutional and Hierarchical Recurrent Neural Network

Most research efforts on image categorization were focused on small-scale or median-scale datasets with dozens of classes due to the limitation of labelled data. However, large-scale image classification with hundreds to thousands of classes is an inevitable trend since the emergence of ILSVRC. Traditional hand-crafted feature based image classification framework, and our feature learning methods introduced in Chapter 3 or Chapter 4 cannot be effective to deal with large-scale image categorization. Thus, in this chapter, we turn to explore and improve the deep convolutional neural networks (CNNs), which are the current most successful and developed frameworks for large-scale image classification.

In this chapter, we integrate CNNs with our proposed hierarchical recurrent neural networks (HRNNs), and develop end-to-end convolutional and hierarchical recurrent neural networks (C-HRNNs). C-HRNNs not only make use of the representation power of CNNs, but also efficiently encodes spatial and scale dependencies among different image regions. Specifically, we propose two recurrent

neural network models to encode the spatial and scale dependencies: 1) hierarchical simple recurrent network (HSRN), which is fast and has low computational cost; and 2) hierarchical long-short term memory recurrent network (HLSTM), which performs better than HSRN with the price of more computational cost. On four of the most challenging large-scale object/scene image classification benchmarks, our C-HRNNs achieve state-of-the-art results on Places 205, SUN 397, MIT indoor, and competitive results on ILSVRC 2012.

5.1 Introduction

CNNs have shown their success in but not limited to the following areas: image classification [5, 24–28, 55, 103], detection [29–32, 104], face recognition [33, 34], etc. The key idea of CNNs is utilizing convolutional and pooling layers to progressively extract more and more abstract and complex patterns. The convolutional layers convolve multiple local filters with input images (or output response maps of the previous layer), and aim to produce translation invariant features at each location in the image. Afterwards, pooling layers are used to summarize the feature responses of the convolutional layers over multiple regions of the image, and compress the size of the response maps. Both convolution and pooling are locally performed on image regions. For example, the representation of the top left image region will not influence the representation of the bottom right region. However, contextual information is very important for object/scene recognition. For example, in an image with label “beach”, if “sand” regions are represented with the reference of “sea” regions, then it is much easier to distinguish them from “road” or “desert sand”. Thus, in CNNs, spatial and scale dependencies among different image regions are not explicitly modeled. To solve this problem, we aim to en-

Figure 5.1: The overall framework of C-HRNNs. **CNN layers:** Extract mid-level representations for image regions by processing five convolutional and pooling layers. **HRNN layers:** (a) Pool the output of the fifth CNN layer into multiple scales. (b) For each scale, spatial dependencies are captured by direct or indirect connections between each region and its surrounding neighbors. (c) For different scales, scale dependencies are encoded by transferring information from the higher-level scales to corresponding areas at the lower-level scales. Take the l -th ($l \in [1, \dots, L]$) scale as an example, information of the yellow block will be transferred to corresponding areas (highlighted by yellow) in the $l+1$ -th to the L -th scales as reference. While the red block in the $l+1$ -th scale will be transferred to its influenced areas (highlighted by red) in the $l+2$ -th to L -th scales. **FC layers:** Collect different scale HRNN outputs and connect to two fully connected layers. (Best viewed in color)

code contextual dependencies in image representation. To learn the dependencies efficiently and effectively, we propose a new class of hierarchical recurrent neural networks (HRNNs), and utilize the HRNNs to learn such contextual information.

RNNs (refer to Section 2.1.2) have achieved great success in natural language processing (NLP) [36–38, 44–47]. RNNs [35, 48] are neural networks developed for modeling dependencies in sequences by using feedback connections among themselves. Thus, they can retain all the processed states in the sequence, and learn patterns from sequential context. Furthermore, because of the reuse of hidden layers, only a limited number of neurons need to be kept in the model. Two

most popular RNN models are the simple recurrent neural network (SRN) and long-short term memory recurrent network (LSTM). Based on which, we will introduce hierarchical SRN (HSRN) and hierarchical LSTM (HLSTM). Both HSRN and HLSTM target on modeling the spatial and scale dependencies among different local image regions. However, they also have different characteristics: HSRN is simple and fast, while HLSTM is more complex but it is able to maintain the long-term dependencies among local image regions far away from each other, and lead to better performance than HSRN.

Our proposed hierarchical recurrent neural networks (HRNNs) model two types of contextual dependencies: *spatial dependencies* and *scale dependencies*.

Firstly, we consider the *spatial dependencies* among image regions from the same scale but at different locations. Since there are no off-the-shelf sequences in images, inspired by the multi-dimensional RNN [105], we generate two dimensional spatial region sequences for images, and represent each region as a function of its neighboring regions. Details will be described in Section 5.3.3.1.

Secondly, we build multiple scale RNNs, and consider *scale dependencies* among image regions from different scales but at the same locations. Information captured from different scales are complementary to each other. Connecting multiple scales can help to learn more robust representation. For example, in an image with label “car”, regions at a lower-level scale mostly contain patterns such as “tire” and “window”, while regions at a higher-level scale include global patterns such as “car”. Knowing the existence of “car” can help the system to increase the representation preference of “tire” in the corresponding local regions. Details will be described in Section 5.3.3.2.

However, HRNN layers are processed based on image regions, while in image classification, no intermediate labels for any of these regions are provided. The

only supervision is the image-level labels. To make use of it, fully connected layers are introduced to collect the outputs of HRNN layers, merge them through the global hidden layer, and finally connect to image-level labels with a softmax layer.

Integrating CNNs with our HRNNs, we propose end-to-end networks called convolutional and hierarchical recurrent neural networks (C-HRNNs). As shown in Figure 5.1, the C-HRNN framework can not only maintain the discriminative representation power of CNNs, but also efficiently encode the spatial and scale contextual dependencies with HRNNs. C-HRNNs have been tested on four most challenging large-scale object/scene image classification benchmarks, and it achieves the state-of-the-arts on Places 205, SUN 397, MIT indoor, and promising results on ILSVRC 2012.

5.2 Recurrent Neural Networks on Vision

Although very popular in NLP, RNNs have rarely been applied to computer vision area. In the recent related works in computer vision, there are mainly four branches of works which involve the recurrent idea.

In the first branch of works, recurrent layers are mainly used as “tied” layers in the “feed-forward” networks, which means different layers share the same parameters. Different from our recurrent networks, these “tied” layers iteratively encode the input data from the same locations with the same network parameters, and these layers focus on reducing the number of parameters, rather than modeling the contextual dependencies among input data from different locations. In [106], shared CNNs are applied to learn pixel label consistency among multi-scale image patches. In DrSAE [107], auto-encoder with rectified linear units are employed to iteratively encode the global digital-number image. In [49], the “tied” CNN (called

as recurrent convolutional network) is employed to assess the contributions of the number of layers, response maps, and parameters. Different from these works, the “recurrent” in our C-HRNNs means learning spatial and scale dependencies among different image regions, and expanding receptive fields of local regions by encoding contextual information.

In the second branch of works, RNNs are used to predict/generate the motion curve of objects/parts in the current/next moment, and applied to visual attention tasks. In [108, 109], RNN is used to build a sequential variational auto-encoder to iteratively analyze/generate image parts by parts (at each iteration, RNNs are used to selectively attends to parts of the image while ignoring others). Different from them, we target on building end-to-end networks for large-scale image classification.

In the third branch of works, RNNs [110, 111] are used to combine the video information over an ordered sequence of video frames for video recognition and description. Differently, our C-HRNNs model the contextual dependencies within single image rather than the sequential appearance/motion dependencies among consecutive frames.

In the fourth branch of works, RNNs are combined with CNNs for image/video description [39–41, 112]. In these works, CNNs are utilized to generate feature representations of image/video data, while RNNs are used to connect the image/video feature domain to the text feature domain, and RNNs mainly focus on modeling the text contextual dependencies in the sentences/paragraphs. Different from these works, our C-HRNNs models the contextual information in image appearance domain.

5.3 Convolutional and Hierarchical Recurrent Neural Networks

As shown in Figure 5.1, our proposed convolutional and hierarchical recurrent neural networks (C-HRNNs) consist of three types of layers: 1) five convolutional (and pooling) layers for extracting middle level image region features; 2) hierarchical recurrent layers for encoding spatial and scale dependencies among different image regions; 3) two fully connected layers for generating global image representation. Finally, an N-way (N indicates the number of categories) softmax loss layer is added on the top for classification.

5.3.1 Convolutional Layers

As shown in the left part of Figure 5.1, given input raw pixel images, firstly, five convolutional layers are processed to progressively extract more and more complex and abstract patterns. According to the analysis in [113], outputs of the fifth convolutional layer are able to capture patterns representing parts and objects. Furthermore, size of the fifth layer response maps is orders of magnitudes smaller than size of the original raw pixel images. Thus, based on such CNN features, our proposed HRNNs can model the contextual dependencies among middle-level regions with semantic meanings, and HRNNs can be processed very efficiently. Furthermore, with back propagation, RNNs can help the CNNs to increase the quality of middle-level and low-level features.

5.3.2 Review of General RNNs

RNNs [35, 48] are originally developed for modeling dependencies in time sequential data. In RNNs, two of the most typical models are the simple recurrent neural network (SRN), and the long-short term memory recurrent neural network (LSTM).

In the following two subsections, SRN and LSTM will be introduced to represent each state t of a given sequence of length T . $x^{(t)}$, $h^{(t)}$, and $y^{(t)}$ are the input, hidden and output representations of the t -th state respectively.

5.3.2.1 Simple Recurrent Neural Nets

As shown in Figure 5.2a, the t -th state in SRN can be represented as:

$$h^{(t)} = \psi_h (W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h) \quad (5.1)$$

$$y^{(t)} = \psi_y (W_{hy}h^{(t)} + b_y) \quad (5.2)$$

where W_{xh} , W_{hh} and W_{hy} are the shared transformation matrices from input to hidden states, previous hidden to current hidden states, and hidden to output states. b_h and b_y are bias terms, ψ_h and ψ_y are non-linear activation functions. Since the expression of each state is based on hidden representation of the previous states, SRN is able to keep “memory” of the whole sequence, and learn patterns based on such sequential context.

Although simple and effective, SRN has the unpleasant “short term memory” problem [114]: during the back-propagation procedure in SRN, the gradients will be multiplied T times by the W_{hh} . Consequently, when T is relatively large, there will be gradient vanishing/exploding problems.

5.3.2.2 Long-Short Term Memory Recurrent Neural Nets

To overcome the above “short term memory” issue, LSTM [114] introduces the “memory block” (combined with multiplication gates and memory cell) to keep long term flow of sequential information. As shown in Figure 5.2b, the t -th state in LSTM can be represented as:

$$i^{(t)} = \sigma(W_{hi}h^{(t-1)} + W_{xi}x^{(t)} + b_i) \quad (5.3)$$

$$f^{(t)} = \sigma(W_{hf}h^{(t-1)} + W_{xf}x^{(t)} + b_f) \quad (5.4)$$

$$o^{(t)} = \sigma(W_{ho}h^{(t-1)} + W_{xo}x^{(t)} + b_o) \quad (5.5)$$

$$g^{(t)} = \phi(W_{hg}h^{(t-1)} + W_{xg}x^{(t)} + b_g) \quad (5.6)$$

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)} \quad (5.7)$$

$$h^{(t)} = o^{(t)} \odot \phi(c^{(t)}) \quad (5.8)$$

$$y^{(t)} = \psi_y(W_{hy}h^{(t)} + b_y) \quad (5.9)$$

in addition to the hidden state $h^{(t)}$, LSTM introduced a memory cell $c^{(t)}$, and four multiplication gates: $i^{(t)}$, $f^{(t)}$, $o^{(t)}$, and $g^{(t)}$, which are the input, forget, output, and input modulation gate respectively. σ is a logistic sigmoid function (thus, $i^{(t)}$, $f^{(t)}$, $o^{(t)}$ range from $[0, 1]$). ϕ is the hyperbolic tangent nonlinearity, and $\odot$ represents element-wise multiplication. Specifically, the self recurrent memory cell $c^{(t)}$ keeps the long-term memory. The input gate $i^{(t)}$ controls the flow of incoming signal to alter the state of $c^{(t)}$. The forget gate $f^{(t)}$ helps the $c^{(t)}$ to selectively maintain and forget the previous state status $c^{(t-1)}$. While the output gate $o^{(t)}$ controls the amount of memory that transmits to $h^{(t)}$.

The “memory block” structure enables LSTM to selectively forget its previous memory states, and learn long-term dynamics which general SRN can hardly

Figure 5.2: General SRN and LSTM structures, where the solid arrows represent the forward transformations, and the dashed arrows represent the recurrent connections from the previous states to the current ones. (a) SRN structure. In each state t of the sequence, there are two inputs for the hidden layer $h^{(t)}$: the current state input $x^{(t)}$, and the previous state hidden unit $h^{(t-1)}$. The predicted output label $y^{(t)}$ depends on $h^{(t)}$. (b) LSTM structure. Similar to SRN, but the long-term memory can be kept by the introduced intermediate gates (input $i^{(t)}$, forget $f^{(t)}$, output $o^{(t)}$, input modulation $g^{(t)}$ gates), and the memory cell $c^{(t)}$.

handle. However, LSTM has more intermediate neurons than SRN, thus LSTM consumes much more computational resources.

In this chapter, rather than modeling contextual correlations among different states in time sequences, we modify RNNs to model contextual dependencies among image region “2D sequences”. Details of our proposed networks will be introduced in the following section.

Figure 5.3: Spatial contextual dependencies modeling in a single scale HRNN layer (6×6 image regions, one dashed box corresponds to one image region). (a): Transmit information from left and top spatial neighbors for each region, and process from top-left corner to bottom-right corner in an acyclic way. (b-d): Similar to (a), except the processing directions are top-right to bottom-left, bottom-left to top-right, and bottom-right to top-left respectively. (e): Combine (a-d), then each image region has direct or indirect contextual references from all the other regions.

5.3.3 Hierarchical Recurrent Layers

In CNNs (Section 5.3.1), convolution and pooling are locally performed on image regions. While the spatial dependencies among different regions from the same scale are ignored, let alone the scale dependencies among image regions from different scales. On the other hand, general RNNs (Section 5.3.2) are designed for modeling dependencies in sequences, however, they cannot be directly applied to images. Thus, as shown in the middle part of Figure 5.1, we propose hierarchical

recurrent layers to model spatial and scale contextual dependencies.

5.3.3.1 Modeling Spatial Contextual Dependencies

Spatial context is an important clue for recognizing images. For example, in an image with label “computer room”, knowing the existence of “computer” can help the system to increase the preference of representing “desk” in the surrounding image regions. In this subsection, we will introduce the spatial RNNs to model spatial contextual dependencies within single scale image feature maps.

There are no existing sequences in images, hence we need to generate region sequences in image domain. Take Alex-net [5] as an example, as described in Section 5.3.1, we utilize the fifth layer CNN feature maps ($256 \times 6 \times 6$, corresponding to number of channels $\times$ height $\times$ width) as the input of the recurrent layers. It can be considered as a 6×6 2D data array, each element in the array is represented as a 256 dimensional vector. Then how to convert such an 2D array into sequences? The most straight-forward way is to scan in a row by row or column by column manner. However, images are 2-dimensional data. For each element, contextual information from all the directions should be taken into consideration. Thus, inspired by [105], we generate “2D sequences” for images, and each element simultaneously receives spatial contextual references from its 2D neighborhood elements.

As shown in (e) of Figure 5.3, spatial contextual information comes from all directions (left, right, top, bottom). If we directly connect all the surrounding elements to the target, each node would simultaneously be the “previous” and “next” element of its neighbors. Then the connections would form a cyclic graph. The resulting network is difficult to be optimized. Thus, four directional “2D sequences” are generated for each scale: top-left to bottom-right, bottom-right to top-left,

bottom-left to top-right, and top-right to bottom-left. Each of them focuses on transferring information from an independent direction through an acyclic path. Take the top-left to bottom-right sequence (as shown in (a) of Figure 5.3) as an example, each element receives references from its nearest neighbor elements in the previous row and the previous column. All the elements will be visited once, and every element can be unrolled into a function of all the previously visited elements. Similarly, the contextual information from the other three directions can be encoded through “2D sequences” as shown in (b-d) of Figure 5.3.

For each of the four directional “2D sequences”, the transformation matrices are shared through the whole sequence. To model the spatial correlations among different image regions, general SRN and LSTM (Section 5.3.2) are modified to model our spatial sequences, and they are called spatial SRN and spatial LSTM in the rest of this section.

Spatial SRN Firstly, the spatial SRN is introduced. The hidden representation of each image region in the “2D sequences” should be:

$$h_{\searrow}^{(r,c)} = \psi_h \left(W_{hh\searrow}^r h_{\searrow}^{(r-1,c)} + W_{hh\searrow}^c h_{\searrow}^{(r,c-1)} + W_{xh\searrow} x^{(r,c)} + b_{h\searrow} \right) \quad (5.10)$$

$$h_{\swarrow}^{(r,c)} = \psi_h \left(W_{hh\swarrow}^r h_{\swarrow}^{(r+1,c)} + W_{hh\swarrow}^c h_{\swarrow}^{(r,c+1)} + W_{xh\swarrow} x^{(r,c)} + b_{h\swarrow} \right) \quad (5.11)$$

$$h_{\nearrow}^{(r,c)} = \psi_h \left(W_{hh\nearrow}^r h_{\nearrow}^{(r+1,c)} + W_{hh\nearrow}^c h_{\nearrow}^{(r,c-1)} + W_{xh\nearrow} x^{(r,c)} + b_{h\nearrow} \right) \quad (5.12)$$

$$h_{\nwarrow}^{(r,c)} = \psi_h \left(W_{hh\nwarrow}^r h_{\nwarrow}^{(r-1,c)} + W_{hh\nwarrow}^c h_{\nwarrow}^{(r,c+1)} + W_{xh\nwarrow} x^{(r,c)} + b_{h\nwarrow} \right) \quad (5.13)$$

$$h^{(r,c)} = h_{\searrow}^{(r,c)} + h_{\swarrow}^{(r,c)} + h_{\nearrow}^{(r,c)} + h_{\nwarrow}^{(r,c)} \quad (5.14)$$

where (r, c) is the position of the element. $x^{(r,c)}$ is the input, which is an image region represented by a 256-dimensional fifth CNN layer feature vector, $h_{\searrow}^{(r,c)}$, $h_{\swarrow}^{(r,c)}$, $h_{\nearrow}^{(r,c)}$, and $h_{\nwarrow}^{(r,c)}$ denote the hidden representations of $x^{(r,c)}$ in the four “2D

sequences” respectively (corresponding to top-left to bottom-right, bottom-right to top-left, bottom-left to top-right, and top-right to bottom-left directions). $h^{(r,c)}$ is the combination of the four directional hidden representations, which is the output of spatial SRN. For each direction, W_{hh}^r and W_{hh}^c are row based and column based hidden to hidden states transformation matrices. W_{xh} is the input to hidden states transformation matrix, b_h is the bias term, and ψ_h is a non-linear activation function (ReLU is used here).

Spatial LSTM Similar to the Equation 5.14 in spatial SRN, each hidden unit $h^{(r,c)}$ of spatial LSTM is also a combination of four directional hidden representations. To make the expression concise and clear, only the functions corresponding to the $\searrow$ direction will be expanded here (corresponding to Equation 5.10):

$$i_{\searrow}^{(r,c)} = \sigma \left(W_{hi\searrow}^r h_{\searrow}^{(r-1,c)} + W_{hi\searrow}^c h_{\searrow}^{(r,c-1)} + W_{xi\searrow} x^{(r,c)} + b_{i\searrow} \right) \quad (5.15)$$

$$f_{\searrow}^{(r,c)} = \sigma \left(W_{hf\searrow}^r h_{\searrow}^{(r-1,c)} + W_{hf\searrow}^c h_{\searrow}^{(r,c-1)} + W_{xf\searrow} x^{(r,c)} + b_{f\searrow} \right) \quad (5.16)$$

$$o_{\searrow}^{(r,c)} = \sigma \left(W_{ho\searrow}^r h_{\searrow}^{(r-1,c)} + W_{ho\searrow}^c h_{\searrow}^{(r,c-1)} + W_{xo\searrow} x^{(r,c)} + b_{o\searrow} \right) \quad (5.17)$$

$$g_{\searrow}^{(r,c)} = \phi \left(W_{hg\searrow}^r h_{\searrow}^{(r-1,c)} + W_{hg\searrow}^c h_{\searrow}^{(r,c-1)} + W_{xg\searrow} x^{(r,c)} + b_{g\searrow} \right) \quad (5.18)$$

$$c_{\searrow}^{(r,c)} = f_{\searrow}^{(r,c)} \odot \left(c_{\searrow}^{(r-1,c)} + c_{\searrow}^{(r,c-1)} \right) + i_{\searrow}^{(r,c)} \odot g_{\searrow}^{(r,c)} \quad (5.19)$$

$$h_{\searrow}^{(r,c)} = o_{\searrow}^{(r,c)} \odot \phi \left(c_{\searrow}^{(r,c)} \right) \quad (5.20)$$

where (r, c) is the current state position, $x^{(r,c)}$ represents the current input data. $i_{\searrow}^{(r,c)}$, $f_{\searrow}^{(r,c)}$, $o_{\searrow}^{(r,c)}$, $g_{\searrow}^{(r,c)}$, correspond to the input, forget, output, and input modulation gates. $c_{\searrow}^{(r,c)}$ denotes the memory cell unit, and finally $h_{\searrow}^{(r,c)}$ is the hidden representations of $x^{(r,c)}$ in top-left to bottom-right direction. Similarly, the other three directions can be achieved.

For each gate function, W_{hp}^r , W_{hp}^c , W_{xp}^c and b_p ($p \in \{i, f, o, g\}$) are hidden-gate

(row), hidden-gate (column), input-gate transformation matrices and bias terms respectively. Both σ and ϕ are non-linear activation functions, in this chapter, sigmoid is used as σ , and tangent is assigned as ϕ .

5.3.3.2 Modeling Scale Contextual Dependencies

Besides spatial contextual dependencies, there also exist scale contextual dependencies among image regions from the same locations but at different scales, which is another important clue for image recognition. For example, again in an image with label “computer room”, knowing the global pattern “computer room” can help the system to increase the preference of representing patterns correspond to “computer” and “desk” in local-level scales. In this subsection, we will focus on modeling scale dependencies.

The final goal of image classification is to achieve a good image-level description, which is based on well-performed local image region representations. When describing a local image region, the traditional way is to only encode its own information. In contrast, if information from higher-level scale regions is given, then the global information would be encoded in the representation of local regions, and lead to more accurate local region descriptions. Thus, we build connections across regions from different scales.

For each element at each scale, its receptive field covers a number of elements at the lower-level scales. More intuitively, as shown in the middle part of Figure 5.1, areas highlighted with yellow at the scale $l + 1$ and $l + 2$ are covered by the receptive field of the yellow element at the scale l . Thus, global information from the higher-level scale l would be transferred to the corresponding areas at the lower-level scales $l + 1$ and $l + 2$. Thus, for element at position (r_l, c_l) on scale l , the scale dependencies from higher-level scales can be encoded through the following

function:

$$s_l^{(r_l, c_l)} = \sum_{j=1}^{l-1} W_{jl} h_j^{(r_j, c_j)} \quad (5.21)$$

where $l \in [2, \dots, L]$, and L is the number of scales. (r_j, c_j) is the position at the higher-level scale j . $h_j^{(r_j, c_j)}$ is scale contextual element (already combined the four directional spatial dependencies, refer to Equation 5.14) from the higher-level scale, and W_{jl} is the scale j to scale l transformation matrix.

5.3.3.3 HRNNs with Spatial & Scale Dependencies

By inserting Equation 5.21 into Equation 5.10-5.13 (spatial SRN) or Equation 5.15-5.20 (spatial LSTM), scale and spatial dependencies can be modeled together in our hierarchical RNNs.

HSRN Take the top-left to right-bottom directional HSRN as an example (refer to Equation 5.10), the hidden representation of each element is:

$$\begin{aligned} \tilde{h}_{\searrow}^{(r, c)} &= \psi_h \left(W_{hh\searrow}^r \tilde{h}_{\searrow}^{(r-1, c)} + W_{hh\searrow}^c \tilde{h}_{\searrow}^{(r, c-1)} + s^{(r, c)} + W_{xh\searrow} x^{(r, c)} + b_{h\searrow} \right) \\ s^{(r, c)} &= \sum_{j=1}^{l-1} W_{jl} \tilde{h}_j^{(r_j, c_j)} \end{aligned} \quad (5.22)$$

HLSTM Similarly, for the HLSTM model (refer to Equation 5.15-5.18), the hidden representation of each gate functions is:

$$\begin{aligned} \tilde{p}_{\searrow}^{(r, c)} &= \sigma \left(W_{hp\searrow}^r \tilde{h}_{\searrow}^{(r-1, c)} + W_{hp\searrow}^c \tilde{h}_{\searrow}^{(r, c-1)} + s^{(r, c)} + W_{xp\searrow} x^{(r, c)} + b_{i\searrow} \right) \\ s^{(r, c)} &= \sum_{j=1}^{l-1} W_{jl} \tilde{h}_j^{(r_j, c_j)} \\ \tilde{p} &\in \{\tilde{i}, \tilde{f}, \tilde{o}, \tilde{g}\} \end{aligned} \quad (5.23)$$

in which, σ is sigmoid function when $\tilde{p} \in \{\tilde{i}, \tilde{f}, \tilde{o}\}$, and σ is tangent when $\tilde{p} = \tilde{g}$.

For both Equation 5.22 and 5.23, the scale index l of each variable is removed for the convenience of expression. Similarly, expressions of the other three directions can be obtained. Afterwards, refer to Equation 5.14, by combining the revised four directional hidden representations, the complete HRNNs (for both HSRN and HLSTM) hidden element expression would be:

$$\tilde{h}^{(r,c)} = \tilde{h}_{\searrow}^{(r,c)} + \tilde{h}_{\swarrow}^{(r,c)} + \tilde{h}_{\nearrow}^{(r,c)} + \tilde{h}_{\nwarrow}^{(r,c)} \quad (5.24)$$

According to the RNNs optimization notes in [44], RNNs can be simply and effectively optimized by back propagation through time (BPTT), which is a variant of back propagation for time sequences. In BPTT, the recurrent nets would be unfolded into feed-forward deep networks, then normal back-propagation can be applied. Utilizing the “weight sharing” setting in Caffe [115], BPTT can be performed with shared RNN weights.

5.3.4 Fully Connected Layers

Different from applications like image labeling, where the label $y^{(r,c)}$ of each pixel or patch level image region $x^{(r,c)}$ is given, in image classification, there is no intermediate labels except the overall image-level label. Thus, Equation 5.2 (corresponds to HSRN) or Equation 5.9 (corresponds to HLSTM) cannot be directly applied. To connect the hierarchical recurrent layers (Equation 5.24) to the image labels, fully connected layers are introduced to merge the information learned by different

scales of HRNNs:

$$g = \psi_g(W_{hg}H + b_g) \quad (5.25)$$

$$y = \varphi_o(W_{go}g + b_o) \quad (5.26)$$

$$\text{where } H = [\tilde{h}_1^T, \dots, \tilde{h}_l^T, \dots, \tilde{h}_L^T]^T$$

$$\text{and } \tilde{h}_l = [(\tilde{h}_l^{(1,1)})^T, \dots, (\tilde{h}_l^{(r_l, c_l)})^T, \dots, (\tilde{h}_l^{(R_l, C_l)})^T]^T$$

where W_{hg} is the fully connected transformation matrix to transform the HRNNs output H to the global hidden layer g . H is the concatenation of HRNN layer outputs $\tilde{h}_l$ ($l \in [1, \dots, L]$) at different scales. For each scale, $\tilde{h}_l$ is the concatenation of all its hidden element expressions $\tilde{h}_l^{(r_l, c_l)}$ ($r_l \in [1, \dots, R_l]$, $c_l \in [1, \dots, C_l]$, R_l and C_l are the number of rows and columns at the scale l respectively). W_{go} is learned to connect g with the class label y , b_g and b_o are the bias terms. ψ_g is a non-linear activation function (ReLU is used in this chapter), and φ_o is the softmax function for classification.

5.4 Experiments

In this section, detailed network structures and settings of our end-to-end C-HRNNs are firstly introduced. Next, C-HRNNs are compared with other popular methods on four challenging object/scene image classification benchmarks: ILSVRC 2012 [54], Places 205 [55], SUN 397 [56], and MIT indoor [52]. Afterwards, effectiveness of different modules of C-HRNNs is analyzed, C-HSRN and C-HLSTM are compared in detail.

5.4.1 Experimental Settings

Following the default data preprocessing settings in Caffe [115], all images are re-sized to 256×256 pixels and subtracted by the pixel mean. For training images, 10 sub-crops of size 224×224 (1 center, 4 corners, and horizontal flips) are extracted. In the remaining part of this section, if not specified, the results are the Top 1 accuracy (or error rates) tested with center crop by using a single model.

As shown in Table 5.1, detailed layer structures of the baseline deep networks (Alex-net [5], ZF-net [113], and SPP-net [29]) and our proposed C-HRNNs are given.

Comparing with ZF-net and SPP-net, our C-HRNNs has the same first five convolutional layers: $96(7 \times 7)$, $256(5 \times 5)$, $384(3 \times 3)$, $384(3 \times 3)$, and $256(3 \times 3)$ respectively. Strides of the first two layers are 2, and the rest are 1. Following each of the first, second and fifth convolutional layers, there is a max pooling layer with kernel size of 3×3 , and stride of 2. Finally, size of output feature maps of the fifth CNN layer is $256 \times 6 \times 6$ (number of channels $\times$ height $\times$ width). Similar to SPP-net, we pool the feature maps into four scales, and achieve response maps with size of $\{6 \times 6, 3 \times 3, 2 \times 2, 1 \times 1\}$.

Different from all of these baseline networks, our C-HRNNs introduce hierarchical recurrent layers (hrnn6 as shown in Table 5.1). For the hierarchical recurrent layers, we process three scale spatial RNN layers with size of $\{6 \times 6, 3 \times 3, 2 \times 2\}$ and one global 1×1 pooling layer, and build cross scale connections among all these four scales. The corresponding numbers of image regions of the four scales are 36, 9, 4, and 1 respectively, and each region is represented as a 256-dimensional feature vector (number of channels in the fifth layer CNN). For each RNN layer, sizes of the transformation matrices (HSRN: W_{hh}^r , W_{hh}^c , W_{xh} , and W_{jl} , refer to

Models	conv1	conv2	conv3	conv4	conv5	hrnn6	fc7	fc8
Alex-net [5]	96x11x11 st. 4, pad 0 LRN, x2 pool map 27x27	256x5x5 st. 1, pad 2 LRN, x2 pool map 13x13	384x3x3 st. 1, pad 1 - map 13x13	384x3x3 st. 1, pad 1 - map 13x13	256x3x3 st. 1, pad 1 x2 pool map 6x6	-	4096 drop- out 0.5	4096 drop- out 0.5
ZF-net [113]	96x7x7 st. 2, pad 1 LRN, x2 pool map 55x55	256x5x5 st. 2, pad 0 LRN, x2 pool map 13x13	384x3x3 st. 1, pad 1 - map 13x13	384x3x3 st. 1, pad 1 - map 13x13	256x3x3 st. 1, pad 1 x2 pool map 6x6	-	4096 drop- out 0.5	4096 drop- out 0.5
SPP-net [29]	96x7x7 st. 2, pad 1 LRN, x2 pool map 55x55	256x5x5 st. 2, pad 0 LRN, x2 pool map 13x13	384x3x3 st. 1, pad 1 - map 13x13	384x3x3 st. 1, pad 1 - map 13x13	256x3x3 st. 1, pad 1 {x2, x4, x7, x13}pool map {6x6, 3x3, 2x2, 1x1}	-	4096 drop- out 0.5	4096 drop- out 0.5
C-HRNNs	96x7x7 st. 2, pad 1 LRN, x2 pool map 55x55	256x5x5 st. 2, pad 0 LRN, x2 pool map 13x13	384x3x3 st. 1, pad 1 - map 13x13	384x3x3 st. 1, pad 1 - map 13x13	256x3x3 st. 1, pad 1 {x2, x4, x7, x13}pool map {6x6, 3x3, 2x2, 1x1}	{36, 9, 4, 1}x256x1x1 st.1, dropout 0.5 - map {6x6, 3x3, 2x2, 1x1}	4096 drop- out 0.5	4096 drop- out 0.5

Table 5.1: Network structures. C-HSRN and C-HLSTM share the same structure, which are indicated as C-HRNNs.

Equation 5.22; HLSTM: W_{hp}^r , W_{hp}^c , W_{xp} , and W_{jl} , refer to Equation 5.23) in the four directional “2D sequences” are 256×256 .

To show the performance gain of introducing spatial and scale dependencies separately, in the following experiments, we introduce an intermediate network called convolutional multi-scale recurrent neural networks (C-MRNNs), which only considers spatial dependencies in multiple scales, and ignores the scale dependencies. Specifically, convolutional multi-scale simple recurrent neural network (C-MSRN) and convolutional multi-scale long-short term memory neural network (C-MLSTM) are tested in the experiments. Furthermore, when all the hidden-hidden weights W_{hh} in C-MSRN are set to 0, and the input-hidden weights W_{ih} are identity matrices, C-MSRN degenerates to SPP-net.

For the fully connected layers, the number of output units of both two layers is 4096, and each of them is applied dropout at the rate of 0.5.

The training batch size is 256, learning rate starts from 0.01 and it is divided by 10 when the accuracy stops increasing, and the weight of momentum is 0.9. All the experiments are run on Caffe [115] with a single NVIDIA Tesla K40 GPU. For ImageNet and Places 205, since there are millions of training data, we randomly initialize all the network parameters and train from scratch. While for SUN 397 and MIT indoor, we utilize the models pretrained on ImageNet and Places to do fine-tuning.

5.4.2 Experimental Results

5.4.2.1 Experimental Results on ILSVRC 2012

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) dataset [54] is one of the most challenging and popular large-scale object image classification datasets.

Methods	test scales	test views	Top 1 val	Top 5 val
MOP-CNN [103] (max pooling)	3	101	44.12%	-
MOP-CNN [103] (VLAD pooling)	3	101	42.07%	-
SPP-net [29]	1	1	38.21%	-
C-MSRN	1	1	36.90%	-
C-HSRN	1	1	36.38%	-
C-MLSTM	1	1	36.01%	-
C-HLSTM	1	1	35.85%	-
Alex-net [5]	1	10	40.7%	18.2%
ZF-net [113]	1	10	38.4%	16.5%
Overfeat[31]	1	10	35.6%	14.7%
SPP-net [29]	1	10	36.2%	14.9%
C-MSRN	1	10	35.2%	14.0%
C-HSRN	1	10	34.8%	13.7%
C-MLSTM	1	10	34.5%	13.5%
C-HLSTM	1	10	34.3%	13.4%

Table 5.2: Comparison of error rates on the ILSVRC 2012 validation set.

ILSVRC 2012 contains 1.2 million training images and 50,000 validation images (50 per class), and they belong to 1000 object categories.

In the upper part of Table 5.2, we compare C-HRNNs with SPP-net[29], which encodes the spatial information by using spatial pyramid pooling. Based on their released model¹, we can only achieve 41.47% top-1 error rate with one testing view. Therefore, we further tune the model with the settings in Section 5.4.1, and finally achieve 38.21% (reported 38.01%) for SPP-net. The performance gap might be caused by the different training settings (when preprocess images, SPP-net keeps the original image aspect ratio, while the standard Caffe [115] does not). Our C-HRNNs and SPP-net use exactly the same convolutional and fully connected layer settings, except SPP-net directly applied $\{6 \times 6, 3 \times 3, 2 \times 2, 1 \times 1\}$ spatial pyramid pooling to pool features from the fifth convolutional layer, while our C-HRNNs

¹https://github.com/ShaoqingRen/SPP_net

model spatial dependencies with RNN for each scale separately, and models scale dependencies across different scales.

For SRN models, comparing with SPP-net, C-MSRN brings 1.31% Top 1 error rate decrease, which indicates the benefit of modeling spatial dependencies. After integrating the scale dependencies, C-HSRN is 1.83% better than SPP-net. Thus, both encoding spatial and scale dependencies can help to generate better image representations.

In LSTM models, performance improvement introduced by modeling spatial and scale dependencies can also be observed. Different from SRN models, LSTM models are able to keep longer-term memory of image region “2D sequences”. Comparing with SPP-net, C-HLSTM is 2.36% better. C-MLSTM achieves 36.01%, which is better than C-MSRN, and C-HLSTM (35.85%) also works better than C-HSRN. But the performance gap between C-HLSTM and C-HSRN (0.53%) is less than the one between C-MLSTM and C-MSRN (0.89%). The reason should be that the introduced scale dependencies from higher scales indirectly extend the long-term ability of C-MSRN, and indent the gap between SRN models and LSTM models.

We also compare with another spatial statistics based CNN method MOP-CNN [103], which directly uses the Caffe CNN [115] to densely extract features from three-scale image patches, and use VLAD pooling to generate global representations. The performance gap indicates that our way of encoding spatial and scale information is more effective.

In the lower part of Table 5.2, C-HRNNs are compared with other deep neural networks with the most general settings: 10 testing views, comparing the Top 1 and Top 5 error rates. The outstanding performances of our C-HRNNs indicate that besides going deeper and wider, RNN is another promising way to increase

Methods	Top 1 val	Top 5 val
Alex-net [5, 116]	50.06%	80.51%
SPP-net [29]	51.57%	81.88%
C-MSRN	52.70%	82.75%
C-HSRN	53.16%	83.07%
C-MLSTM	53.75%	83.36%
C-HLSTM	53.91%	83.48%

Table 5.3: Comparison of accuracy on Places 205.

the image representation power of neural networks.

5.4.2.2 Experimental Results on Places 205

Places 205 dataset [116] is currently the largest scene categorization dataset, which has just been released at the end of 2014. Different from ILSVRC 2012, it focuses on scene images rather than object centric ones. It has 2.5 million training images from 205 scene categories, which is twice the size of ILSVRC 2012, and much more challenging. There are 20,500 images (100 per category) in the validation set.

As shown in Table 5.3, C-HLSTM update the state-of-the-art result on Places 205 (previous best result was 50.06% achieved by Alex-net) with the accuracy of 53.91%. Considering the performance gain of merely introducing spatial dependencies, C-MSRN and C-MLSTM outperform SPP-net by 1.13% and 2.18% respectively. Further integrating scale dependences, C-HSRN and C-HLSTM bring 1.59% and 2.34% improvements correspondingly.

5.4.2.3 Experimental Results on SUN 397

SUN 397 [56] is another popular large-scale scene image recognition benchmark. There are 100,000 images from 397 scene classes in total. The general splittings in [56] are used here, in which, there are 50 images per class for training, and

Methods	Accuracy
MOP-CNN [103] (max pooling)	48.50%
MOP-CNN [103] (VLAD pooling)	51.98%
Alex-net [5] (ILSVRC ft)	44.42%
Alex-net [5] (Places ft)	54.55%
SPP-net [29] (ILSVRC ft)	49.02%
C-MSRN (ILSVRC ft)	51.76%
C-HSRN (ILSVRC ft)	52.59%
C-MLSTM (ILSVRC ft)	52.67%
C-HLSTM (ILSVRC ft)	52.78%
SPP-net [29] (Places ft)	57.23%
C-MSRN (Places ft)	59.32%
C-HSRN (Places ft)	59.90%
C-MLSTM (Places ft)	60.08%
C-HLSTM (Places ft)	60.34%
Xiao et al.[56]	38.00%
IFV [117]	47.20%
MTL-SDCA [118]	49.50%

Table 5.4: Comparison of accuracy on SUN 397.²

50 images per class for testing. Since the number of training images is too small (20,000), we introduce the models pre-trained on ILSVRC 2012 and Places 205, and use the training images from SUN 397 to fine-tune the network. We also increase the learning rates of the HRNN layers (10 times higher than the other layers), and aim to focus more on spatial dependencies specifically exist in SUN 397.

As shown in the upper part of Table 5.4, our C-HRNNs performs better than existing CNNs. After fine-tuning on SUN 397, C-HRNNs are able to learn more data adaptive spatial dependencies and significantly outperform SPP-net: 1) Based on models pre-trained on ILSVRC, the performance gains of C-HSRN and C-HLSTM are 3.57% and 3.76% respectively; 2) Based on models pre-trained on Places, the accuracy improvements of C-HSRN and C-HLSTM are 2.67% and

3.11% correspondingly.

When applying fine-tuning based on Places 205, C-HLSTM achieves the state-of-the-art performance on the SUN 397 with the classification accuracy of 60.34%, which outperforms the previous best result (MOP-CNN 51.98%) by 8.36%. Another observation is that the performances of the fine-tuned models based on Places 205 consistently perform better than the ones based on ILSVRC 2012. The reason should be that both Places 205 and SUN 397 are scene datasets, their domain gap is smaller than the gap between ILSVRC 2012 (object dataset) and SUN 397.

In the lower part of Table 5.4, traditional state-of-the-art shallow methods are compared with the CNNs. Most of these works heavily depend on combining multiple densely extracted hand-crafted features, and the image level representations are usually very high-dimensional. Another drawback of these methods is that the testing procedures are generally very time consuming, because of the feature extraction steps are slow. Comparing with them, our C-HRNNs perform much better with much less computational cost in testing, and much lower-dimensional features.

5.4.2.4 Experimental Results on MIT Indoor

As mentioned in Section 4.5.1, MIT indoor [52] is a very challenging and popular indoor scene image classification benchmark. This dataset focuses on indoor scene scenarios, which usually contains lots of objects, and has larger variations than outdoor scenes. There are 67 different scene scenarios in MIT indoor in total, and the widely used splitting provided by [52] are applied in our experiments. In each class, around 80 training images, and around 20 testing images are selected.

²ILSVRC ft and Places ft represent the models fine-tuned based on the models pre-trained on ILSVRC 2012 and Places 205 respectively.

Methods	Accuracy
MOP-CNN [103] (max pooling)	64.85%
MOP-CNN [103] (VLAD pooling)	68.88%
Alex-net [5] (ILSVRC ft)	61.57%
Alex-net [5] (Places ft)	68.24%
SPP-net [29] (ILSVRC ft)	66.32%
C-MSRN (ILSVRC ft)	68.28%
C-HSRN (ILSVRC ft)	68.88%
C-MLSTM (ILSVRC ft)	69.18%
C-HLSTM (ILSVRC ft)	69.25%
SPP-net [29] (Places ft)	72.09%
C-MSRN (Places ft)	74.18%
C-HSRN (Places ft)	74.85%
C-MLSTM (Places ft)	75.30%
C-HLSTM (Places ft)	75.67%
ROI + GIST [52]	26.50%
Object Bank [97]	37.60%
Visual Concepts [80]	46.40%
MMDL [100]	50.15%
IFV [83]	60.77%
MLrep + IFV [81]	66.87%
ISPR + IFV [102]	68.50%

Table 5.5: Comparison of accuracy on MIT indoor.²

Because of the limitation of dataset size, we also utilize the pre-train models on ILSVRC 2012 and Places 205, and do fine-tuning. For the other baseline deep neural networks, such fine-tuning is also applied.

As shown in the upper part of Table 5.5, C-HRNNs are able to outperform the other deep neural nets with obvious gaps. Comparing with the state-of-the-art MOP-CNN, our C-HLSTM achieves the accuracy of 75.67%, which is 6.79% better. Comparing with SPP-net: 1) Based on models pre-trained on ILSVRC, the performance gains of C-HSRN and C-HLSTM are 2.56% and 2.93% respectively; 2) Based on models pre-trained on Places, the accuracy improvements of C-HSRN and C-HLSTM are 2.76% and 3.58% correspondingly.

In the lower part of Table 5.5, results of the state-of-art traditional shallow methods are given. Although very powerful on MIT Indoor, these methods cost much more computation power to perform middle-level patch searching and clustering, the feature dimensions are relatively high, and most of them can hardly be applied to large-scale benchmarks. In contrast, our C-HRNNs are end-to-end feature learning frameworks with 4096-dimensional output features, and C-HRNNs can easily handle large-scale data.

5.4.3 Analysis of C-HRNNs

In this subsection, we will analyze the effectiveness of our C-HRNNs from different perspectives.

5.4.3.1 C-HRNNs Visualization

Firstly, patterns learned by the hrnn6 layer (refer to Table 5.1) of C-HLSTM are visualized in Figure 5.4. On the left part of Figure 5.4, six testing image region on the $\{3 \times 3\}$ scale (refer to Section 5.4.1) are given, and the receptive field of each region is highlighted with blue rectangle in the original images. On the right part of Figure 5.4, the top 8 nearest neighbors of each testing image region are shown. These nearest neighbors are searched from all the local region features extracted from training images, and measured by χ^2 distance. For every two nearest neighbors rows, the top row is the nearest neighbors searched by utilizing C-HLSTM hrnn6 layer features, and the bottom row is the results of using SPP-net conv5 layer features.

Comparing the visualization results of our C-HLSTM with the SPP-net, we can observe obvious better local image region representations. Take the first testing

Figure 5.4: Visualization of the hrnn6 (C-HLSTM)/conv5 (SPP-net) layer features. The colored rectangles mark the receptive fields of each image region. First column on the left: testing image regions (blue rectangles) at scale 3x3. For each testing region, the regions on the right are the top 8 nearest neighbors searched by using C-HLSTM hrnn6 features (top row), and SPP-net conv5 features (bottom row). Green rectangles are the nearest neighbors with the same labels as the testing image region, red rectangles are the ones with the wrong labels. (Best viewed in color)

Figure 5.5: Same as Figure 5.4 for more sample images. (Best viewed in color)

image region as an example, it is the right bottom area in the “radiator grille” image. By using our C-HLSTM, this region is more likely to be represented as the “radiator grille” from the same or different car models, and less likely to be mismatched to similar patterns from other unrelated classes. Take the last testing image “partridge” as another example. The testing region contains “body of partridge” and the background “gravel”. For our C-HLSTM, contextual information has been taken into consideration, thus, this region is represented as the “body of

partridge”. In contrast, the SPP-net wrongly focused on “gravel”, and missed the target object. Similarly, better local region visualization results of C-HLSTM can be observed in other classes, such as man-made buildings like “steel arch bridge”, creatures like “sea urchin” etc.

5.4.3.2 RNNs vs Modified CNNs

Since C-HRNNs have more parameters than the original CNNs, we aim to quantitatively show whether the performance gain is from encoding contextual dependencies, or simply from increasing the number of parameters. For comparison convenience, the MRNNs (multi-scale RNN) are compared with the modified CNN.

For each MRNN scale, there are four directional “2D sequences”. 1) For MSRN, each direction has three transformation matrices W_{hh}^r , W_{hh}^c , and W_{xh} ; 2) For MLSTM, each direction has four gate functions, and in each gate, there are W_{hp}^r , W_{hp}^c , and W_{xp} . Thus, for MRNN, there are 12 transformation matrices, while for MLSTM, there are 48 matrices, all of these transformation matrices have size 256×256 . We replace each MRNN layer with 12/48 parallel convolutional layers with 256 kernels of size $1 \times 1 \times 256$, and gather them together afterwards, which are called modified CNN here.

Testing on ILSVRC 2012, the MSRN achieves 36.90% in error rate, while the modified CNN gets 37.73%. Similarly, the MLSTM gets 36.01% in error rate, while the modified CNN gets 37.49%. These obvious gaps indicate that RNNs are able to learn contextual dependencies which cannot be captured by CNNs.

5.4.3.3 Effect of Number of Spatial Context Directions

In our C-HRNNs, four directional “2D sequences” are employed, are they really able to learn complementary information to each other?

In Table 5.6, performance of C-HRNNs with different directions of spatial context are given. The first four rows of Table 5.6 show the performances of using each single directional “2D sequence”. From which, we can observe that different direction performs similarly to each other. On the last three rows of Table 5.6, results of combination of two directions, and the complete four directions are given. Comparing the results of using two directions and single direction, improvements can be observed. When combining all the four directions, spatial dependencies from all directions can be covered, and the best performance can be achieved.

Methods	Error	Methods	Error
C-HSRN ₁ (↘)	36.96%	C-HLSTM ₁ (↘)	36.39%
C-HSRN ₁ (↖)	37.03%	C-HLSTM ₁ (↖)	36.46%
C-HSRN ₁ (↗)	36.95%	C-HLSTM ₁ (↗)	36.45%
C-HSRN ₁ (↙)	36.89%	C-HLSTM ₁ (↙)	36.50%
C-HSRN ₂ (↘↖)	36.85%	C-HLSTM ₂ (↘↖)	35.97%
C-HSRN ₂ (↗↙)	36.59%	C-HLSTM ₂ (↗↙)	36.03%
C-HSRN ₄	36.38%	C-HLSTM ₄	35.85%

Table 5.6: Error rates of applying C-HRNNs with different spatial contextual directions on ILSVRC 2012. C-HSRN₁ and C-HLSTM₁ use one directional HRNNs; C-HSRN₂ and C-HLSTM₂ utilize two directions; C-HSRN₄ and C-HLSTM₄ use all the four directions.

5.4.3.4 HRNNs Complexity

Although very powerful, our HRNNs do not bring much extra computational burden or memory usage.

There are three scale HRNN layers with spatial dependencies encoded: 6×6 , 3×3 , and 2×2 , each of them has 12 transformation matrices in HSRN and 48 transformation matrices in HLSTM, and there are 6 hierarchical connections (3 from 1×1 , 2 from 2×2 , and 1 from 3×3). Thus, there are 42 transformation matrices in HSRN and 150 matrices in HLSTM in total, each one has size of

Figure 5.6: Selected per class accuracies tested on Places 205: SPP-net vs C-HLSTM, where the horizontal axis represent the 205 different scene categories, and the vertical axis indicates the accuracy percentage differences compared with SPP-net. Left 10 classes: the top 10 classes on which C-HLSTM works better than SPP-net; Right 10 classes: the top 10 classes on which C-HLSTM works worse than SPP-net.

256×256 . Thus, the HSRN layers have 2,752,512 parameters, and the HLSTM layers have 9,830,400 parameters. The number of parameters in HLSTM is almost four times HSRN, which make the HLSTM models be able to learn more complex patterns with the price of more computation resources. In contrast, in CNN, the fully connected layers have most of the network parameters, e.g. the second fully connected layer needs to learn a 4096×4096 weight matrix, which has 16,777,216 parameters, comparing with which, our HRNN layers have much fewer parameters.

In terms of memory consumption, HRNN layers do not cost much extra memory except some intermediate hidden layer output, i.e. $h^{(r,c)}$ and gate units $p^{(r,c)}$ (only exist in HLSTM models) for each image region, which are 256-dimensional vectors. While in CNN, the most memory consuming part is the first convolutional layer. In our setting, output of the first CNN layer has 1,161,600 dimensions, comparing with which, the HRNNs costs negligible extra memory to save intermediate data.

5.4.3.5 C-HRNNs Success & Failure Cases

Figure 5.6 shows the per class accuracies of C-HLSTM and SPP-net on Places 205 [116]. The left 10 items of Figure 5.6 are the top 10 classes on which C-HLSTM brings most performance gain than SPP-net. While the right 10 items of Figure 5.6 are the 10 classes on which C-HLSTM brings most accuracy drops.

In Figure 5.7, the final classification results of using C-HLSTM and SPP-net [29] are visually compared. The Figure 5.7a shows the results on SUN 397 (fined-tuned based on ILSVRC 2012 models). Figure 5.7b shows the results on Places 205. For each dataset, we show the images on which C-HLSTM leads to the highest accuracy improvement (the two rows above the dashed line), and the images on which C-HLSTM leads to the highest accuracy drop (the row below the dashed line).

From the first two rows of Figure 5.7a, we can clearly observe that the SPP-net focuses on predicting image regions, while ignoring the contextual information. For example, the label of the first image in the first row is “landing deck”, our C-HLSTM can correctly recognize it, while the SPP-net wrongly recognizes it as the “windmill” with a very high confidence score. It’s because SPP-net wrongly recognized the rotor blades of the helicopter as the windmill blades. In contrast, our C-HLSTM takes the context such as the body of helicopter and deck into consideration. Thus, our C-HLSTM works better when the local image regions are confusing, but contextual information can help to make better decisions.

In the third row of Figure 5.7a, we observe some interesting results. For example, the first image of the third row is “rope bridge”, which is relatively small in the image, while the forest is more obvious. Thus, our C-HLSTM wrongly recognize it as “rainforest”. For the third image of the third row, the first word that comes

Figure 5.7: C-HLSTM vs SPP-net. (a) Result on SUN 397; (b) Results on Places 205. For each dataset, the two rows above the dashed line: images misclassified by SPP-net, but correctly classified by C-HLSTM. The row below the dashed line: images correctly classified by SPP-net, but misclassified by C-HLSTM. Under each image, the first row shows the predicted label of using C-HLSTM, and the second row shows the predicted label of using SPP-net, the prediction confidence scores are shown in the bracket, and correct labels are in bold.

to mind is cliff, while the ground truth label “light house” just represents a small region on the “cliff”. Thus, our C-HLSTM makes mistakes when class labels are based on local regions rather than the global image.

Similar phenomena can also be observed on Places 205 (Figure 5.7b).

5.5 Conclusions

In this chapter, we propose an end-to-end deep learning framework to encode spatial and scale contextual dependencies in image representation, which is called C-HRNNs. In C-HRNNs, CNN layers are firstly utilize to extract middle-level representations for local image regions. Based on the CNN layer outputs, our proposed hierarchical recurrent layers are then applied to model the spatial dependencies among different image regions from the same scale, and the scale dependencies among image regions from different scales but at the same locations. In our proposed hierarchical recurrent neural networks, HSRN and HLSTM are introduced as two specific instances, which correspond to a fast recurrent model, and a sophisticated but more effective recurrent model respectively. By integrating CNN and HRNNs, our C-HRNNs show outstanding performances on large-scale image classification.

Chapter 6

Conclusions & Future Works

6.1 Conclusions

This thesis focuses on studying hierarchical feature learning methods for image categorization. Specifically, discriminative information, shareable correlation, and contextual dependencies among image regions are explored and encoded in image representation.

Firstly, we propose a discriminative hierarchical feature learning method to encode class-wise discriminative information in image representation for object image classification. In the learning scheme, the distance values between features from the same classes are forced to be small, while the distance values between features from different classes are forced to be big. The distance is defined as our proposed “Patch-to-Class” distance (measure the distance between a feature descriptor and a class).

Secondly, to solve the scene image classification problem, deep discriminative and shareable correlations among different scene scenarios are encoded in local features progressively layer by layer. Two discriminative training exemplar selec-

tion pre-processing methods are also proposed to further speed up the training procedure and enhance the discriminative power of the learned features.

Thirdly, the end-to-end convolutional and hierarchical recurrent neural networks are proposed for large-scale image recognition. Spatial and scale dependencies among different local image regions are modeled by hierarchical recurrent neural networks, and convolutional neural networks are integrated to further enhance the discriminative representation power.

6.2 Future Works

To conclude the thesis, I would like to discuss some research directions which can be explored in the future.

For the discriminative feature learning method introduced in Chapter 3 and deep discriminative and shareable feature learning method described in Chapter 4, the discriminative and shareable information can be encoded in other learning frameworks. For example, in CNNs, image regions are processed with the same filters, and the filters are highly redundant. Is it possible to take the class-wise correlations into consideration, and selectively train convolution filters that are less redundant while keeping the most discriminative patterns? Afterwards, CNNs might need fewer network parameters to achieve the same performance.

For the convolutional and hierarchical recurrent neural networks (CHRNNs) presented in Chapter 5, there are lots of future extensions that can be done. RNNs are able to model contextual information with a very limited number of network parameters, thus, RNNs can be easily integrated with other frameworks. For example, our HRNNs currently build on CNNs, it can be easily constructed based on any network other than CNNs (e.g. deep restricted Boltzmann machine [46],

auto-encoder [107]), hand crafted features (e.g. SIFT [1], HOG [2]), or even from scratch, better combinations or structures can be further explored. Furthermore, it should also be interesting to apply RNNs on other computer vision applications to encode spatial/sequential context, such as face recognition, video recognition, image labeling/segmentation, saliency detection, etc.

In a more general aspect, in current deep neural networks, the layer structures are usually fixed when doing training, for different datasets, the “best fit” network should be various. For instance, the Alex-net [5] has seven layers, and it works very well on ImageNet [54]. However, directly training such a big network on small datasets like Caltech 101 [50] will come across severely overfitting problem, and lead to very unsatisfactory results. Is it possible to automatically learn network structures given input data, or build dynamic network structures which can be adaptive to the size of input data? There are some related works, such as reconfigurable CNNs [119], in which the network configuration is allowed to be partially activated during inference. In contrast, we propose to dynamically inserting or removing layers based on the input data. For each layer, a supervised pre-training will be firstly applied, and new layers will be added until the performance cannot be further improved. Afterwards fine tuning will be processed for the whole network. I believe this is another promising future direction to further extend the flexibility of deep neural networks.

Author's Publications

1. **Zhen Zuo**, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, and Bing Wang. “Learning Contextual Dependencies with Convolutional Hierarchical Recurrent Neural Network.” Submitted to Transactions on Image Processing (TIP), IEEE, 2015.
2. **Zhen Zuo**, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. “Convolutional Recurrent Neural Networks: Learning Spatial Dependencies for Image Representation.” In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2015.
3. Shuai, Bing, **Zhen Zuo**, and Gang Wang. “Quaddirectional 2D-Recurrent Neural Networks For Image Labeling.” Signal Processing Letters (SPL), IEEE, 2015.
4. Shuai, Bing, Gang Wang, **Zhen Zuo**, Bing Wang, and Lifan Zhao. “Integrating Parametric and Non-Parametric Models For Scene Labeling.” In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4249-4258. IEEE, 2015.
5. **Zhen Zuo**, Gang Wang, Bing Shuai, Lifan Zhao, and Qingxiong Yang. “Exemplar based Deep Discriminative and Shareable Feature Learning for scene

- image classification.” Pattern Recognition (PR), Elsevier, 2015.
6. **Zhen Zuo**, Gang Wang, Bing Shuai, Lifan Zhao, Qingxiong Yang, and Xudong Jiang. “Learning discriminative and shareable features for scene classification.” In Proceedings of the European Conference on Computer Vision (ECCV), pp. 552-568. Springer, 2014.
 7. **Zhen Zuo**, and Gang Wang. “Learning discriminative hierarchical features for object recognition.” Signal Processing Letters (SPL), IEEE 21, no. 9, 1159-1163, 2014.
 8. **Zhen Zuo**, and Gang Wang. “Recognizing trees at a distance with discriminative deep feature learning.” In Proceedings of the Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on, pp. 1-5. IEEE, 2013.

Bibliography

- [1] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *CVPR*, vol. 1, 2005, pp. 886–893.
- [3] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2002.
- [4] B. S. Manjunath and W. Y. Ma, “Texture features for browsing and retrieval of image data,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1996.
- [5] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [6] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, 1995.
- [7] Q. V. Le *et al.*, “Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis,” in *CVPR*, 2011.

BIBLIOGRAPHY

- [8] W. Y. Zou *et al.*, “Deep learning of invariant features via simulated fixations in video,” in *NIPS*, 2012, pp. 3212–3220.
- [9] J. Deng *et al.*, “What does classifying more than 10,000 image categories tell us?” in *ECCV*, 2010.
- [10] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.
- [11] A. Oliva and A. Torralba, “Building the gist of a scene: The role of global image features in recognition,” *Progress in brain research*, vol. 155, pp. 23–36, 2006.
- [12] G. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [13] H. Lee *et al.*, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *ICML*, 2009, pp. 609–616.
- [14] V. Nair and G. E. Hinton, “3d object recognition with deep belief nets,” in *NIPS*, 2009.
- [15] H. Lee, C. Ekanadham, and A. Y. Ng, “Sparse deep belief net model for visual area v2,” in *NIPS*, 2008.
- [16] R. Salakhutdinov and G. E. Hinton, “Deep boltzmann machines,” in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

BIBLIOGRAPHY

- [17] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [18] Y. Bengio *et al.*, “Greedy layer-wise training of deep networks,” in *NIPS*, 2007.
- [19] Q. V. Le *et al.*, “Building high-level features using large scale unsupervised learning,” in *ICML*, 2012.
- [20] X. Yan *et al.*, “Modeling video dynamics with deep dynencoder,” in *ECCV*, 2014.
- [21] J. Zhang *et al.*, “Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment,” in *ECCV*, 2014.
- [22] Y. Bengio *et al.*, “Generalized denoising auto-encoders as generative models,” in *NIPS*, 2013.
- [23] Q. V. Le *et al.*, “Ica with reconstruction cost for efficient overcomplete feature learning,” in *NIPS*, 2011, pp. 1017–1025.
- [24] C.-Y. Lee *et al.*, “Deeply-supervised nets,” *arXiv preprint arXiv:1409.5185*, 2014.
- [25] K. Chatfield *et al.*, “Return of the devil in the details: Delving deep into convolutional nets,” in *BMVC*, 2014.
- [26] C. Szegedy *et al.*, “Going deeper with convolutions,” *arXiv preprint arXiv:1409.4842*, 2014.
- [27] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

BIBLIOGRAPHY

- [28] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *CVPR*, 2012.
- [29] K. He *et al.*, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in *ECCV*, 2014.
- [30] R. Girshick *et al.*, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014.
- [31] P. Sermanet *et al.*, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” *arXiv preprint arXiv:1312.6229*, 2013.
- [32] W. Ouyang *et al.*, “Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection,” *arXiv preprint arXiv:1409.3505*, 2014.
- [33] Y. Taigman *et al.*, “Deepface: Closing the gap to human-level performance in face verification,” in *CVPR*, 2014.
- [34] Y. Sun, X. Wang, and X. Tang, “Deep learning face representation from predicting 10,000 classes,” in *CVPR*, 2014.
- [35] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
- [36] T. Mikolov *et al.*, “Recurrent neural network based language model.” in *INTERSPEECH*, 2010.
- [37] I. Sutskever, “Training recurrent neural networks,” Ph.D. dissertation, University of Toronto, 2013.

BIBLIOGRAPHY

- [38] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *ICML*, 2014.
- [39] X. Chen and C. L. Zitnick, “Learning a recurrent visual representation for image caption generation,” *arXiv preprint arXiv:1411.5654*, 2014.
- [40] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” *arXiv preprint arXiv:1412.2306*, 2014.
- [41] S. Venugopalan *et al.*, “Translating videos to natural language using deep recurrent neural networks,” *arXiv preprint arXiv:1412.4729*, 2014.
- [42] R. Gens and P. Domingos, “Discriminative learning of sum-product networks,” in *NIPS*, 2012.
- [43] A. Hyvärinen, J. Hurri, and P. Hoyer, *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*. Springer, 2009.
- [44] T. Mikolov, “Statistical language models based on neural networks,” Ph.D. dissertation, Brno University of Technology, 2012.
- [45] J. Koutník *et al.*, “A clockwork rnn,” in *ICML*, 2014.
- [46] I. Sutskever, G. E. Hinton, and G. W. Taylor, “The recurrent temporal restricted boltzmann machine,” in *NIPS*, 2009.
- [47] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription,” in *ICML*, 2012.

BIBLIOGRAPHY

- [48] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the “echo state network” approach.* GMD-Forschungszentrum Informationstechnik, 2002.
- [49] D. Eigen *et al.*, “Understanding deep architectures using a recursive convolutional network,” *arXiv preprint arXiv:1312.1847*, 2013.
- [50] L. Fei-Fei, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories,” *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.
- [51] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *CVPR*, vol. 2, 2006, pp. 2169–2178.
- [52] A. Quattoni and A. Torralba, “Recognizing indoor scenes,” in *CVPR*, 2009.
- [53] J. Wang *et al.*, “Locality-constrained linear coding for image classification,” in *CVPR*, 2010, pp. 3360–3367.
- [54] J. Deng *et al.*, “Imagenet: A large-scale hierarchical image database,” in *CVPR*, 2009.
- [55] B. Zhou *et al.*, “Learning deep features for scene recognition using places database,” in *NIPS*, 2014.
- [56] J. Xiao *et al.*, “Sun database: Large-scale scene recognition from abbey to zoo,” in *CVPR*, 2010.

BIBLIOGRAPHY

- [57] B.-C. Kuo, D. Landgrebe *et al.*, “Nonparametric weighted feature extraction for classification,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 42, no. 5, pp. 1096–1105, 2004.
- [58] O. Boiman, E. Shechtman, and M. Irani, “In defense of nearest-neighbor based image classification,” in *CVPR*, 2008, pp. 1–8.
- [59] H. Zhang *et al.*, “Svm-knn: Discriminative nearest neighbor classification for visual category recognition,” in *CVPR*, 2006.
- [60] Y. Luo *et al.*, “Manifold regularized multitask learning for semi-supervised multilabel image classification,” *Image Processing, IEEE Transactions on*, vol. 22, no. 2, pp. 523–536, 2013.
- [61] D. Dai and L. Van Gool, “Ensemble projection for semi-supervised image classification,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2072–2079.
- [62] J. Weston *et al.*, “Deep learning via semi-supervised embedding,” in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 639–655.
- [63] H. Liao, E. McDermott, and A. Senior, “Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription,” in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 368–373.
- [64] P. F. Felzenszwalb *et al.*, “Object detection with discriminatively trained part-based models,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

BIBLIOGRAPHY

- [65] A. Frome, Y. Singer, and J. Malik, “Image retrieval and classification using local distance functions,” in *NIPS*, 2007.
- [66] A. Frome *et al.*, “Learning globally-consistent local distance functions for shape-based image retrieval and classification,” in *ICCV*, 2007.
- [67] S. McCann and D. G. Lowe, “Local naive bayes nearest neighbor for image classification,” in *CVPR*, 2012, pp. 3650–3656.
- [68] Z. Wang, Y. Hu, and L. T. Chia, “Image-to-class distance metric learning for image classification,” in *ECCV*, 2010.
- [69] T. Tuytelaars *et al.*, “The nbnn kernel,” in *ICCV*, 2011, pp. 1824–1831.
- [70] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *VISSAPP*, 2009, pp. 331–340.
- [71] A. Coates, H. Lee, and A. Y. Ng, “An analysis of single-layer networks in unsupervised feature learning,” in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 215–223.
- [72] K. Yu, Y. Lin, and J. Lafferty, “Learning image representations from the pixel level via hierarchical sparse coding,” in *CVPR*, 2011, pp. 1713–1720.
- [73] K. Sohn *et al.*, “Efficient learning of sparse, distributed, convolutional feature representations for object recognition,” in *ICCV*, 2011, pp. 2643–2650.
- [74] R. Gens and P. Domingos, “Discriminative learning of sum-product networks,” in *NIPS*, 2012, pp. 3248–3256.
- [75] J. Donahue *et al.*, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *ICML*, 2014.

BIBLIOGRAPHY

- [76] Z. Jiang, Z. Lin, and L. S. Davis, “Learning a discriminative dictionary for sparse coding via label consistent k-svd,” in *CVPR*, 2011.
- [77] J. Mairal *et al.*, “Supervised dictionary learning,” in *NIPS*, 2008.
- [78] M. Yang *et al.*, “Fisher discrimination dictionary learning for sparse representation,” in *ICCV*, 2011.
- [79] S. Kong and D. Wang, “A dictionary learning approach for classification: separating the particularity and the commonality,” in *ECCV*, 2012.
- [80] Q. Li, J. Wu, and Z. Tu, “Harvesting mid-level visual concepts from large-scale internet images,” in *CVPR*, 2013.
- [81] C. Doersch, A. Gupta, and A. A. Efros, “Mid-level visual element discovery as discriminative mode seeking,” in *NIPS*, 2013.
- [82] J. Sun, J. Ponce *et al.*, “Learning discriminative part detectors for image classification and cosegmentation,” in *ICCV*, 2013.
- [83] M. Juneja *et al.*, “Blocks that shout: Distinctive parts for scene classification,” in *CVPR*, 2013.
- [84] B. Yao and L. Fei-Fei, “Action recognition with exemplar based 2.5 d graph matching,” in *ECCV*, 2012.
- [85] T. Malisiewicz, A. Gupta, and A. A. Efros, “Ensemble of exemplar-svms for object detection and beyond,” in *ICCV*, 2011.
- [86] S. Gao, I. W.-H. Tsang, and Y. Ma, “Learning category-specific dictionary and shared dictionary for fine-grained image categorization,” *Image Processing, IEEE Transactions on*, vol. 23, no. 2, pp. 623–634, 2014.

BIBLIOGRAPHY

- [87] H. O. Song *et al.*, “Sparselet models for efficient multiclass object detection,” in *ECCV*, 2012.
- [88] H. O. Song, T. Darrell, and R. B. Girshick, “Discriminatively activated sparselets,” in *ICML*, 2013.
- [89] Z. Wang *et al.*, “Linear distance coding for image classification,” *IEEE Trans. Image Processing*, vol. 22, no. 2, pp. 537–548, 2013.
- [90] S. Singh, A. Gupta, and A. A. Efros, “Unsupervised discovery of mid-level discriminative patches,” in *ECCV*, 2012.
- [91] G. Wang, D. Hoiem, and D. Forsyth, “Learning image similarity from flickr groups using fast kernel machines,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2177–2188, 2012.
- [92] L.-J. Li and L. Fei-Fei, “What, where and who? classifying events by scene and object recognition,” in *ICCV*, 2007.
- [93] J. Wu and J. M. Rehg, “Centrist: A visual descriptor for scene categorization,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1489–1501, 2011.
- [94] R. Margolin, L. Zelnik-Manor, and A. Tal, “Otc: A novel local descriptor for scene classification,” in *ECCV*, 2014.
- [95] R. Girshick *et al.*, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *arXiv preprint arXiv:1311.2524*, 2013.
- [96] M. Pandey and S. Lazebnik, “Scene recognition and weakly supervised object localization with deformable part-based models,” in *ICCV*, 2011.

BIBLIOGRAPHY

- [97] L.-J. Li *et al.*, “Object bank: A high-level image representation for scene classification & semantic feature sparsification,” in *NIPS*, 2010.
- [98] Y. L. Boureau, Y. L. F. Bach, and J. Ponce, “Learning mid-level features for recognition,” in *CVPR*, 2010.
- [99] X. Li and Y. Guo, “Latent semantic representation learning for scene classification,” in *ICML*, 2014.
- [100] X. Wang *et al.*, “Max-margin multiple-instance dictionary learning,” in *ICML*, 2013.
- [101] S. Gao, I.-H. Tsang, and L.-T. Chia, “Laplacian sparse coding, hypergraph laplacian sparse coding, and applications,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 92–104, 2013.
- [102] D. Lin *et al.*, “Learning important spatial pooling regions for scene classification,” in *CVPR*, 2014.
- [103] Y. Gong *et al.*, “Multi-scale orderless pooling of deep convolutional activation features,” in *ECCV*, 2014.
- [104] X. Zeng *et al.*, “Deep learning of scene-specific classifier for pedestrian detection,” in *ECCV*, 2014.
- [105] A. Graves and J. Schmidhuber, “Offline handwriting recognition with multidimensional recurrent neural networks,” in *NIPS*, 2009.
- [106] P. Pinheiro and R. Collobert, “Recurrent convolutional neural networks for scene labeling,” in *ICML*, 2014.

BIBLIOGRAPHY

- [107] J. T. Rolfe and Y. LeCun, “Discriminative recurrent sparse auto-encoders,” *arXiv preprint arXiv:1301.3775*, 2013.
- [108] K. Gregor *et al.*, “Draw: A recurrent neural network for image generation,” *arXiv preprint arXiv:1502.04623*, 2015.
- [109] V. Mnih *et al.*, “Recurrent models of visual attention,” in *NIPS*, 2014, pp. 2204–2212.
- [110] J. Donahue *et al.*, “Long-term recurrent convolutional networks for visual recognition and description,” *arXiv preprint arXiv:1411.4389*, 2014.
- [111] J. Y.-H. Ng *et al.*, “Beyond short snippets: Deep networks for video classification,” *arXiv preprint arXiv:1503.08909*, 2015.
- [112] J. Mao *et al.*, “Deep captioning with multimodal recurrent neural networks (m-rnn),” *arXiv preprint arXiv:1412.6632*, 2014.
- [113] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional neural networks,” *arXiv preprint arXiv:1311.2901*, 2013.
- [114] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [115] Y. Jia *et al.*, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.
- [116] B. Zhou *et al.*, “Learning deep features for scene recognition using places database,” in *NIPS*, 2014.

BIBLIOGRAPHY

- [117] J. Sánchez *et al.*, “Image classification with the fisher vector: Theory and practice,” *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.
- [118] M. Lapin, B. Schiele, and M. Hein, “Scalable multitask representation learning for scene classification,” in *CVPR*, 2014.
- [119] K. Wang *et al.*, “3d human activity recognition with reconfigurable convolutional neural networks,” in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 97–106.