The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Hierarchical Architectures and Learning Algorithms for Multi-Label Image Classification and Scene Categorization

Zenghai Chen

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

August 2013

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Zenghai Chen
_____ (Name of student)

Abstract

This thesis presents hierarchical architectures and learning algorithms for multi-label image classification and scene categorization. Three main contributions are reported in the thesis. They include: (1) an adaptive recognition model based on neural networks for image annotation; (2) a hierarchical neural approach for multi-instance multi-label image classification; and (3) a hybrid holistic and object-based approach for scene categorization.

In the first investigation, we propose an adaptive recognition model based on neural networks for annotating images. The Adaptive Recognition Model (ARM) consists of an adaptive ClassiFication Network (CFN) and a nonlinear Correlation Network (CLN). The adaptive CFN aims to annotate an image with labels, and the CLN is used to unveil the correlative information of labels for annotation refinement. Image annotation is carried out by an ARM in two stages. In the first stage, the features extracted from regions of the input images are fed to a CFN to produce classification labels. In the second stage, the CLN uses label correlations learnt from the training images to refine the classification result. The ARM works in a forward-propagating manner, resulting in high efficiency in image annotation. Furthermore, the computational time of an ARM is insensitive to the number of regions of the input image and the vocabulary size. In this thesis, the effect of label correlation in image annotation is, comprehensively, studied on a real image dataset and a synthetic image dataset. The exploitation of a controllable synthetic dataset

helps to systematically study the function of label correlation and effectively analyze the performance of the ARM. Experimental results demonstrate the efficiency and effectiveness of the proposed ARM.

In the second investigation, we propose a multi-instance multi-label algorithm based on hierarchical neural networks for image classification. Image annotation can be regarded as a multi-instance multi-label image classification problem. But different from the first investigation that requires regional ground truth for model training, the model proposed in the second investigation can be trained using the image ground truth only. In particular, the proposed model, termed Multi-Instance Multi-Label Neural Network (MIMLNN), consists of two stages of MultiLayer Perceptrons (MLPs). For multi-instance multi-label image classification, all the regional features are fed to the first-stage MLPs, with one MLP copy processing one image region. After that, the MLP in the second stage incorporates the outputs of the first-stage MLPs to produce the final labels for the input image. The first-stage MLP is expected to model the relationship between regions and labels, while the second-stage MLP aims at capturing the label correlation for classification refinement. The classical error Back-Propagation (BP) approach is adopted to tune the parameters of MIMLNN. In view of that the traditional gradient descent algorithm suffers from the long-term dependency problem, a refined BP algorithm named Rprop is extended to effectively train MIMLNN. Experiments are conducted on a synthetic dataset and the widely-used Corel dataset. Experimental results demonstrate the superior performance of MIMLNN by comparing with state-of-the-art algorithms for multi-instance

multi-label image classification.

In the third investigation, we target at scene categorization (termed scene classification as well). We first employ a deep learning algorithm of a hierarchical architecture to classify scenes, and show that the deep learning algorithm is a promising holistic approach for scene classification. After that, we propose a hybrid holistic and object-based approach for scene classification. In particular, if the decisions made by holistic and object-based approaches are identical, the scene class agreed by them is selected as the final decision. Otherwise, a majority voting scheme is employed to make the final decision based on the results of all the classifiers of both holistic and object-based approaches.

At the end of this thesis, a conclusion is drawn and three future research directions are pointed out. The three directions all concentrate on deep neural networks. In particular, the first direction is to explore an efficient image classification network based on MIMLNN using deep neural networks. In the second direction, we would like to label each image pixel, and then explore deep neural networks to obtain the scene class by learning from all the pixel labels. In the third direction, we would like to perform scene parsing using deep neural networks.

List of Publications

Journal Papers:

1. **Z. Chen**, H. Fu, Z. Chi, and D. Feng, "An Adaptive Recognition Model for Image Annotation," *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1120-1127, 2012.

2. **Z. Chen**, Z. Chi, H. Fu, and D. Feng, "Multi-Instance Multi-Label Image Classification: A Neural Approach," *Neurocomputing*, vol. 99, pp. 298-306, 2013.

3. **Z. Chen**, B. Xu, Z. Chi, and D. Feng, "Mathematical Formulation of Knitted Fabric Spirality Using Genetic Programming," *Textile Research Journal*, vol. 82, no. 7, pp. 667-676, 2012.

4. **Z. Chen**, Jing Wu, and Z. Chi, "A Mutation Particle Swarm Optimization Algorithm for Multilayer Perceptron Training with Applications,"*International Journal of Information Acquisition*, vol. 9, no. 1, pp. 1350003-1 -1350003-8, 2013.

Conference Papers:

1. **Z. Chen**, Z. Chi, H. Fu, and D. Feng, "Combining Holistic and Object-Based Approaches for Scene Classification," *Proceedings of the 5th International Symposium on Computational Intelligence and Design* (*ISCID 2012*), pp. 65-68, 28-29 October 2012, Hangzhou, China.

2. **Z. Chen**, H. Fu, Z. Chi, and D. Feng, "A Neural Network Model with Adaptive Structure for Image Annotation," *Proceedings of the 11th International Conference on Control, Automation, Robotics and Vision* (*ICARCV 2010*), pp. 1865-1870, 7-10 December 2010, Singapore.

Acknowledgements

I am indebted to a number of individuals who, directly or indirectly, contributed to my research work and this thesis. In particular, I would like to express my most sincere thanks and gratitude to Dr. Zheru Chi, my supervisor, for his outstanding and constructive guidance in my PhD study and research. I am very thankful to Prof. Dagan Feng, my co-supervisor, for his valuable suggestions. I wish to sincerely thank Dr Hong Fu, another co-supervisor, for her helpful suggestions and generous assistance. I deeply appreciate their valuable help and support in my research, study and life.

I would also like to thank Mr. Chi Xu, Mr. Xiaoyu Zhao, Dr. Zhen Liang, my friends, for their selfless assistance.

Last but not the least, I would like to express my great gratitude and love to my parents for their upbringing, self-giving and constant support. They are my irreplaceable spiritual mentors. Special love is expressed to my father, who passed away in 2011, for his being with me in my first twenty-five years. His words are carved in my heart, and his spirit will be with me, till the end of my life.

List of Abbreviations

ARM:	Adaptive Recognition Model
AUC:	Area Under Curve
BP:	Back-Propagation
CAN:	Concept Association Network
CD:	Contrastive Divergence
CDBN:	Convolutional Deep Belief Networks
CENTRIST:	CENsus TRansform hISTogram
CFN:	ClassiFication Network
CLN:	CorreLation Network
CMRM:	Cross-Media Relevance Model
CRM:	Continuous-space Relevance Model
CRMM:	Convolutional-Recursive Neural Networks
CT:	Census Transform
DBN:	Deep Belief Networks
DN:	Deconvolutional Networks
FP:	False Positive
FPR:	False Positive Rate
GD:	Gradient Descent
GL:	Graph Learning
HMM:	Hidden Markov Model
HMP:	Hierarchical Matching Pursuit
LSA:	Latent Semantic Analysis

MBRM:	Multiple Bernoulli Relevance Models
MIL:	Multi-Instance Learning
MLL:	Multi-Label Learning
MIMLL:	Multi-Instance Multi-Label Learning
MIMLNN:	Multi-Instance Multi-Label Neural Network
MLP:	Multi-Layer Perceptron
OMP:	Orthogonal Matching Pursuit
PLSA:	Probabilistic Latent Semantic Analysis
PR:	Precision-Recall
RBM:	Restricted Boltzmann Machine
ROC:	Receiver Operating Characteristic
SIFT:	Scale-Invariant Feature Transform
SISLL:	Single-Instance Single-Label Learning
SPM:	Spatial Pyramid Matching
SVD:	Singular Value Decomposition
SVM:	Support Vector Machine
TM:	Translation Model
TP:	True Positive
TPR:	True Positive Rate

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

With the prevalence of digital imaging devices such as digital cameras and mobile phones, a large number of images are produced every day. An emerging issue is how to efficiently and effectively search required image items from a huge image database. To achieve that, the images should be accurately classified into various categories, such that they can be well organized and retrieved. Multi-label image classification and scene categorization are two fundamental and challenging topics of computer vision. Although many successes of the two topics have been achieved, the performance is still far from satisfactory.

Multi-label image classification is to find out multiple objects in an image and then assign corresponding object labels to the image. It is more challenging than traditional single-label image classification, since the contents of multi-label image classification are much more complex. Image annotation is a classical multi-label image classification problem. Image annotation aims to find out mappings between low-level visual features and high-level labels/concepts from labeled images, and then annotate unlabeled images based on the learnt mappings. Apart from direct mappings between visual features and labels, label correlation, reflecting co-occurrence relationships among different labels, has also been adopted by many researchers to refine image annotation performance. However, many existing label correlation

modeling methods characterize linear, symmetric, and positive correlation only. Some can characterize nonlinear but symmetric correlation only; while others may characterize asymmetric but positive correlation only. We consider that in practice label correlation can be nonlinear, asymmetric, both positive and negative. The co-occurrence relationships between labels can be very complicated. In addition, little existing work systematically investigates the effect of label correlation. Furthermore, most of previous correlation models are designed according to experts' experiences. Those models are good for some problems, but they may not work well for some other tasks. It is thus meaningful to explore models that can automatically learn label correlation from training images. With the development of deep learning, neural networks recapture significant attention in recent years. Therefore, in this thesis, we propose neural network methods to automatically learn label correlation for refining multi-label image classification. In particular, we first propose an Adaptive Recognition Model (ARM) for image annotation. The model consists of two stages of neural networks. The first-stage neural networks learn the mappings between the visual features and labels of image regions; while the neural network in the second stage learns label correlations from training images. The correlation values between different labels can be explicitly known from the weights of the neural network. This helps the analysis of label correlation. The main disadvantage of an ARM is that it needs regional ground truth for training. In light of that, we then propose another neural network model based on ARM. The new model, termed Multi-Instance Multi-Label Neural Network (MIMLNN) can automatically learn the mappings

between visual features and labels of regions, as well as label correlation, just based on regional visual features and image ground truth. We do not need to provide regional ground truth for training MIMLNN.

Scene categorization is challenging because different scenes may consist of similar objects and thus possess similar visual features. For example, both *coasts* and *lakes* have water and rocks. Sometimes it is even difficult for human beings to accurately distinguish these two scenes, not to mention a computer. Scene categorization is termed scene classification as well. In the rest of this thesis, we use both scene categorization and scene classification alternatively. There are two main strategies for scene classification: holistic and object-based. A holistic strategy characterizes a scene image using global visual features only, without considering the regional object attributes. By contrast, an object-based strategy represents a scene image by modeling the object attributes. Both of them have advantages and disadvantages. When the scene contents are simple, holistic strategy performs well. If the scene contents are complex and contain multiple objects, the holistic strategy may not be effective. By contrast, when the scenes consist of different objects, the object-based strategy is advantageous. However, if the scenes are of simple contents, the object-based strategy may be inefficient. Furthermore, the accuracy of regional object recognition significantly influences the final classification performance of an object-based strategy. In view of this, in this thesis, we propose to combine holistic and object-based strategies for scene classification. To our best knowledge, very little work investigates the combination of holistic and object-based strategies.

Deep learning is a very hot topic of machine learning in recent years, since it has been demonstrated that deep or hierarchical architectures can represent higher level and more abstract features than shallow architectures. Therefore, our proposed methods concentrate on hierarchical architectures in this thesis. The proposed two methods for multi-label image classification, and the methods for scene categorization are all of hierarchical architectures.

1.2 Statements of Originality

This thesis presents hierarchical architectures and learning algorithms for multi-label image classification and scene categorization. The work described in this thesis was carried out at the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, between August 2009 and August 2013, under the supervision of Dr. Zheru Chi, Prof. Dagan Feng, and Dr. Hong Fu.

The thesis consists of six chapters and two appendixes. The work described in this thesis was originated by the author except where acknowledged and referenced, or where the results are widely known. The following states the original contributions: (1) An adaptive recognition model based on neural networks for image annotation is the work of the author. In this investigation, we propose a two-stage neural network model. The neural networks in the first stage establish the relationships between regional visual features and labels. The neural network in the second stage learns the label correlation from training image to refine the results from the first-stage neural networks. A proposed synthetic image dataset and a real image dataset are utilized for

the experiments to evaluate our proposed model.

(2) A hierarchical neural approach for multi-instance multi-label image classification is the work of the author. In this investigation, we propose a hierarchical neural model to automatically learn the mappings between regional visual features and labels, as well as the label correlation just using regional visual features and the image ground truth of training images. A proposed synthetic image dataset together with the widely-used Corel image dataset are used for the experiments to evaluate the effectiveness of our proposed approach.

(3) Combining holistic and object-based approaches for scene classification is the work of the author. In this investigation, we first employ a state-of-the-art deep learning method for recognizing scenes, and show that deep learning is a promising holistic approach for scene classification. We then propose a hybrid holistic and object-based approach to effectively classify scenes.

1.3 Outline of the Thesis

The thesis consists of six chapters and two appendixes. The thesis is outlined as follows.

Chapter 2 introduces the basic principles and definitions of image annotation, multi-instance multi-label image classification, scene classification, deep learning; and reviews important developments achieved in these areas.

In Chapter 3, we propose an Adaptive Recognition Model (ARM) based on neural networks to annotate images. The proposed model ARM consists of a ClassiFication

Network (CFN) and a CorreLation Network (CLN). The annotation is carried out by ARM in two stages. Firstly, an input image is segmented into a number of regions. The visual features extracted from the regions are fed to the classifier of the CFN. Secondly, the classification result is refined in the CLN using the label correlation to generate the final annotation for the input image. Besides a real image dataset, a synthetic image dataset is constructed for experiments. Based on the proposed synthetic dataset and a real dataset, we systematically analyze the effect of label correlation in image annotation.

In Chapter 4, in view of that the model ARM proposed in Chapter 3 requires regional ground truth for training, we extend the architecture of ARM and propose a training method that does not utilize the regional ground truth. Given the regional visual features and the image ground truth of training images, the proposed model, termed Multi-Instance Multi-Label Neural Network (MIMLNN), can automatically learn the mappings between regional visual features and labels, as well as the label correlation. The classical error Back-Propagation (BP) method is employed to train MIMLNN. Considering that gradient descent, a classical BP algorithm, suffers from the long-term dependency problem, an advanced BP algorithm Rprop is extended to train MIMLNN. The experiments are conducted on a synthetic image dataset and the popular Corel image dataset.

In Chapter 5, we investigate a combination of holistic and object-based approaches for scene categorization. We first employ a state-of-the-art deep learning method to learn feature representations of natural scenes to perform classification. We

show that a deep learning method is a promising holistic approach for scene classification. Then, we propose to combine holistic and object-based approaches for scene classification. We report that the performance of our combined approach is better than those of individual methods.

Chapter 6 concludes this thesis with final remarks and discusses some possible directions of future research.

Appendix A tabulates the names and entry numbers in the label vector of all the 54 synthetic labels used in Chapter 3.

Appendix B describes the implementation details of the Rprop algorithm used in Chapter 4.

Chapter 2 Literature Review

In this chapter, the basic principles and definitions of multi-label learning, image annotation, scene classification, and deep learning are introduced; and recent developments in these areas are reviewed and discussed.

2.1 Multi-Label Learning and Image Annotation

2.1.1 Multi-Label Learning

Traditional supervised learning concentrates on Single-Instance Single-Label Learning (SISLL); that is, an object is represented by an instance and associated with only one label. SISLL is successful, but not practical, since a real-world object may be represented by multiple instances and associated with multiple label. Therefore, multi-instance learning (MIL) (Chen et al., 2006; Chen and Wang, 2004; Feng et al., 2011) and multi-label learning (MLL) (Boutell et al., 2004; Kang et al., 2006; Xu, 2011) have been proposed. For the task of image classification, in MIL, an image is a bag of instances, with each instance representing an image region. As informative regional features are utilized, this approach characterizes images with complex contents pretty well. However, MIL targets at binary classification. It assigns an image with one label only. By contrast, MLL deals with multi-label tasks, and it is able to associate an image with multiple labels. But MLL regards an image as a single instance. The regional features are not employed. The issue is that different labels of an image usually relate to different regions. The use of global features is not powerful

to discriminate the image labels. Therefore, some researchers have developed multi-instance multi-label learning (MIMLL) based on MIL and/or MLL for multi-label image classification (Zhou and Zhang, 2007; Zha et al., 2008; Zhang and Wang, 2009).

For MIMLL image classification, each image is comprised of a bag of regions and associated with multiple labels. In the learning phase, the relationship between the image regions and labels is unknown. The aim of learning is to figure out the relationship between the regions and labels from training images, and then the learnt relationship can be used to classify unlabeled images. Zhou et al. (Zhou and Zhang, 2007) proposed MIMLBoost and MIMLSVM for multi-instance multi-label scene classification. MIMLBoost transforms an MIML learning task into an MIL problem and solve the MIL problem by using MIBoost (Xu and Frank, 2004), while MIMLSVM transforms an MIML learning task into an MLL problem and tackle the MLL problem by adopting MLSVM (Boutell et al., 2004). A drawback of MIMLBoost and MIMLSVM is that both of them do not take label correlation into account.

2.1.2 Image Annotation

2.1.2.1 Definition of Image Annotation

Image annotation is an MLL problem. When an image is partitioned into and represented by a bag of regions, image annotation becomes an MIMLL problem. Image annotation is a process that employs various machine learning algorithms to

find out the mappings between low-level visual features and high-level object labels from labeled images, and then propagates the labels to unlabeled images based on the learnt mappings. Fundamental visual features include color, shape and texture. High-level concepts are the labels that are used to describe image contents. For example, an image that contains a tiger can be annotated with label “tiger”.

2.1.2.2 Label Correlation

Apart from direct mappings between visual features and labels, the use of label correlation for annotation improvement is another research topic in image annotation. Label correlation is a type of relationship that reflects the frequency of co-occurrence of annotated labels. If two labels are frequently annotated to the same images, then they have strong mutual correlation. If the label correlation of testing images is similar with that of training images, then the use of correlative information of labels extracted from training images would benefit the annotation.

2.1.2.3 Important Developments

Various machine learning approaches have been employed to handle the annotation task. These approaches include the translation model (Duygulu et al., 2002), relevance model (Jeon et al, 2003; Lavrenko et al., 2004; Feng et al., 2004), hidden Markov model (Ghoshal et al., 2005; Li and Wang, 2003), conditional random field (Li and Sun, 2006; Wang and Gong, 2007), graph model (Liu et al., 2009; Tang et al., 2010; Jamieson et al., 2010), neural networks (Fu et al., 2010), support vector

machine (SVM) (Qi and Han, 2007), hypothetical local mapping (Li and Wang, 2008), etc.

An early important work is a translation model (TM) proposed by Duygulu et al. (Duygulu et al., 2002), in which image annotation is treated as a task of translation. Duygulu et al. represented images in blobs and then employed a translation model to translate the blobs into a set of semantic labels. Jeon et al. proposed a cross-media relevance model (CMRM) for image annotation (Jeon et al, 2003). Differing from translating labels to blobs of TM, CMRM takes advantage of the joint probability of labels and blobs to predict labels for unlabeled images. The performance of CMRM is superior to that of TM. Based on CMRM, Lavrenko et al. developed a continuous-space relevance model (CRM) (Lavrenko et al., 2004). CRM utilizes continuous-valued features rather than discrete features used in CMRM for region description. Experimental results demonstrate that CRM performs much better than CMRM. In addition to CRM, Feng et al. proposed another relevance model named multiple Bernoulli relevance models (MBRM) to model the joint probability of semantic words/labels and visual features (Feng et al., 2004). MBRM uses a multiple Bernoulli model to estimated word probabilities and a non-parametric kernel density estimate to compute the image feature probabilities. Compared with CRM that models words using a multinomial distribution, the multiple Bernoulli model makes MBRM applicable to various lengths of annotations. As a result, MBRM outperforms CRM.

Latent semantic analysis (LSA), which is originally designed for text retrieval, is proposed for image annotation as well (Monay and Gatica-Perez, 2003). Monay and

Gatica-Perez treated an image as a document, each label assigned to the image as a word of the document. The images were first represented with visual features. Then latent variables were introduced to link image features with words so as to capture co-occurrence information of images and labels. In a later work, Monay and Gatica-Perez further refined the annotation performance using probabilistic latent semantic analysis (PLSA), which was regarded as a probabilistic version of LSA (Monay and Gatica-Perez, 2004).

Ghoshal et al. developed a generative stochastic model named hidden Markov model (HMM) for image annotation (Ghoshal et al., 2005). In that work, images are modeled as having been stochastically generated by HMM whose states represent concepts, and annotated with the a posteriori probability of concepts presented in them. Being different from sole use of HMM (Ghoshal et al., 2005), Zhao et al. combined HMM and Support Vector Machine (SVM) to solve annotation problems (Zhao et al., 2009). As the combined model takes advantages of both discriminative classification and generative model, the performance is superior to that of HMM. Li and Wang proposed a Two-dimensional Multi-resolution Hidden Markov Models (2D MHMMs) to couple images and concepts (Li and Wang, 2003). They defined each category of images as a concept represented by a 2D MHMM. The 2D MHMM associated to each image category is used to extract representative information for the category. Conditional random field (CRF), being a discriminative alternative to HMM, is another important attempt to image annotation (Li and Sun, 2006 ; Wang and Gong, 2007).

Graph model is another powerful tool to handle annotation problems. Liu et al. proposed an image-based graph learning and a word-based graph learning for image annotation (Liu et al., 2009). The image-based graph learning is performed to obtain candidate annotations, while the word-based graph learning is developed to refine the annotations based upon word correlation.

Fu et al. proposed a concept association network (CAN) based on neural networks for image annotation (Fu et al., 2010). CAN extracts label correlation using a linear system. Since label correlation is taken into account and advanced attention-driven techniques (Fu et al., 2006; Fu et al., 2009) are adopted for image segmentation, promising performance is achieved by CAN.

In general, most of above approaches use a region-based annotation scheme, which first segments images into regions, and then extracts regional features for annotation. Besides a mapping from low-level visual features to high-level labels, label correlation is also employed to refine the annotation performance. However, the effect of label correlation has not yet been systematically investigated. Furthermore, those region-based approaches usually associate each segmented region with one label only, which may not be effective in practice since a region sometimes can be annotated with multiple labels. Our proposed image annotation model is motivated by CAN. It is hence worth giving a more detailed description of CAN at the end of this section.

As shown in Figure 2-1, CAN is of a three-layer structure: feature stimulus, visual classifiers, and concept nodes (Fu et al., 2010). The annotation is conducted by

a CAN in three stages. An image is first segmented into regions, from which visual features are extracted as feature stimulus. The feature stimulus is, then, presented to the visual classifiers, which produce the outputs for corresponding concept nodes. In the concept node layer, the image is finally annotated by considering both the information from the lower layers and the outputs from other concept nodes. The number of classifiers used is equal to the number of concepts. For the dataset used by Fu et al. (Fu et al., 2010), there are in total 98 labels, resulting in that 98 classifiers need to be trained. Obviously, it is time consuming and may not be practical, as practical annotation vocabulary may have hundreds of or even thousands of labels. Another problem of the CAN lies in its long annotation time. The annotation problem of the CAN is finally converted into finding out a node combination with the maximum response in the concept node layer. In order to exhaust all possible node combinations, the outputs of all the classifiers need to be obtained for each region first, and then top ten outputs of each region are selected to do the combinations. Suppose that an image has R regions and the annotation vocabulary has T ($T \geqslant 10$) labels. Then, $R \times T$ classifier outputs need to be obtained. After that, top ten outputs for each region are chosen for combination, and total 10^R combinations are computed to get the maximum concept combination as the final annotation. It can be seen that the annotation time is significantly dependent of the number of regions R and the vocabulary size T. Therefore, although promising performance has been achieved by CAN, it is necessary for us to develop a more effective and efficient annotation model for image annotation of a large image database.

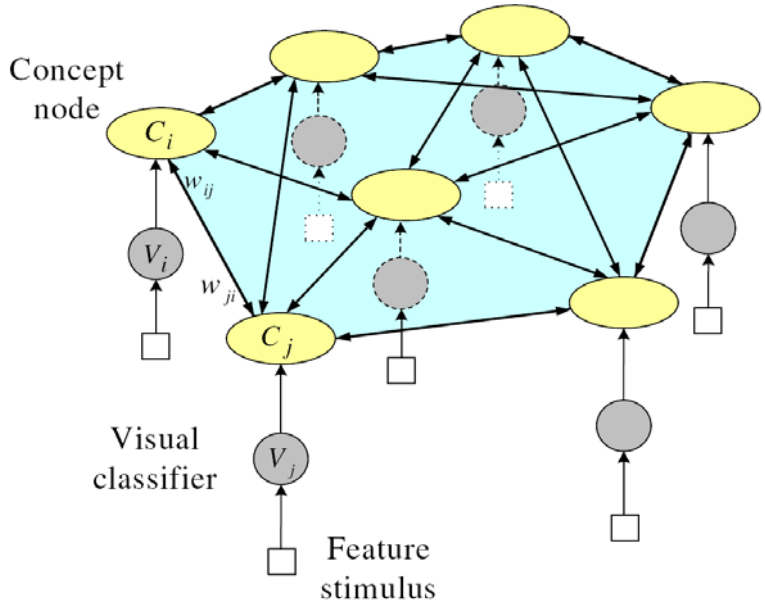


Figure 2-1. Architecture of concept association network (CAN) (Fu et al., 2010).

2.2 Scene Classification

2.2.1 Basic Definitions

Scene classification approaches can be grouped into two categories: holistic and object-based. A holistic strategy represents a scene image using global visual features; while an object-based strategy represents a scene image by modeling the object attributes. Figure 2-2 shows the procedures of a holistic strategy and an object-based strategy for scene classification. For the scene *beach*, the holistic strategy first extracts global visual features of the image, and then predict a scene class based on global features. By contrast, the object-based strategy first recognizes the objects, e.g., sky, water, people, and sand. The scene class is then predicted according the occurrences of the objects.

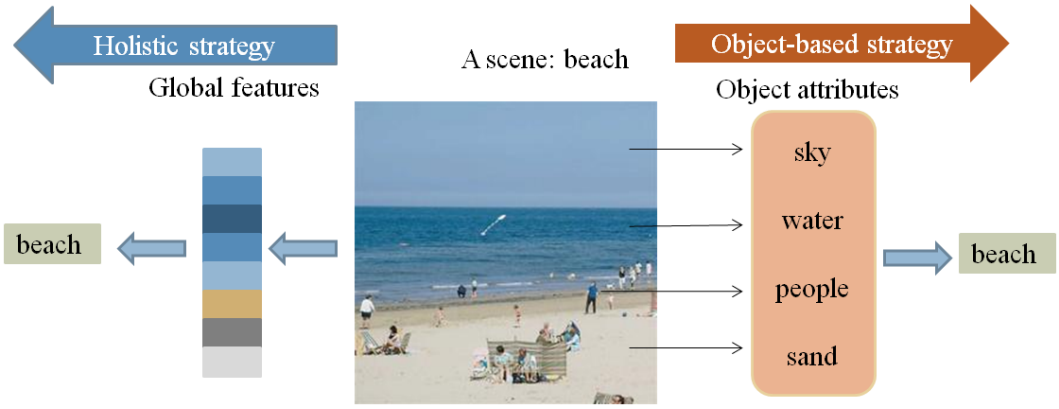


Figure 2-2. Procedures of a holistic strategy and an object-based strategy for scene classification.

2.2.2 Holistic Approaches

Many popular scene classification approaches are holistic, such as Bag-of-words (Fei-Fei and Perona, 2005), SIFT (Lowe, 1999), Gist (Oliva and Torralba, 2001), SPM (Lazebnik et al., 2011), CENTRIST (Wu and Rehg, 2011) etc.

The bag-of-words approach (Fei-Fei and Perona, 2005) characterizes the images using an intermediate representation. First of all, a dictionary of visual words is constructed using clustering techniques such as K-means based on image visual features. After that, the visual features of an image is projected to the dictionary to obtain a visual word vector as the representation of the image.

SIFT has four essential steps: keypoint detection, keypoint localization, orientation assignment and keypoint descriptor (Lowe, 1999). Keypoint detection is to detect interest points, which are called keypoints. To do that, an image is first convolved with a number of Gaussian filters at different scales. Then the differences of successive Gaussian-convolved images are computed. The maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales are then taken as the

keypoints. Keypoint detection produces many candidates, some of which may be unstable. The step of keypoint localization then tries to reject points that have a low contrast by performing a fitting to the data for accurate location, scale, and ratio of principal curvatures. After that, in the orientation assignment step, each keypoint is assigned one or more orientations based on local image gradient directions, such that the invariance to rotation can be achieved. In the final step, each keypoint found by previous steps is represented by a keypoint descriptor vector. A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location. These samples are then accumulated into 8-bin orientation histograms summarizing the contents over 4×4 subregions. At the end, a 128-dimension ($8\times4\times4$) descriptor vector is generated for each keypoint representation.

Gist represents a real world scene by a spatial envelope, which is a set of perceptual properties, namely naturalness, openness roughness, ruggedness and expansion (Oliva and Torralba, 2001). It has been shown that the spatial envelope properties are related to the shape of the scene, and are meaningful to human observers. The properties are estimated from two image spectral representations: the global energy spectrum and the spectrogram. The estimation of spatial envelope properties from the global energy spectrum is then solved using Discriminate Spectral Templates (DST); while the estimation from the spectrogram is solved using Windowed Discriminate Spectral Template (WDST). DST describes how each spectral component contributes to a spatial envelope property; while WDST describes

how the spectral components at different spatial locations contribute to a spatial envelope property. They together model the Gist features of a scene.

SPM characterizes image visual features using a number of SIFT descriptors (Lowe, 1999). Then a vocabulary of visual words is built based on the SIFT descriptors using K-means clustering. After that, each SIFT descriptor can be represented by the visual words. Finally, the visual words are concatenated through a spatial pyramid to represent the images.

CENTRIST represents an image using the histogram of Census Transform (CT) (Wu and Rehg, 2011). CT compares the intensity value of a pixel with its eight neighbor pixels. If a pixel is larger than or equal to one of its neighbors, a bit 1 is set in the corresponding neighbor location. Otherwise, a bit 0 is set. The eight bits are then collected to form a decimal number (CT value) between 0 and 255. The histogram of the CT values for all the pixels is taken as CENTRIST representation.

The above holistic representations perform well for scenes that have very few objects and simple contents. But they may not work well for scenes that contain multiple complex objects, since the holistic representations do not take into account object attributes.

2.2.3 Object-Based Approaches

To address the shortcoming of holistic approaches, some researchers propose to use an object-based strategy (Vogel and Schiele, 2007; Cheng and Wang, 2010; Luo et al., 2005; Serrano et al., 2004).

Vogel and Schiele (Vogel and Schiele, 2007) partitioned an image into regular 10×10 regions. A concept classifier then annotates each region with a concept. The area ratio occupied by each concept is then calculated to form a concept occurrence vector as the representation of the scene image. The work proposed by Vogel and Schiele (Vogel and Schiele, 2007) does not take into account the spatial relationships between the concepts. In light of that, Cheng and Wang proposed contextual Bayesian networks to perform semantic modeling of natural scenes (Cheng and Wang, 2010). Differing from that Vogel and Schiele partitioned an image using a regular grid, Cheng and Wang partitioned an image using automatic segmentation. The segmented regions are recognized by an object classifier. Based on the object recognition result, the spatial relationships between objects and concept occurrence are modeled by Bayesian networks to infer scene classes.

Object-based strategy is advantageous for scenes consisting of complex objects. If the scenes are of simple contents, however, object-based strategy may be inefficient. Furthermore, the accuracy of regional object recognition significantly influences the final classification performance.

2.3 Deep Learning and Feature Learning

Traditional feature extraction concentrates on feature engineering, which is to manually design feature representation algorithms according to experts' experiences. The hand-designed feature representations usually perform well for some specific tasks. But they may not work well for other tasks. Furthermore, the parameters of

hand-designed feature representations need to be manually tuned. However, how to choose appropriate parameters could be a tough job, even for an experienced engineer. Feature learning, therefore, attracts significant attention in recent years. Feature learning aims to automatically learn features from raw data like image pixels, rather than manually design features according to experts' experiences. It moves feature representation closer to human vision mechanism. Feature learning (also termed unsupervised feature learning), however, did not achieve many progresses until 2006, when deep learning started to boom (Bengio, 2009; Krizhevsky et al., 2012; Farabet et al., 2013). Very little research concentrates on deep learning before 2006, as deep architectures are difficult to train (Bengio and Glorot, 2010). Deep or hierarchical architectures are good at extracting high-level and abstract features (Bengio, 2009). However, the training of these multilayer architectures using a traditional gradient descent algorithm with random weight initialization always falls into local minima. But if the weights are well initialized so that they are close enough to a good solution, a gradient descent algorithm would work well (Bengio, 2009). In 2006, Hinton and Salakhutdinov proposed a greedy layer-wise method to pretrain deep neural networks so as to gain good initial weights, and then a gradient descent algorithm is employed to fine-tune the weights (Hinton and Salakhutdinov, 2006). Figure 2-3 shows the training procedure of a deep autoencoder proposed by Hinton and Salakhutdinov. The training process has three steps. The pretraining step, treats every two layers as a restricted Boltzmann machine (RBM), and trains the RBM using contrastive divergence (CD) algorithm (Hinton and Salakhutdinov, 2006). After that, the network

is unrolled into an encoder and a decoder. Finally gradient descent is adopted to fine-tune the deep network.

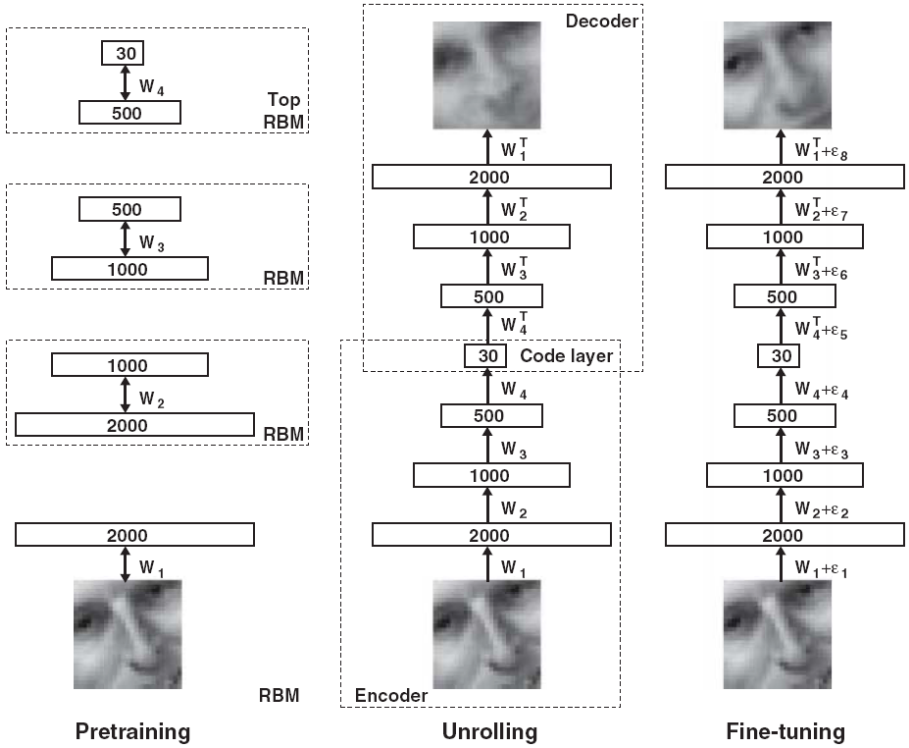


Figure 2-3. Training procedure of a deep autoencoder (Hinton and Salakhutdinov, 2006).

Based on the pretraining method, Hinton et al. proposed deep belief networks (DBNs) (Hinton et al., 2006). DBN achieves outstanding performance on small-size digit recognition. However, the accuracy dramatically decreases when DBN is scaled to full-size images. The main reason could be that DBN cannot capture spatial cues of images. To address this problem, Lee et al. proposed convolutional deep belief networks (CDBNs), which adopt convolution operators to deal with spatial contents, to learn features for full-size images (Lee et al., 2009). In addition to CDBN, several other hierarchical feature learning methods based on deep learning, e.g.,

Deconvolutional Networks (DN) (Zeiler, 2010), Hierarchical Matching Pursuit (HMP) (Bo et al., 2011), Convolutional-Recursive Neural Networks (CRNN) (Socher et al., 2012), have been proposed for handling full-size images in recent years.

Chapter 3 An Adaptive Recognition Model Based on Neural Networks for Image Annotation

3.1 Background

Automatic image annotation, usually termed image annotation, is a process that employs various machine learning algorithms to find out the mappings between low-level visual features and high-level labels/concepts from labeled images, and then propagates the labels to unlabeled images based on the learnt mappings. In recent years, apart from the direct mapping between visual features and labels, label correlation, which reflects co-occurrence relationship of different labels, has also been adopted to refine the performance of image annotation (Wang et al., 2009; Liu et al., 2009; Fu et al., 2010; Zhang and Ma, 2009). However, many existing label correlation learning methods characterize linear, symmetric, and positive correlation only. We consider that in practice label correlation can be nonlinear, asymmetric, both positive and negative. We need nonlinear correlation because that in practice the co-occurrence relationships between labels can be very complicated. A linear model may not be effective enough to represent label correlation. Asymmetric correlation means that two labels can have different influences to each other. like “clouds” and “sky”. In a natural image, “clouds” always appears together with “sky”; but sometimes “sky” appears without “clouds”. Positive correlation indicates that the occurrence of a label

increases the probability of the occurrence of another label like “sky” and “clouds”; while negative correlation indicates that the occurrence of a label inhibits the occurrence of another label, like “sun” and “moon”. Moreover, little existing work systematically studies the effect of label correlation in image annotation. How significantly the label correlation plays a role, in what situation and to what extent that the label correlation can improve the annotation performance remain unknown. To address these problems, in this chapter, the effect of label correlation is comprehensively and systematically investigated based on a proposed synthetic image dataset and a real image dataset.

Because of their powerful information processing abilities and simple implementations, neural networks have been widely applied in a variety of domains (Zhu and Wang, 2011; Yu et al., 2010; Chang et al., 2009; Chen et al., 2009; Zhu and Cao, 2011; Burse et al., 2010). Neural networks have also been adopted to solve the image annotation problem (Zhao et al., 2008; Tsurugai et al., 2008). However, Zhao et al. (Zhao et al., 2008) made use of simple global visual features only, while it has been demonstrated that global visual features may not characterize images with multiple complex objects well (Wang et al., 2009). Tsurugai et al. (Tsurugai et al., 2008) did not take into account label correlative information. Recently, a concept association network (CAN) has been proposed for image annotation (Fu et al., 2010). As advanced attention-driven techniques (Fu et al., 2006; Fu et al., 2009) are adopted for image segmentation, promising performance is achieved by CAN (Fu et al., 2010). However, the CAN suffers from an inefficiency problem. Its annotation time is

significantly influenced by the vocabulary size and the number of segmented regions of the input image. In addition, CAN characterizes label correlation by using a positive linear system. But a linear system may not be able to effectively characterize complex label correlation.

In view of this, we propose an adaptive recognition model (ARM) for image annotation in this chapter. Different from that the CAN characterizes label correlations using a hand-designed linear system, ARM uses a nonlinear two-layer perceptron to automatically learn label correlation from training images, with the correlation values storing in the connection weights of the perceptron. The label correlation can be intuitively analyzed based on the connection weights. Besides a real image dataset, we also construct a synthetic image dataset to evaluate our proposed model. To our best knowledge, a synthetic dataset has not yet been used for experiments of image annotation in the previous work. There are two main advantages of conducting annotation experiments on synthetic images. Firstly, the use of synthetic images can markedly reduce or even get rid of various noises introduced in inaccurate segmentation and feature representation. Thus, we can concentrate our investigation on the properties and performance of our model. Secondly, the synthetic images are controllable, and hence, we can construct synthetic images according to our requirements for specific research. For example, we can control the extent of correlation among some labels in the synthetic dataset to investigate the effect of label correlation network in image annotation.

3.2 Adaptive Recognition Model (ARM)

Figure 3-1 illustrates the annotation flowchart of our proposed ARM. The right dash box in Figure 3-1 represents an ARM, which consists of a classification network (CFN) and a correlation network (CLN). The CFN is made up of a number of copies of a classifier and an integration layer that integrates the classifier outputs. The annotation is carried out as follows. Given an input image *I*, first of all, it is segmented using state-of-the-art segmentation techniques such as an attention-driven model (Fu et al., 2006; Fu et al., 2009) or JSEG (Deng and Manjunath, 2001) into *R* regions, from which visual features are extracted. The visual features representing the *R* regions are then inputted to the classifiers of CFN with one classifier copy processing one region. After that, the classifier outputs are assembled in the integration layer to produce a classification result. Finally, the classification result is refined in the CLN using the correlative relationship of labels to generate the final annotation for the input image. The CFN and CLN are discussed in full details in the subsequent two sections, respectively.

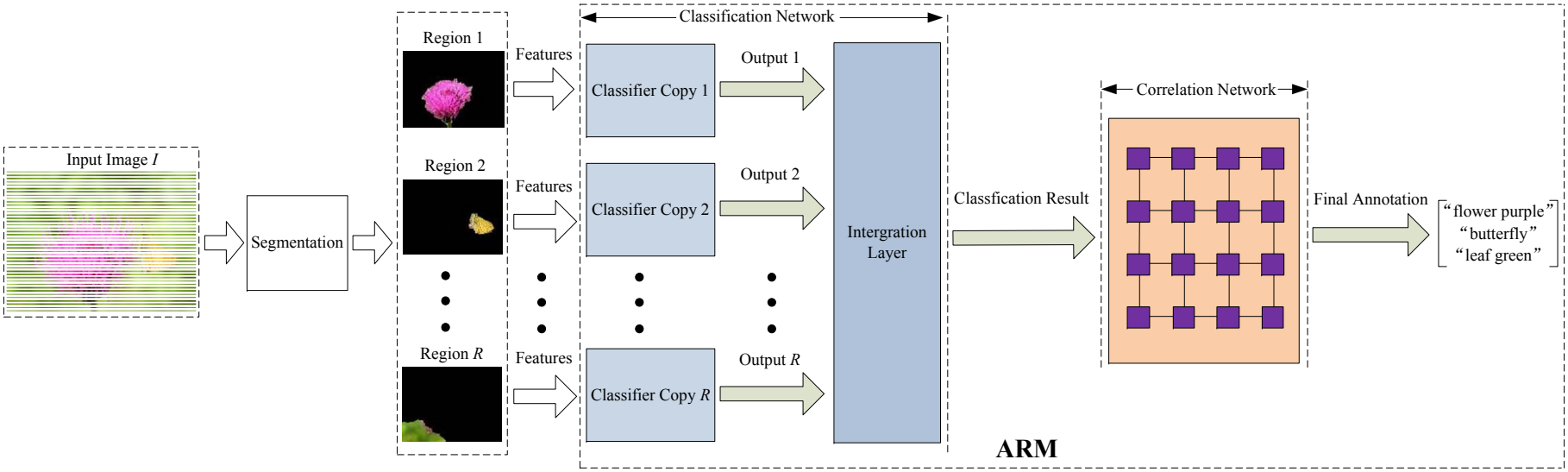


Figure 3-1. Annotation flowchart of our proposed ARM. The right dash box represents the ARM.

3.2.1 Classification Network

The CFN consists of a number of classifier copies and an integration layer. The classifier copies are duplicated from the same classifier. Thus, only one classifier needs to be trained. The classifier is a feed-forward neural network with three layers, i.e., the input layer, hidden layer, and output layer. In mathematics, a three-layer neural network classifier with *M* input nodes, *N* hidden nodes, and *T* output nodes can be formulated in matrix form as follows:

$$\boldsymbol{h} = \boldsymbol{f}_1\left(\boldsymbol{W}^I \boldsymbol{x} + \boldsymbol{b}^H\right) \tag{3-1}$$

and

$$\boldsymbol{y} = \boldsymbol{f}_2\left(\boldsymbol{W}^O \boldsymbol{h} + \boldsymbol{b}^O\right) \tag{3-2}$$

where $\boldsymbol{x} = \{x_i\}, i = 1, \cdots, M$ is the input vector. $\boldsymbol{y} = \{y_i\}, k = 1, \cdots, T$ is the output vector. $\boldsymbol{h} = \{h_j\}, j = 1, \cdots, N$ is the output of the hidden layer. $\boldsymbol{W}^I = \{w^I_{j,i}\}, i = 1, \cdots, M, j = 1, \cdots, N$ and $\boldsymbol{W}^O = \{w^O_{k,j}\}, j = 1, \cdots, N,\ k = 1, \cdots, T$ are the input weight matrix and output weight matrix, respectively. $\boldsymbol{b}^H = \{b^H_j\}, j = 1, \cdots, N$ is the bias vector of the hidden layer, and $\boldsymbol{b}^O = \{b^O_k\}, k = 1, \cdots, T$ is the bias vector of the output layer. Function arrays $\boldsymbol{f}_1(\cdot)$ and $\boldsymbol{f}_2(\cdot)$ are, respectively, made of *N* and *T* transfer functions $f(\cdot)$. In this thesis, $f(\cdot)$ is a sigmoid function, which is defined as follows:

$$f(x) = \frac{1}{1 + \mathrm{e}^{-\beta x}} \tag{3-3}$$

where β is a smooth parameter. The output of $f(\cdot)$ is a real value between 0 and 1. Substituting Eq. (3-1) into Eq. (3-2), we can obtain

$$\boldsymbol{y} = \boldsymbol{f}_2\left(\boldsymbol{W}^O \boldsymbol{f}_1\left(\boldsymbol{W}^I \boldsymbol{x} + \boldsymbol{b}^H\right) + \boldsymbol{b}^O\right). \tag{3-4}$$

The input–output relationship of the classifier is characterized in Eq. (3-4).

The outputs of the classifiers are integrated in the integration layer according to the following equation:

$$\boldsymbol{u} = \boldsymbol{g}\left(\sum_{r=1}^{R} \boldsymbol{y}_r\right) \tag{3-5}$$

where $\boldsymbol{u}$ is the output vector of the integration layer. $\boldsymbol{y}_r$ is the output vector of the rth classifier. R is the number of classifier copies used for the input image (equal to the number of regions). $\boldsymbol{g}(\cdot)$ is a function array made of T piecewise functions $g(\cdot)$ defined as follows:

$$g(x) = \begin{cases} 1, & \text{if} \quad x > 1 \\ x, & \text{if} \quad 0 \le x \le 1 \end{cases}$$

Note that the input x is non-negative, since the output of the transfer function $f(\cdot)$ defined in Eq. (3-3) is between 0 and 1. We can expand Eq. (3-5) in element form as

$$u_k = g\left(\sum_{r=1}^{R} y_{k,r}\right) \tag{3-6}$$

where u_k is the kth element of $\boldsymbol{u}$, and $y_{k,r}$ is the kth element of the rth classifier output vector.

Compared with CAN, there are three main advantages of a CFN. Firstly, only one classifier needs to be trained. Secondly, the classification results of the regions are assembled in the integration layer. This structure enables the CFN to exploit not only regional features, but also global features to some extent for image classification. Thirdly, many region-based annotation approaches including a CAN associate one segmented region with only one label, which may not be effective to characterize

segmented regions in practice, while our approach allows one region to be associated with multiple labels.

3.2.2 Correlation Network

The CLN is a two-layer (the input layer and output layer) perceptron. The output of the integration layer in CFN is the input of CLN. The formulation of CLN can be mathematically expressed as

$$\boldsymbol{v} = \boldsymbol{f}_3\left(\boldsymbol{W}^C\boldsymbol{u} + \boldsymbol{b}^C\right) \tag{3- 7}$$

where $\boldsymbol{u} = \{u_k\},\ k = 1, \cdots,\ T$ is the input vector as defined in Eq. (3-5). The weight matrix $\boldsymbol{W}^C = \left\{w_{l,k}^C\right\},\ k = 1, \cdots,\ T,\ l = 1, \cdots,\ T$ is a correlation matrix, and $\boldsymbol{b}^C = \left\{b_l^C\right\},\ l = 1, \cdots,\ T$ is the bias vector of the output layer. $\boldsymbol{f}_3(\cdot)$ is a function array that is made of T transfer functions $f(\cdot)$ defined in Eq. (3-3). $\boldsymbol{v} = v_k,\ k = 1, \cdots,\ T$ is the output vector of CLN. $\boldsymbol{v}$ is also the final annotation vector of the ARM. The following treatments are taken.

1) select R labels corresponding to the R largest elements of output $\boldsymbol{v}$ as the final annotation, if $R \leqslant 5$.
2) select five labels corresponding to the five largest elements of output $\boldsymbol{v}$ as the final annotation, if $R > 5$.

Note that the maximum annotation length is set to be 5 here. The maximum length is selected according to the ground truth statistics of the image dataset. For example, the maximum length can be determined by the maximum label length or the most frequently appeared label length in the ground truth of the training images. We set the

maximum annotation length to 5 here because the real images used in the experiments are associated with at most five labels. In a future application, we can increase the maximum length when new labels and images are added.

We can expand Eq. (3-7) into

$$v_k = f\left(\sum_{l=1}^{T} w_{l,k}^{C} u_l + b_k^{C}\right). \tag{3-8}$$

As the bias b_k^C is a constant after training, v_k is mainly determined by $\sum_{l=1}^{T} w_{l,k}^{C} u_l$, where $w_{l,k}^{C}$ is a correlation coefficient that reflects the occurrence frequency of the lth label given the kth label. After the training of CLN, the correlative information of labels is stored in matrix $\boldsymbol{W}^C$. Thus, the correlation matrix $\boldsymbol{W}^C$ is the key component to analyze the label correlation of training images. Note that $w_{l,k}^{C}$ can be different from $w_{k,l}^{C}$. If $w_{l,k}^{C} = w_{k,l}^{C}$, $\boldsymbol{W}^C$ is a symmetric matrix. Figure 3-2 illustrates an example of the correlation matrix $\boldsymbol{W}^C$ for three labels: “sky”, “sun”, and “moon”. The correlation values have been normalized to [−1, 1]. Note that the correlation value can be negative as shown between “sun” and “moon.” Negative correlation represents that the occurrence of one label inhibits the occurrence of some other labels in the same images, e.g., “sun” and “moon”. A value of -0.48 means that the occurrence of “moon” (the column) inhibits the occurrence of “sun” (the row) with a degree 0.48. In summary, there are two main differences between the ARM and CAN in characterizing label correlation. Firstly, the CAN adopts a simple linear network to represent label correlation, while our proposed ARM employs a nonlinear network, which is more effective to characterize the complex nonlinear relationship between labels. Secondly, the CAN processes only positive correlation, while the ARM is able

to deal with both positive and negative label correlation.

	sky	sun	moon
sky	0.95	0.58	0.47
sun	0.79	0.91	-0.48
moon	0.82	-0.52	0.89

Figure 3-2. Example of the correlation matrix $\boldsymbol{W}^C$ for three labels: “sky”, “sun”, and “moon”.

3.3 Training Algorithms of ARM

In our approach, CFN and CLN are trained separately. Both CFN and CLN are trained by the conventional error back-propagation (BP) algorithm.

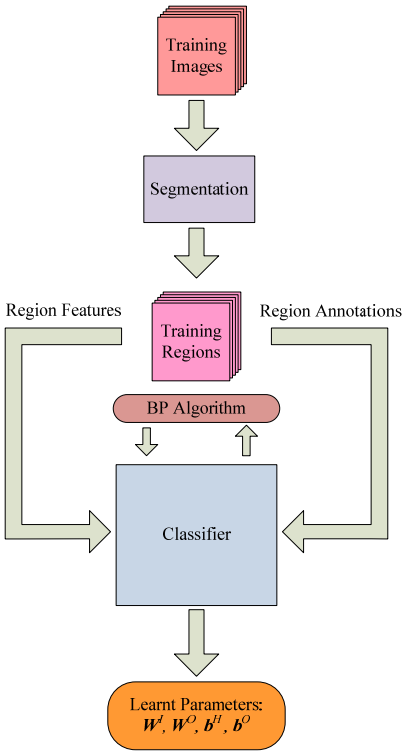


Figure 3-3. Training process of the CFN.

The parameters of CFN that need to be trained are weight matrices $\boldsymbol{W}^I$ and $\boldsymbol{W}^O$, and bias vectors $\boldsymbol{b}^H$ and $\boldsymbol{b}^O$ of the region classifier. The training of CFN is hence converted to the training of an MLP classifier (three-layer feed-forward neural network) using the BP algorithm. Figure 3-3 illustrates the training process of the classifier. Training images are first segmented into regions. Every region is represented by a feature vector and associated with a binary 0-1 label vector, in which value 1 denotes the annotated label. If a region is too ambiguous to be annotated with any labels, its label vector has only value 0. The region features and annotations are then fed to the classifier, where region features are the classifier input and region annotations are the target output. Given randomly initialized parameters $\boldsymbol{W}^I$, $\boldsymbol{W}^O$, $\boldsymbol{b}^H$ and $\boldsymbol{b}^O$, the BP algorithm updates iteratively the parameters according to the error between the target output and the actual output of the classifier. After a number of iterations, the expected parameters $\boldsymbol{W}^I$, $\boldsymbol{W}^O$, $\boldsymbol{b}^H$ and $\boldsymbol{b}^O$ can be learnt.

The parameters of CLN to be trained are the correlation matrix $\boldsymbol{W}^C$ and bias vector $\boldsymbol{b}^C$. The training process of CLN is depicted in Figure 3-4. In contrast to the classifier trained on regions, the training of CLN is based on images. Given a training image associated with a label vector $\boldsymbol{k}$, vector $\boldsymbol{k}$ is used as both the input and target output of CLN. That is $\boldsymbol{u} = \boldsymbol{v} = \boldsymbol{k}$ (Eq. 3-7). Therefore, the training of CLN aims to learn the correlation matrix $\boldsymbol{W}^C$ and bias vector $\boldsymbol{b}^C$ using the annotations of training images.

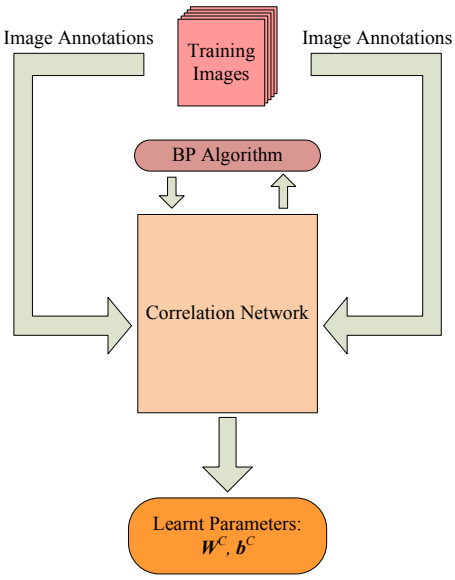


Figure 3-4. Training process of CLN.

3.4 Experiments

The experiments are conducted on a both synthetic image dataset and a real image dataset. The subsequent two sub-sections describe two datasets and their experimental results, respectively. A discussion on label correlation is presented at the end of this section.

3.4.1 Synthetic Image Dataset

3.4.1.1 Description of the Dataset

Six fundamental colors, namely “red”, “green”, “blue”, “yellow”, “cyan” and “magenta”, and nine basic shapes, namely “round”, “triangle”, “rectangle”, “octagon”, “4-point star”, “5-point star”, “moon”, “heart” and “lighting bolt” depicted in Figure 3-5 are used to construct synthetic images. As shown in Figure 3-5, each color covers

a certain range of values from dark to bright instead of using a constant value. When a color is selected for image construction, the color's value is randomly chosen in its value range.

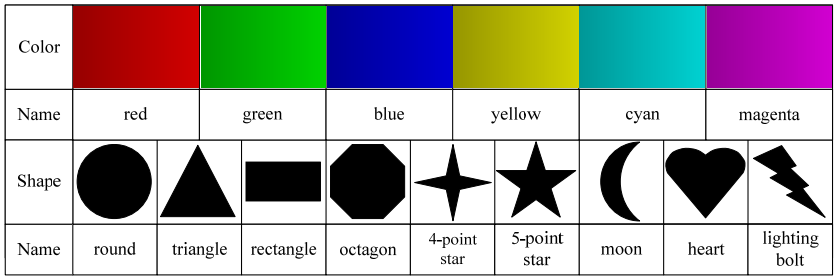


Figure 3-5. Six colors and nine shapes used for synthetic image construction.

One shape filled in with one color constructs one synthetic object, whose ground truth consists of the shape name and the color name. For example, shape "round" and color "red" make up an object named "round_red". Nine shapes and six colors totally construct 54 different objects, from which a 54-dimensional label vector is formed. The 54 synthetic labels and their corresponding entry numbers in the label vector are listed in Table A-1 in Appendix A. Table A-1 will help us to investigate the label correlation later on. A synthetic image is made of one to four objects with a white background. The white background is used to highlight the objects for better visualization. The objects can be arbitrarily rotated in the images. To simplify the experiments, we do not consider overlapped objects. Hence, the objects can be well segmented by JSEG algorithm (Deng and Manjunath, 2001), with one segmented region well representing one object.

In order to study the effect of label correlation, we design two synthetic image sets for experiments. R ($1 \leq R \leq 4$) objects are randomly generated for each image of

Set 1. In other words, Set 1 has no artificial correlation. In Set 2, we let some objects always appear together to make artificial correlations. Specifically, “round” and “triangle”, “rectangle” and “octagon”, “4-point star” and “5-point star”, “moon” and “heart” filled in with the same color are made in the same images. For example, “round_red” and “triangle_red” appear together. They are called a label-pair. A label-pair has strong mutual correlation. There are in total 24 label-pairs. According to Table A-1, the 24 label-pairs can be represented by number-pairs as 1-7, 2-8, 3-9, 4-10, 5-11, 6-12, for “round”-“triangle” pairs, 13-19, 14-20, 15-21, 16-22, 17-23, 18-24, for “rectangle”-“octagon” pairs, 25-31, 26-32, 27-33, 28-34, 29-35, 30-36, for “4-point star”-“5-point star” pairs, and 37-43, 38-44, 39-45, 40-46, 41-47, 42-48 for “moon”-“heart” pairs. Note that *i-j* and *j-i* are the same. Thus, the correlation matrix $\boldsymbol{W}^C$ is a symmetric matrix in theory. Figure 3-6 and Figure 3-7 respectively display example images of Set 1 and Set 2. Both Set 1 and Set 2 have 1000 images, in which 300 images are for training, 300 for validation and 400 for testing. The ratio among training size, validation size and testing size is 3:3:4. For each region, 2*2*2 RGB color histogram and 7 invariant moments are used to characterize the visual content. This generates a 15-dimensional feature vector. Note that we do not take into account the regions that represent the white background. In other words, the white background of a synthetic image is not involved in either training or testing. However, for real images presented in Section 3.4.2, all the image regions including the background are utilized for experiments. The number of hidden nodes of the classifier is chosen 20 for the synthetic image dataset.

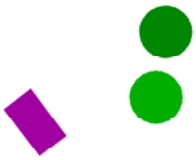
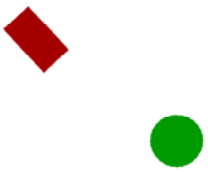
Image			
Ground truth	round_magenta	round_red octagon_yellow moon_blue lighting bolt_red	round_green rectangle_magenta
Image			
Ground truth	round_blue triangle_cyan 5-point star_yellow moon_green	round_green rectangle_red	octagon_magenta 4-point star_blue 4-point star_yellow lighting bolit_blue

Figure 3-6. Examples of synthetic images of Set 1.

Image			
Ground truth	lighting bolt_cyan	round_red triangle_red lighting bolt_blue	rectangle_magenta octagon_magenta 4-point star_cyan 5-point star_cyan
Image			
Ground truth	rectangle_red octagon_red moon_yellow heart_yellow	rectangle_green octagon_green	moon_blue heart_blue lighting bolt_green

Figure 3-7. Examples of synthetic images of Set 2.

3.4.1.2 Experimental Results

Mean per-word precision and mean per-word recall are two important measures to evaluate the annotation performance (Duygulu et al., 2002; Feng et al., 2004). The definition of precision and recall for label *w* is defined as follows.

$$precision(w) = \frac{A}{B}, \qquad recall(w) = \frac{A}{C}, \tag{3-9}$$

where A is the number of images correctly annotated with w, B is the number of images automatically annotated with w, C is the number of images manually annotated with w (ground truth). Mean per-word precision and mean per-word recall are obtained by averaging per-word precision and per-word recall over all the labels manually labeled to the testing images. One problem in these measures is that when the labels have significantly different numbers of images manually annotated with them, the final evaluation results are considerably impacted by the annotation of some images. For example, suppose that in the testing images, label “a” is manually annotated to 1000 images, while label “b” is manually annotated to only 1 image. When the precision and recall are averaged over these two labels to obtain mean per-word precision and mean per-word recall, “a” and “b” are treated equally. This is actually unfair, as the annotation of the image manually labeled with “b” dramatically influences the overall performance. To solve this issue, we adopt mean per-image precision and mean per-image recall (Tang and Lewis, 2007) for annotation evaluation here. Considering that precision and recall always conflict with each other, we also design an F-score based on precision and recall. The precision, recall, and

F-score for image I are defined as follows:

$$precision(I) = \frac{O}{P}, \tag{3-10}$$

$$recall(I) = \frac{O}{Q}, \tag{3-11}$$

$$F(I) = \frac{2 \times precision(I) \times recall(I)}{precision(I) + recall(I)} \tag{3-12}$$

where O is the number of labels correctly annotated to I, P is the number of labels automatically annotated to I, Q is the number of labels manually annotated to I. As the numbers of labels annotated to different images vary in a small range, the problem existing in mean per-word precision and mean per-word recall can be significantly alleviated. Hereafter, mean per-image precision, mean per-image recall and mean per-image F-score are termed precision, recall and F-score for short.

To study the effect of label correlation, we compare the annotation performances of CFN only and CFN+CLN (ARM). The experimental results on Set 1 and Set 2 are depicted in Figure 3-8 and Figure 3-9, respectively. As can be seen from Figure 3-8, the precision, recall and F-score are 0.835, 0.844, 0.840 respectively for CFN, and 0.828, 0.837, 0.832 respectively for CFN+CLN. The performance of CFN+CLN is similar with that of CFN, meaning that the use of label correlation does not produce better annotation results on Set 1. By contrast, Figure 3-9 shows that on Set 2, CFN+CLN for annotation significantly improves the precision from 0.845 to 0.902, recall from 0.854 to 0.910, and F-score from 0.850 to 0.906. This can be accounted for by that there is no obvious correlation among labels in Set 1, while there is strong artificial correlation among some labels in Set 2. Consequently, the CLN trained by

Set 2 is able to refine the annotation results using the helpful correlative information of labels learnt from Set 2.

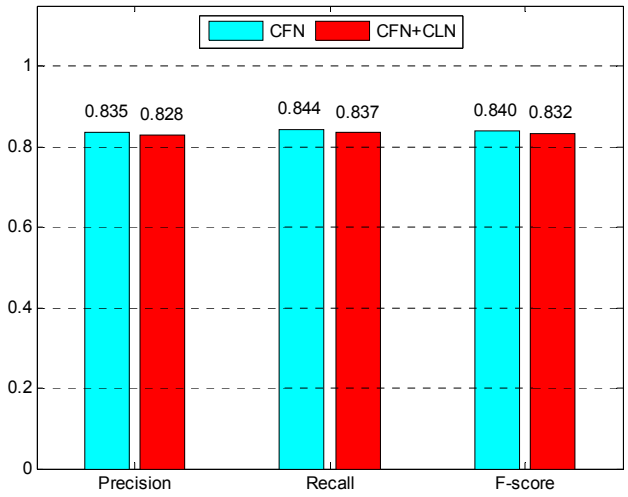


Figure 3-8. Annotation performance of CFN and CFN+CLN on synthetic image Set 1.

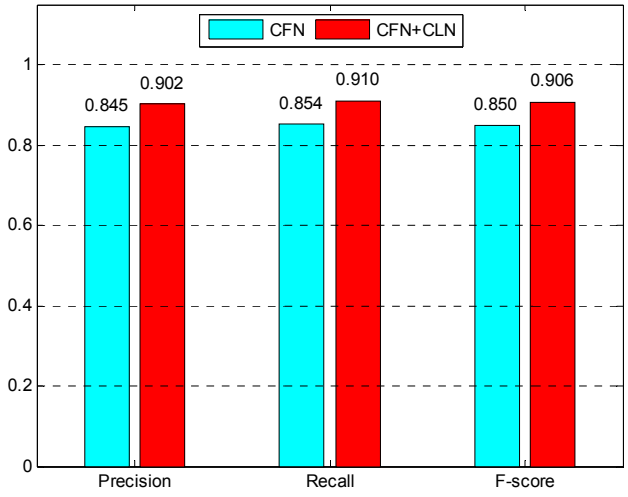


Figure 3-9. Annotation performance of CFN and CFN+CLN on synthetic image Set 2.

Figure 3-10 and Figure 3-11, respectively, visualize the correlation matrices $\boldsymbol{W}^C$ learnt from Set 1 and Set 2. A brighter region represent a larger value (a stronger

correlation) while a darker region a smaller value (a weaker correlation). The correlation matrix $\boldsymbol{W}^C$ of Set 1 has a bright diagonal region, indicating that the labels of Set 1 have strong self-correlations only, (little mutual correlation exists among different labels). The $\boldsymbol{W}^C$ of Set 1 can be roughly regarded as an identity matrix. That is why combining CFN and CLN for annotation does not produce improvement as shown in Figure 3-8. Differing from $\boldsymbol{W}^C$ of Set 1, the correlation matrix $\boldsymbol{W}^C$ of Set 2 visualized in Figure 3-11 has bright elements not only on the diagonal, but also on other 8 regions, each of which has 6 elements. As these 8 regions symmetrically locate on both sides of the diagonal, we only need to choose 4 regions above the diagonal for analysis. From left to right, the 4 regions are named Regions 1, 2, 3 and 4. The positions of the 4 regions' elements can be labeled with 24 label entry number pairs as 1-7, 2-8, 3-9, 4-10, 5-11, 6-12 for Region 1, 13-19, 14-20, 15-21, 16-22, 17-23, 18-24 for Region 2, 25-31, 26-32, 27-33, 28-34, 29-35, 30-36 for Region 3, and 37-43, 38-44, 39-45, 40-46, 41-47, 42-48 for Region 4. The label corresponding to the 24 entry number pairs have a strong mutual correlation. Clearly, the 24 entry number pairs match the 24 label-pairs presented in Section 3.4.1.1 with Region 1 representing "round"-"triangle" pairs, Region 2 "rectangle"-"octagon" pairs, Region 3 "4-point star"-"5-point star" pairs, and Region 4 "moon"-"heart" pairs. It is worth pointing out that there are six elements on the diagonal which are brighter than the other elements in Figure 3-11. That is to say, six labels have a stronger self-correlation than the other 48 labels. The entry numbers of these six elements are 49 to 54, corresponding to labels "lighting bolt-red", "lighting bolt-green", "lighting bolt-blue",

"lighting bolt-yellow", "lighting bolt-cyan" and "lighting bolt-magenta". The reason for their stronger self-correlations can be explained by that they are not forced to appear with any labels. So they have opportunities to be annotated alone to an image, which strengthens their self-correlations. The other 48 labels always emerge with their partners, by which the mutual correlation is strengthened and the self-correlation is weakened. Evidently, the proposed CLN is effective to characterize the label correlation.

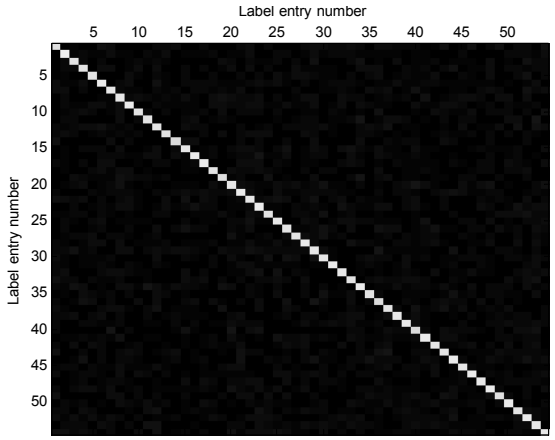


Figure 3-10. Visualization of the correlation matrix $\boldsymbol{W}^C$ learnt from synthetic image Set 1, where a brighter region represents a larger value (a stronger correlation).

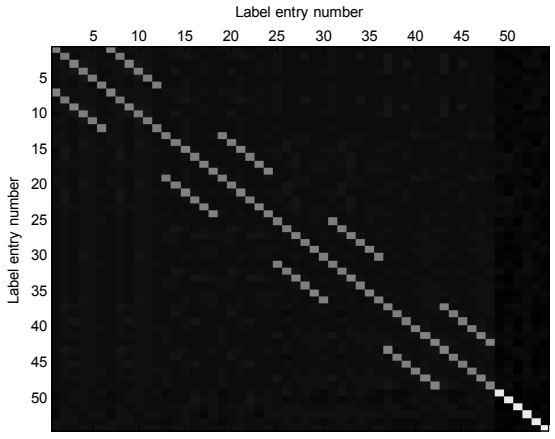


Figure 3-11. Visualization of the correlation matrix $\boldsymbol{W}^C$ learnt from synthetic image Set 2, where a brighter region represents a larger value (a stronger correlation).

In another experiment, we employ CFN and CLN trained on Set 2 to annotate the testing images of Set 1. Figure 3-12 displays the annotation performance of CFN and CFN+CLN, indicating that the use of CLN seriously degrades the performance. The precision and recall decrease by 0.292 and 0.293, respectively. F-score drops from 0.815 to 0.522. This experiment demonstrates that the label correlation can refine the annotation on the premise that the testing images have the same or similar label correlation with the training images. In other words, the label correlation of training images should be applicable to testing images. Otherwise, the correlative information is useless or even harmful to image annotation.

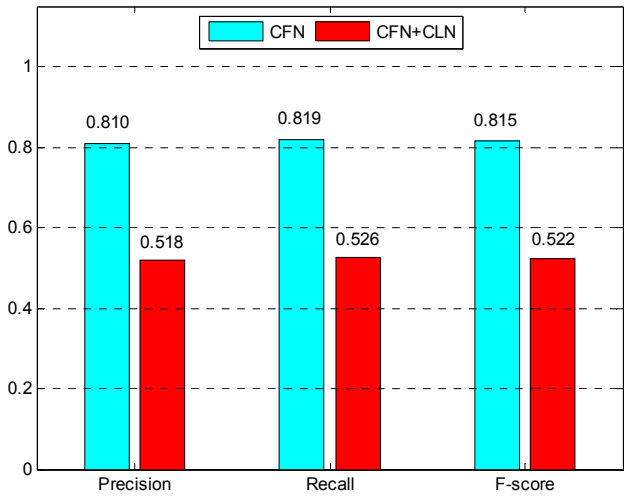


Figure 3-12. Annotation performance of CFN and CFN+CLN on the testing images of Set 1, where CFN and CLN are trained on Set 2.

3.4.2 Real Image Dataset

3.4.2.1 Description of the Dataset

Concept Association Network (CAN) (Fu et al., 2010) was experimented on a real image dataset collected from the Internet. For a convenient comparison, we use the same dataset. This dataset contains 5306 images and 98 annotated labels. Every image is manually annotated with 1 to 5 labels in ground truth. The images are segmented by attention-driven model (Fu et al., 2006; Fu et al., 2009). Every region is manually annotated, and represented by a 37-dimensional feature vector. The 37-dimensional features consist of a 27-dimensional HSV color histogram and a 10-dimensional edge histogram. In our experiments, 30% of images are for training, 30% for validation and 40% for testing. This is equivalent to 1592 training images. 1592 validation images and 2122 testing images. As the real image dataset has a higher dimension of features and more labels than the synthetic image dataset, we empirically increase the number of hidden nodes of the classifier to 50.

3.4.2.2 Experimental Results

Figure 3-13 illustrates the annotation performance of CAN and ARM. ARM-5 represents ARM that sets the maximum annotation length to 5, which is also the scheme defined in Section 3.2. Figure 3-13 shows that the precision of CAN (0.354) is 0.034 higher than that of ARM (0.32). However, compared with CAN whose recall is 0.446, the recall of ARM is 0.539, 0.093 higher than that of CAN. ARM also

achieves a slightly larger F-score (0.402) than CAN (0.395). The main reason for why CAN outperforms ARM in precision can be explained as follows. Among 2122 testing images, 2068 images have less than four labels, and there are in total 1850 images whose number of regions is larger than the number of labels in ground truth. However, our annotation scheme presented in Section 3.2 determines the number of labels for an image based on the number of regions of the image when the number of regions is not larger than 5, and sets the maximum annotation length to 5. Hence, for the real dataset we used, this scheme generates redundant labels in most cases, which reduces the annotation precision. Now we set the maximum annotation length to 3 and retest ARM (termed as ARM-3). The performance is also shown in Figure 3-13. This time the precision of ARM increases to 0.353, roughly equal to that of CAN. Although the recall of ARM decreases from 0.539 to 0.508, it is still more than 6% higher than that of CAN, and the F-score of ARM is 2% higher than that of CAN. Figure 3-13 demonstrates that ARM performs at least comparatively and usually better than CAN. In the experiments reported in the following, we still use 5 as the maximum annotation length. Table 3-1 tabulates the annotation performance of ARM when different (training+validation)/testing data divisions are used. The training size is equal to the validation size. The precision, recall, and F-score gradually increase as the proportion of training and validation images grows from 0.1 to 0.9. This is reasonable since the larger the number of training and validation images used, the better the model can be trained.

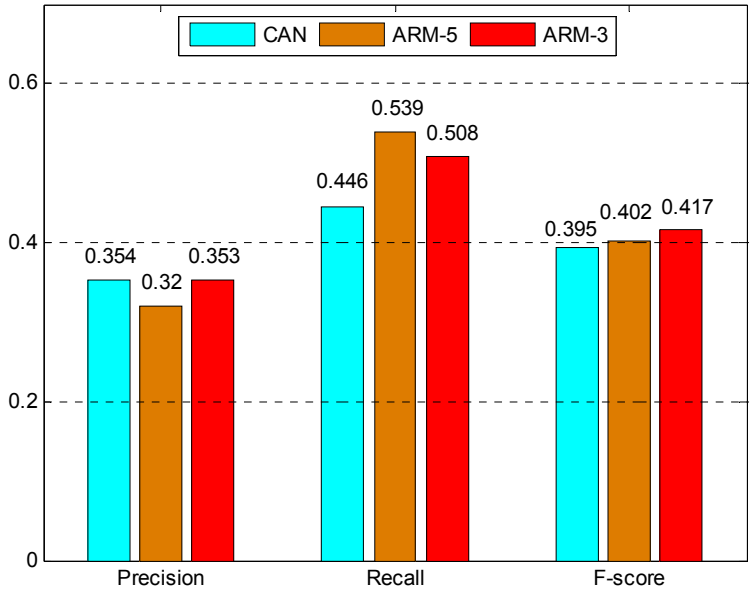


Figure 3-13. Annotation performance of CAN and ARM on the real dataset. ARM-5 represents ARM whose maximum annotation length is set to 5, and ARM-3 represents ARM whose maximum annotation length is set to 3.

Table 3-1. Performance of the ARM for different (training+validation)/testing data divisions.

(train+val.)/test	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1
Precision	0.279	0.288	0.297	0.308	0.314	0.320	0.328	0.342	0.348
Recall	0.475	0.486	0.508	0.522	0.529	0.539	0.551	0.569	0.580
F-score	0.352	0.362	0.375	0.388	0.394	0.402	0.411	0.427	0.435

Figure 3-14 illustrates some testing images and corresponding annotations produced by ARM when the maximum annotation length is set to 5. Some objects such as “fence”, “vase” and “bear” are difficult to be labeled correctly. One main reason that accounts for a poor detection of these objects is that their training samples are far from adequate. For instance, there is only one training image that contains “vase”, and the numbers of training images for “fence” and “bear” are only two and

three, respectively. It is even not easy for a human being to recognize an object if he/she has seen the object for one or two times only, not to mention an annotation model. Another reason lies in visual features. Taking “bear” as an example, its fur has similar color as glacier. As a result, it is difficult for an annotation model to distinguish “bear” from the background when the “bear” is in the glacier.

Image				
Ground truth	automobile road sky	sky trees_green water_sea	eagle mountain sky	earth snow tiger
ARM	automobile road sky snow	grass_green sky trees_green water_sea	eagle mountain sky	snow tiger trees_green water
Image				
Ground truth	chicken earth	blossom_pink fence leaf_green	flower_red vase	bear glacier
ARM	fish flower_red leaf_green	architecture great_wall leaf_green trees_green trunk	blossom_colorful fish flower_red	cloud glacier sky

Figure 3-14. Examples of testing images and corresponding annotations by ARM.

To compare the efficiency of CAN and ARM, we recorded the annotation time of the testing images for both models. The simulations are run on Matlab on an Intel Core2 Quad CPU 2.83 GHz, 6 GB RAM PC. As segmentation and feature extraction are not our focuses, the annotation time of an image is defined as a time slot between inputting visual features to the model and the model generating the final annotation for the image. The total annotation time and the mean per-image annotation time of

CAN and ARM on 2122 testing images are tabulated in Table 3-2. In contrast to CAN that needs to annotate an image with 1.47 seconds, the mean per-image annotation time of ARM is 0.02 seconds only. Obviously, our ARM significantly outperforms CAN in terms of efficiency. This is attributed to the forward-propagating working manner of ARM. In previous sections, we mentioned that the annotation time of CAN on an image is seriously influenced by the number of regions of the image and the vocabulary size. Subsequent experiments will demonstrate this point.

Table 3-2. The total annotation time and the mean per-image annotation time of CAN and ARM on 2122 testing images.

Model	CAN	ARM
Total annotation time (second)	3116.12	45.92
Mean per-image annotation time (second)	1.47	0.02

The annotation time shown in Table 3-2 is recorded on a vocabulary that has 98 labels. We now reduce the vocabulary size to 10, and gradually increase the size to 80. For each vocabulary size, a corresponding number of labels are randomly selected from the 98 labels for experiments. The random selection of labels is performed five times for each vocabulary size, and the averaging results of the five sets of labels are reported. With respect to different vocabulary sizes, the mean per-image annotation time of CAN and ARM is shown in Figure 3-15. As can be seen from Figure 3-15, the annotation time of CAN roughly linearly grows as vocabulary size increases. For the ARM, the mean per-image annotation time has increased very slightly, from 0.0209

seconds to 0.0215 seconds. In other words, the ARM is able to retain its high efficiency as vocabulary size grows, which is preferable as vocabulary size is always in a large scale and varied in a practical application. Figure 3-16 shows corresponding annotation performance in terms of F-score for ARM and CAN. As vocabulary size increases, the annotation F-score of both ARM and CAN decrease. Figure 3-16 also indicates that the difference between ARM and CAN decreases when the vocabulary size grows. This can be accounted for by that a large vocabulary size increases the complexity of the ARM, as a result of which the training of the ARM tends to be difficult and its annotation superiority over CAN reduces accordingly. Thus, the ARM is more effective for small vocabulary annotation than large vocabulary annotation.

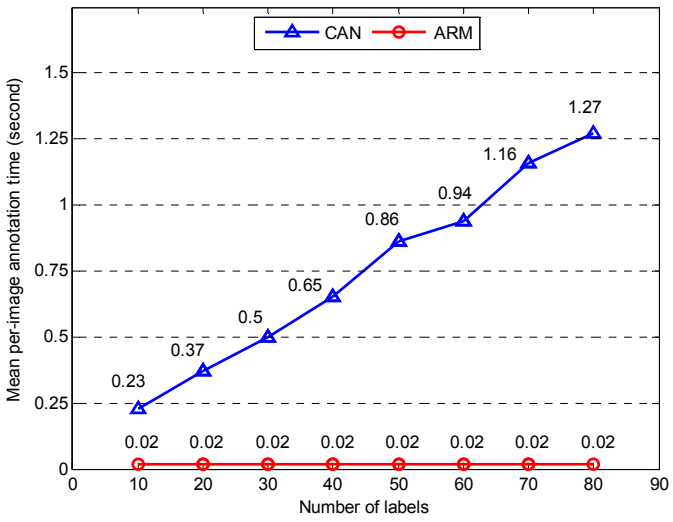


Figure 3-15. Mean per-image annotation time of CAN and ARM with respect to different vocabulary sizes.

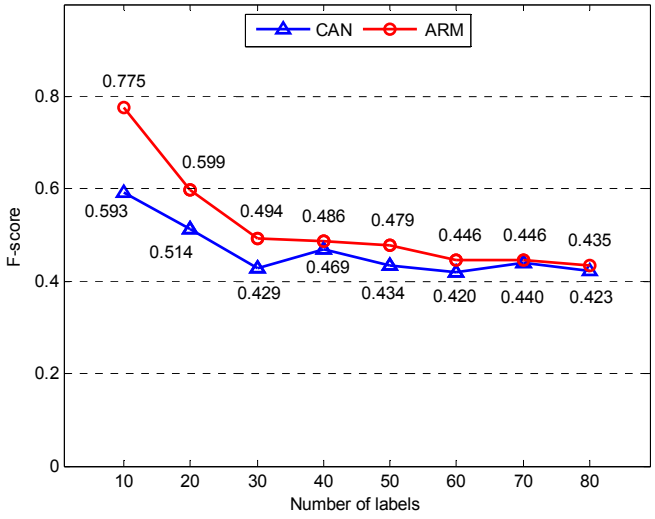


Figure 3-16. Annotation performance of CAN and ARM with respect to different vocabulary sizes.

We choose three sets of images from the 2122 testing images used in the experiments shown in Figure 3-13 to investigate the relationship between various numbers of regions in an image and the annotation time of CAN and ARM. The three sets all contain 100 images. Each image in the same set has the same number of regions. The first set has two regions in each image, while the other two sets have, respectively, three and four regions. ARM and CAN are the same to the ones used in the experiments shown in Figure 3-13, which are trained on the 3184 training and validation images and with 98 labels. Figure 3-17 depicts the mean per-image annotation time of CAN and ARM on three sets of images. As the number of regions in each image increases from two to four, the mean annotation time of CAN significantly grows from 0.9139 seconds to 2.029 seconds. By contrast, the mean annotation time of the ARM rises only slightly from 0.0196 seconds to 0.0234 seconds. This demonstrates that the annotation time of the ARM is insensitive to the

number of regions in each image. Figure 3-18 shows the corresponding annotation performance in terms of F-score for ARM and CAN. We can see that ARM outperforms CAN for all three sets of images.

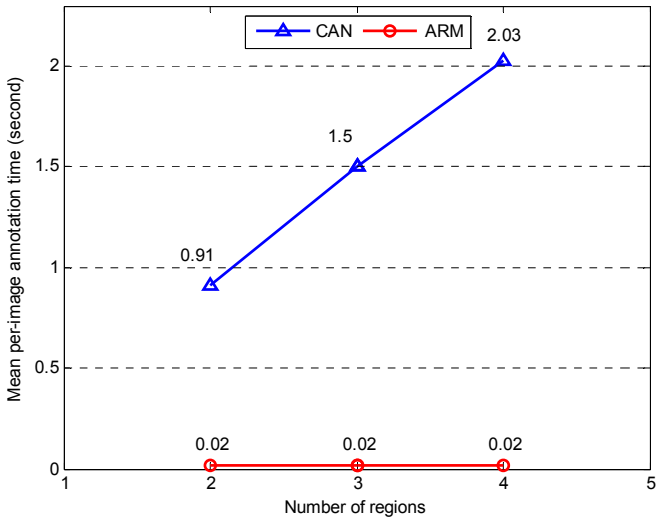


Figure 3-17. Mean per-image annotation time of CAN and ARM on three sets of images.

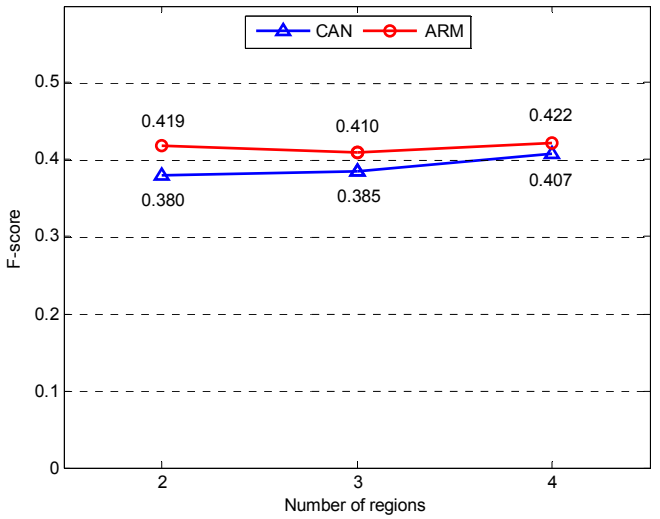


Figure 3-18. Annotation performance of CAN and ARM on three sets of images.

In the final experiment, we investigate the effect of label correlation in another aspect using ARM on the real dataset. The CFN of the ARM used in the experiments

shown in Figure 3-13 is named CFN 1. Besides CFN 1, four more CFNs are trained. The differences of these five CFNs are as follows. CFN 1 has the smallest training error, while CFN 2 to CFN 5 have gradually a larger training error. This can be attained by stopping the training of each CFN as soon as the training error reaches a predefined value. Excluding the case of overtraining, a larger training error always means a poorer annotation. That is to say, CFN 1 outperforms CFN 2, CFN 2 outperforms CFN 3, etc. We consider two types of annotations for each of the five CFNs: 1) annotation by CFN only; and 2) annotation by CFN + CLN, where CLN is the same as that is used in the experiments shown in Figure 3-13. The annotation performance of CFN and CFN + CLN for five CFNs is summarized in Table 3-3. As can be seen from Table 3-3, the performance of CFN for five CFNs (CFN 1 to CFN 5) gradually decreases from 0.310 to 0.046 in terms of precision, from 0.526 to 0.083 in terms of recall, and from 0.390 to 0.060 in terms of F-score. Nonetheless, compared with CFN only, CFN + CLN achieve an improved performance. The improvement made by CLN rises from 3.2% to 23.4% in precision, from 2.5% to 21.7% in recall, and from 3.0% to 22.8% in F-score. These results suggest that the poorer annotation the CFN can achieve, the more significant improvement CLN is able to make. The improvement made by the CLN for CFN 1 and CFN 2 is small with, respectively, 3% and 3.2% in F-score. This can be explained by that the label correlation of the dataset is not strong, since most of the images have only two to three labels. The experiment demonstrates that the label correlation plays a more important role when the annotation accuracy of a model is far from satisfactory.

Table 3-3. A performance comparison of CFN and CFN+CLN for five CFNs on 2122 testing images.

Model	Mean per-image precision	Precision improved by CLN (%)	Mean per-image recall	Recall improved by CLN (%)	Mean per-image F-score	F-score improved by CLN (%)
CFN 1	0.310	3.2	0.526	2.5	0.390	3.0
CFN 1+CLN	0.320		0.539		0.402	
CFN 2	0.201	3.5	0.345	2.6	0.254	3.2
CFN 2+CLN	0.208		0.354		0.262	
CFN 3	0.117	6.0	0.187	5.3	0.144	5.7
CFN 3+CLN	0.124		0.197		0.152	
CFN 4	0.078	11.5	0.147	8.8	0.102	10.6
CFN 4+CLN	0.087		0.160		0.113	
CFN 5	0.047	23.4	0.083	21.7	0.060	22.8
CFN 5+CLN	0.058		0.101		0.074	

3.4.3 Discussion on Label Correlation

Based on experimental results on both the synthetic dataset and the real dataset, the following conclusions for label correlation can be drawn.

1) In order for the label correlation to play an important role in the annotation improvement, the dataset must have strong correlative information among annotated labels.

2) The label correlation that is extracted from the training dataset is helpful only if the images to be annotated have the similar label correlation patterns. Otherwise, the use of label correlation is helpless or even harmful.

3) The label correlation plays a more important role when a visual classifier has a poorer performance.

In summary, if the training dataset possesses strong label correlation and such a label correlation is applicable to unlabeled images, it is always beneficial to incorporate label correlation for image annotation. The crucial issue is how to guarantee that the label correlation of the training dataset can be generalized to unlabeled images, which is also a tough research topic that needs more investigation.

3.5 Summary

In this chapter, we have presented an ARM for image annotation. The ARM consists of an adaptive CFN and a nonlinear CLN. The adaptive CFN enables the proposed ARM to exploit regional features together with global features for image

annotation. The CLN is capable of encoding label correlative information to improve the annotation results. Thanks to its forward-propagating working manner, the ARM is of high efficiency in annotating images. Furthermore, the computational time of an ARM is insensitive to the number of regions of the input image and vocabulary size. Besides a real image dataset, we have also designed a synthetic image dataset for experiments. The synthetic dataset is effective in the investigation of label correlation and the annotation performance of the ARM. Experimental results demonstrate the effectiveness and efficiency of our proposed model. A systematic study of label correlation to improve annotation performance has also been presented in this chapter.

Chapter 4 Multi-Instance Multi-Label Image Classification: A Hierarchical Neural Approach

4.1 Background

In Chapter 3, we propose a two-stage neural network model ARM for image annotation. ARM achieves high annotation efficiency and accuracy. However, ARM requires regional ground truth for training. The issue is that it is laborious and time consuming to manually label each image region, especially when the training set is in large scale. It is thus desirable for us to explore a method that only needs global image ground truth for training. To address this problem, in this chapter, a hierarchical neural architecture based on ARM is proposed for Multi-Instance Multi-Label image classification. Both MIML image classification and image annotation aim to assign multiple labels to an image. The difference is that for MIML image classification, an image is partitioned and represented by multiple instances/regions, and usually an MIML algorithm does not need to know regional ground truth for training; while for image annotation, an image can be represented by its global features or by multiple regions. To differ from the task performed in Chapter 3 that requires regional ground truth, we term the target task in this chapter MIML image classification rather than image annotation. The proposed neural model is thus termed Multi-Instance Multi-Label Neural Network (MIMLNN).

MIMLNN is different from ARM in two folds. Firstly, from the architecture perspective, MIMLNN extends the CorreLation Network (CLN) from a two-layer perceptron to a three-layer perceptron, because a perceptron with three layers (usually termed multi-layer perceptron) is more powerful in modeling complex functions than that with two layers. MIMLNN is expected to model more complex, nonlinear, both positive and negative, and asymmetric label correlation. Furthermore, by controlling the number of nodes in the hidden layer, we can significantly reduce the number of connection weights. It has been proved that if being well trained, a neural network with one more layer can approximate the same function using much less weights, and thus can achieve better generalization performance (Hinton and Salakhutdinov, 2006). Secondly, from the training or learning perspective, MIMLNN uses image ground truth only. MIMLNN is expected to automatically learn the region-label mapping and label correlation just by using region-level features and image-level labels of training images. In particular, the classical error Back-Propagation (BP) is employed to tune the weights of MIMLNN. A traditional BP algorithm is gradient descent. The gradient descent algorithm updates neural weights according to the partial derivative magnitude of a predefined error function. Since the error back-propagates by multiplying the derivative of the sigmoid function, the value of which is between 0 and 1, the gradient magnitude could be very small for deep layers when the error back-propagates through multiple layers. As a result, the weights at very deep layers are difficult to update. This is called the long-term dependency problem (Cho et al., 2003; Cho et al., 2005). To address this problem, we extend the refined BP algorithm

Rprop (Riedmiller, H. Braun, 1993) for MIMLNN training. Rprop algorithm updates weighs depending on the derivative sign rather than the derivative magnitude of the error signal. Therefore, the problem of long-term dependency can be significantly alleviated. The experiments reported in this chapter are conducted on a synthetic image dataset and the popular Corel image dataset. One main advantage of using a synthetic dataset is that the contents of synthetic images are controllable. For example, we can design specific label correlation for our research. Note that the synthetic dataset used in this chapter is different from that used in Chapter 3. To demonstrate the superior performance of MIMLNN, several state-of-the-art methods, e.g., MIMLBoost and MIMLSVM (Zhou and Zhang, 2007), are implemented for a comparison purpose.

4.2 Multi-Instance Multi-Label Neural Network (MIMLNN)

Figure 4-1 shows the image classification flowchart of MIMLNN. MIMLNN consists of two stages of MLPs. The first-stage MLP is named MLP^1, and the second-stage MLP is denoted as MLP^2. Given an image, the classification is conducted by MIMLNN as follows. First of all, the image is divided into a number of regions by performing segmentation or using a regular grid. After that, the features extracted from the regions are fed to MLP^1s, with one copy of MLP^1 processing one region's features. Finally, the responses of all the MLP^1s in the first stage are incorporated in MLP^2 to generate the final labels.

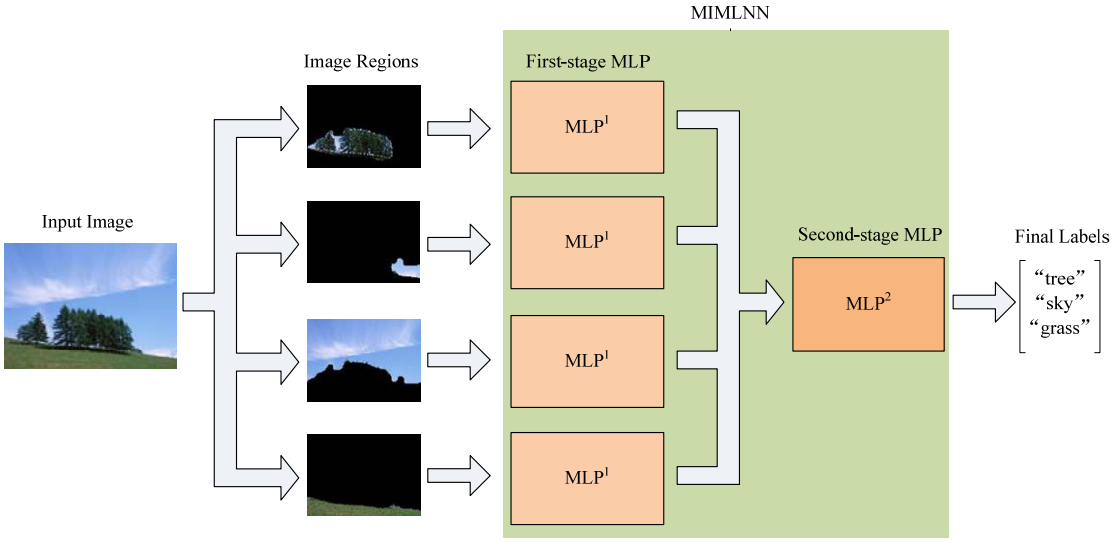


Figure 4-1. Image classification flowchart of MIMLNN.

Both MLP^1 and MLP^2 are of three layers: input layer, hidden layer and output layer. In mathematics, the MLP^1 can be formulated as follows:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{W}^b \boldsymbol{f}(\boldsymbol{W}^a \boldsymbol{x} + \boldsymbol{b}^a) + \boldsymbol{b}^b), \tag{4-1}$$

where x is the input vector; y is the output vector; $\boldsymbol{W}^a$ is the weight matrix between the input layer and the hidden layer; $\boldsymbol{W}^b$ is the weight matrix between the hidden layer and the output layer; $\boldsymbol{b}^a$ and $\boldsymbol{b}^b$ are the bias vectors of the hidden layer and output layer, respectively. Function array $\boldsymbol{f}(\cdot)$ is made of sigmoid transfer functions $f(\cdot)$. The formulation of $f(\cdot)$ is defined as:

$$f(x) = 1/(1+\mathrm{e}^{-x}). \tag{4-2}$$

The output of sigmoid function $f(\cdot)$ is a real value between 0 and 1. Similarly, we can have the formulation of MLP^2 as follows:

$$\boldsymbol{v} = \boldsymbol{f}(\boldsymbol{W}^d \boldsymbol{f}(\boldsymbol{W}^c \boldsymbol{u} + \boldsymbol{b}^c) + \boldsymbol{b}^d), \tag{4-3}$$

where the input vector $\boldsymbol{u}$ is the summation of the outputs of MLP^1s: $\boldsymbol{u} = \sum_{i=1}^{n} \boldsymbol{y}^i$, with n denoting the number of image regions. By substituting Eq. (4-1) into Eq. (4-3), we

can have the formulation of the proposed model MIMLNN:

$$\boldsymbol{v} = \boldsymbol{f}(\boldsymbol{W}^d \boldsymbol{f}(\boldsymbol{W}^c \sum_{i=1}^{n} (\boldsymbol{f}(\boldsymbol{W}^b \boldsymbol{f}(\boldsymbol{W}^a \boldsymbol{x}^i + \boldsymbol{b}^a) + \boldsymbol{b}^b)) + \boldsymbol{b}^c) + \boldsymbol{b}^d), \tag{4-4}$$

where $\boldsymbol{x}^i$ is the feature vector of the ith image region. Eq. (4-4) characterizes the input-output relationship of MIMLNN.

The output of MLP^2, $\boldsymbol{v}$, is the label vector with each entry associating with a label. Thus, vectors $\boldsymbol{y}$, $\boldsymbol{u}$, and v have the same dimension. In the training phase, $\boldsymbol{v}$ is the target binary label output of the input image, with 1 denoting a ground truth label. In the testing phase, the real-number vector $\boldsymbol{v}$ is finally converted to a binary vector by using a threshold (0.5 in our experiments), with 1 denoting a selected label and 0 denoting not. If all the nodes of $\boldsymbol{v}$ are smaller than the threshold, the node with the maximum value is selected as the final label.

The structure of MIMLNN has the following advantages. First of all, the first-stage network uses the same MLP. This structure enables MIMLNN to process images with various numbers of regions. Secondly, the function of MLP^1 is to automatically model the relationship between the image regions and labels. Furthermore, most region-based image classification methods associate a region with at most one label. By contrast, MLP^1 is a multi-output network, as a result of which an image region is allowed to have more than one label. This mechanism is more practical, since in real applications one image region may associate with multiple labels. Thirdly, the outputs of MLP^{1}s are incorporated in the second-stage MLP^2. Comparing with many other methods that model the region-label relationship, MLP^2 is able to make use of label–label information such as label correlation to refine the

classification result. The issue is how to propagate the image-level labels to the regions, that is how to train MIMLNN based on the image-level labels and region-level features. The following section will present the training procedure.

4.3 Training Algorithms of MIMLNN

4.3.1 Gradient Descent Algorithm

Gradient descent algorithm is a traditional approach to train MLP. It updates neural weights/biases according to the negative gradient of an error function. The error function can be defined as follows:

$$E = \frac{1}{2}(\boldsymbol{o} - \boldsymbol{v})^T (\boldsymbol{o} - \boldsymbol{v}), \tag{4-5}$$

where $\boldsymbol{o}$ is the target output of MIMLNN. Notation T denotes transposition. Then we can have the update rule as follows:

$$w_{ij}(t+1) = w_{ij}(t) - \lambda \frac{\partial E}{\partial w_{ij}(t)}, \tag{4-6}$$

where parameter λ is the learning rate. MLP^2 can be directly trained by applying the updating rule in Eq. (4-6). To train MLP^1, the error is back-propagated in the following way:

1. Error E is back-propagated to the MLP^1s, obtaining error $E^1, E^2, \ldots, E^n$ for all the n MLP^1s, respectively.
2. The weights/biases of MLP^1 are updated n times according to Eq. (4-6), sequentially using error $E^1, E^2, \ldots, E^n$.

Specifically, let $\boldsymbol{p}$ and $\boldsymbol{q}$ respectively denote the activations (output vectors) of the

hidden layers of MLP[1] and MLP[2]. That is, $\boldsymbol{p} = \boldsymbol{f}(\boldsymbol{W}^a \boldsymbol{x} + \boldsymbol{b}^a)$, and $\boldsymbol{q} = \boldsymbol{f}(\boldsymbol{W}^c \boldsymbol{u} + \boldsymbol{b}^c)$. Let $\boldsymbol{\delta}$, $\boldsymbol{\xi}$, $\boldsymbol{\psi}$, and $\boldsymbol{\varphi}$ respectively denote the input vectors of the four transfer functions of MIMLNN. That is, $\boldsymbol{p} = \boldsymbol{f}(\boldsymbol{\delta})$, $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{\xi})$, $\boldsymbol{q} = \boldsymbol{f}(\boldsymbol{\psi})$, and $\boldsymbol{v} = \boldsymbol{f}(\boldsymbol{\varphi})$. In addition to that, we define $\Delta\boldsymbol{p} = -\partial E / \partial \boldsymbol{p}$, $\Delta\boldsymbol{y} = -\partial E / \partial \boldsymbol{y}$, $\Delta\boldsymbol{q} = -\partial E / \partial \boldsymbol{q}$, $\Delta\boldsymbol{v} = -\partial E / \partial \boldsymbol{v}$. After simple calculation, we can obtain

$$\Delta v_i = \left(o_i - v_i\right) f'\left(\varphi_i\right), \tag{4-7}$$

$$\Delta q_i = f'\left(\psi_i\right) \sum_j \left(w_{ji}^d \Delta v_j\right), \tag{4-8}$$

$$\Delta y_i^k = f'\left(\xi_i^k\right) \sum_j \left(w_{ji}^c \Delta v_j\right), \tag{4-9}$$

$$\Delta p_i^k = f'\left(\delta_i^k\right) \sum_j \left(w_{ji}^b \Delta y_j^k\right), \tag{4-10}$$

where k denotes the kth MLP[1] copy, and $f'(\cdot)$ denotes the derivative of transfer function $f(\cdot)$. Given an input image with n regions, by expanding Eq. (4-6), we can finally have the gradient descent algorithm for training MIMLNN as described in Algorithm 1, where parameter λ is the learning rate. In our experiments, all the weights/biases are randomly initialized from the zero-mean unit-standard deviation normal distribution. The training is stopped based on the performance of the validation data. Specifically, the training will be stopped if the error on the validation data does not reduce within a certain number of iterations.

Algorithm 1. Gradient descent algorithm for training MIMLNN

Input: training dataset D_{tr}, validation dataset D_{val}

1: randomly initialize $\boldsymbol{W}^a$, $\boldsymbol{b}^a$, $\boldsymbol{W}^b$, $\boldsymbol{b}^b$, $\boldsymbol{W}^c$, $\boldsymbol{b}^c$, $\boldsymbol{W}^d$, $\boldsymbol{b}^d$

2: **repeat**

3: feed forward to compute $\boldsymbol{p}$, $\boldsymbol{y}$, $\boldsymbol{q}$, $\boldsymbol{v}$, and the error E on the training set D_{tr}

4: back-propagate to compute $\Delta\boldsymbol{p}$, $\Delta\boldsymbol{y}$, $\Delta\boldsymbol{q}$, $\Delta\boldsymbol{v}$

5: $w_{ij}^d(t+1) = w_{ij}^d(t) + \lambda\Delta v_i q_j$, $b_i^d(t+1) = b_i^d(t) + \lambda\Delta v_i$

6: $w_{ij}^c(t+1) = w_{ij}^c(t) + \lambda\Delta q_i u_j$, $b_i^c(t+1) = b_i^c(t) + \lambda\Delta q_i$

7: **for** $k = 1:n$ **do**

8: $w_{ij}^b(t+1) = w_{ij}^b(t) + \lambda\Delta y_i^k p_j^k$, $b_i^b(t+1) = b_i^b(t) + \lambda\Delta y_i^k$

9: $w_{ij}^a(t+1) = w_{ij}^a(t) + \lambda\Delta p_i^k x_j^k$, $b_i^a(t+1) = b_i^a(t) + \lambda\Delta p_i^k$

10: $w_{ij}^b(t) = w_{ij}^b(t+1)$, $b_i^b(t) = b_i^b(t+1)$

11: $w_{ij}^a(t) = w_{ij}^a(t+1)$, $b_i^a(t) = b_i^a(t+1)$

12: **end for**

13: feed forward to compute the error E_{val} on the validation et D_{val}

14: **until** E_{val} does not reduce anymore

Output: $\boldsymbol{W}^a$, $\boldsymbol{b}^a$, $\boldsymbol{W}^b$, $\boldsymbol{b}^b$, $\boldsymbol{W}^c$, $\boldsymbol{b}^c$, $\boldsymbol{W}^d$, $\boldsymbol{b}^d$

4.3.2 Long-Term Dependency Problem

A gradient descent algorithm is easy to implement. However, it suffers from the long-term dependency problem (Cho et al., 2003), especially when the network has many layers. This is because the back-propagating error is multiplied by the derivative of the sigmoid function, the value of which is between 0 and 1. Therefore, the gradient for very deep layers could become very small, resulting in that the parameters are difficult to update. Taking MIMLNN as an example, the error E has to back-propagate through four layers to reach the weights w_{ij}^a of MLP[1]. Thus, E is multiplied by the derivative of the sigmoid function for four times. In particular, the derivative of the sigmoid function is defined as follows:

$$f'(x) = f(x)\left(1 - f(x)\right) = \mathrm{e}^{-x} \Big/ \left(1 + \mathrm{e}^{-x}\right)^2 . \tag{4-11}$$

The value of $f'(\cdot)$ is between 0 and 1. According to Algorithm 1, the update-value Δw_{ij}^a of w_{ij}^a can be expanded as follows:

$$\begin{aligned}
\Delta w_{ij}^a &= \lambda \Delta p_i^k x_j^k \\
&= \lambda x_j^k f'(\delta_i^k) \sum\nolimits_j (w_{ji}^b \Delta y_j^k) \\
&= \lambda x_j^k f'(\delta_i^k) \sum\nolimits_j (w_{ji}^b f'(\xi_j^k) \sum\nolimits_l (w_{lj}^c(t) \Delta q_l)) \\
&= \lambda x_j^k f'(\delta_i^k) \sum\nolimits_j (w_{ji}^b f'(\xi_j^k) \sum\nolimits_l (w_{lj}^c(t) f'(\psi_l) \sum\nolimits_r (w_{rl}^d(t) \Delta v_r))) \\
&= \lambda x_j^k f'(\delta_i^k) \sum\nolimits_j (w_{ji}^b f'(\xi_j^k) \sum\nolimits_l (w_{lj}^c(t) f'(\psi_l) \sum\nolimits_r (w_{rl}^d(t)(o_r - v_r) f'(\varphi_r))))
\end{aligned} \tag{4-12}$$

Obviously, the error has been multiplied by $f'(\cdot)$ for four times. Thus, the update-value Δw_{ij}^a could be very small, making the weight updating inefficient.

4.3.3 Rprop Algorithm

To solve the long-term dependency problem, the refined BP algorithm Rprop (Riedmiller, H. Braun, 1993) is employed to train MIMLNN. Rprop does not take into account the derivative magnitude of the error signal for parameter updating. The sign of the derivative can determine the direction of the weight update. Generally, the update-value is increased by a factor $\eta+$ when the derivative retains its sign, and the update-value is decreased by a factor $\eta-$ when the derivative changes its sign. As a result, the derivative magnitude does not affect the weight update. Even though the gradient is very small, the weights/biases can be effectively tuned. Appendix B describes the implementation details of Rprop algorithm. Before implementing Rprop algorithm, we need to compute gradient $\partial E / \partial w_{ij}(t)$ for each weight and bias using the equation presented in Section 4.3.1. Note that Rprop is a batch training algorithm.

That is, the gradient used in Rprop is computed by averaging the gradients for all the patterns presented. As described in Appendix B, there are some parameters of Rprop that need to be set. In our experiments, we set $\Delta_0 = 0.1$, $\Delta_{\max} = 50.0$, $\Delta_{\min} = 1e^{-6}$, $\eta^+ = 1.2$, and $\eta^- = 0.5$, as suggested by Riedmiller and Braun (Riedmiller and Braun, 1993).

4.4 Experiments

4.4.1 Experimental Results on a Synthetic Image Dataset

The first dataset used to evaluate the performance of MIMLNN is a synthetic image dataset. The synthetic images are comprised of nine objects: "round", "triangle", "rectangle", "octagon", "4-point star", "5-point star", "moon", "heart", and "lighting bolt". Each object has six colors: red, green, blue, yellow, cyan, and magenta. Each synthetic image contains 1 to 4 non-overlapping objects and a white background. In order to make some label correlations, objects "round" and "triangle", "rectangle" and "octagon", "4-point star" and "5-point star", "moon" and "heart" with the same colors are designed to always appear in the same images. Figure 4-2 shows example images of the synthetic dataset.

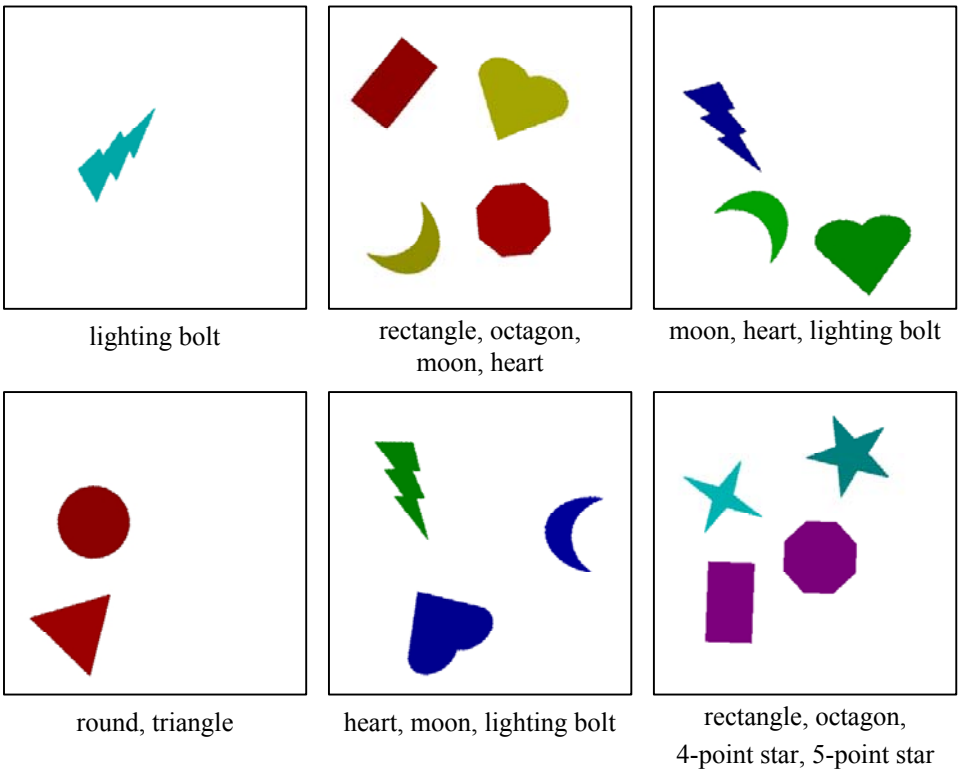


Figure 4-2. Examples of the synthetic dataset.

Each synthetic image is segmented by using JSEG (Deng and Manjunath, 2001), a widely-used segmentation technique. Since the synthetic images are quite simple, all the objects can be well segmented by JSEG, with each segmented region exactly representing one object. Therefore, we do not need to worry about the uncertainty introduced by the inaccurate segmentation. Each region is represented by seven invariant moments and the three average values of three color channels R, G, B. Note that the color features are not useful to characterize the nine objects/labels here. The three average values of RGB can be regarded as noise to the features. We take the color features into account so as to find out whether or not the proposed approach MIMLNN can distinguish the nine labels based on the noisy features. The dataset has in total 1000 images. We use 400 images for training, 400 images for validation and

200 images for testing. In our experiments, different numbers of hidden nodes of MLP^1 and MLP^2 are tested, and the one with the best performance on the validation data is retained. Each experiment is performed ten times, and the average performance is reported. To simplify the implementation, MLP^1 and MLP^2 use the same number of hidden nodes. According to the experimental results, MIMLNN achieves the best performance on the validation data when MLP^1 and MLP^2 have ten hidden nodes. Since we deal with nine objects with 10 features, the configurations of MLP^1 and MLP^2 are 10-10-9 and 9-10-9, respectively. There are various metrics for the evaluation of classification performance. For multi-label image classification, precision, recall and F-score (Goutte and Gaussier, 2005) are three widely used metrics. They are adopted to evaluate the performance of MIMLNN for the synthetic dataset. The average performance of nine classes is reported.

To investigate the effect of the second-stage network, the performance of using and without using MLP^2 is compared. The results are shown in Table 4-1. The use of MLP^2 improves the performance. This is because the synthetic dataset processes some label correlations, and MLP^2 can capture these correlations for classification refinement. Almost 100% accuracy is achieved when using MLP^2, meaning that the proposed MIMLNN classifies the synthetic images very well. In other words, MIMLNN is able to automatically associate the labels with regions based on the noisy features. The experiments on synthetic dataset demonstrate the feasibility and effectiveness of MIMLNN. However, the synthetic objects are rather simple, and the segmentation and feature extraction are ideal, which is not practical in real

applications. Thus, we use a real dataset to evaluate MIMLNN in the subsequent experiments.

Table 4-1. Performance comparison of using and without using MLP^2 on the synthetic dataset.

Method	Precision	Recall	F-score
MLP^1	0.9552	0.9542	0.9547
$MLP^1 + MLP^2$	0.9995	0.9997	0.9996

4.4.2 Experimental Results on the Corel Dataset

The Corel dataset (Duygulu et al., 2002) is a popular image classification dataset. In this chapter, we select 1000 Corel images for experiments. Ten objects/classes, namely "sky", "sun", "clouds", "tree", "people", "buildings", "plane", "bear", "snow" and "tiger" are used to label the images. Figure 4-3 shows example images of the Corel dataset. All the images are segmented using Normalized Cuts (Shi and Malik, 1997), and the maximum number of regions for one image is ten. In total 30 features (including region color and standard deviation, region average orientation energy, region size, location, convexity, first moment, and the ratio of region area to boundary length squared) adopted in (Duygulu et al., 2002) are used to characterize a region. Again, we use 400 images for training, 400 images for validation, and 200 images for testing. The number of hidden nodes is determined by the performance on the validation data. The final configurations of MLP^1 and MLP^2 are 30-20-10 and 10-20-10 (30 features and 10 labels), respectively. Each experiment is performed ten times, and the average performance is reported. For the Corel dataset, besides

precision, recall and F-score, two other popular metrics AUC (Area Under Curve) of PR (Precision-Recall) and AUC of ROC (Receiver Operating Characteristic) (Davis and Goadrich, 2006) are adopted for the performance evaluation. For the PR curve, the *x*-axis is recall and *y*-axis is precision. For the ROC curve, the *x*-axis is False Positive Rate (FPR) and *y*-axis is True Positive Rate (TPR). False Positive (FP) is that the prediction is positive and the actual value is negative, and FPR is the fraction of FP samples out of negative samples. True Positive (TP) is that the prediction is positive and the actual value is also positive. TPR is the fraction of TP samples out of positive samples. TPR is actually equivalent to recall. The AUC of ROC describes the probability that a randomly chosen positive sample is ranked higher than a randomly chosen negative sample. For both PR curve and ROC curve, a higher AUC denotes a better performance.

Figure 4-3. Examples of the Corel dataset.

For comparison purposes, MIMLBoost and MIMLSVM are implemented and

evaluated on the Corel 1000 dataset. The training and validation sets are combined as a new training set for MIMLBoost and MIMLSVM learning. The parameters of MIMLBoost and MIMLSVM are determined by twofold cross-validation. The performance of MIMLBoost, MIMLSVM, and MIMLNN are tabulated in Table 4-2, where MIMLNN (GD) denoting MIMLNN trained by gradient descent (GD) and MIMLNN (Rprop) denoting MIMLNN trained by Rprop. As can be seen from Table 4-2, MIMLNN trained by Rprop outperforms MIMLBoost and MIMLSVM for all the five metrics, especially for recall and F-score. Furthermore, compared with gradient descent, the Rprop training algorithm significantly improves the performance of MIMLNN. This experiment demonstrates the superior performance of MIMLNN and the efficiency of the Rprop algorithm.

Table 4-2. Performance of MIMLSVM, MIMLBoost, and our proposed MIMLNN using gradient descent (GD) and Rprop on the Corel dataset.

Method	Precision	Recall	F-score	AUC (RP)	AUC (ROC)
MIMLSVM	0.5415	0.3753	0.4434	0.4769	0.7991
MIMLBoost	0.5105	0.384	0.4383	0.455	0.8157
MIMLNN (GD)	0.5023	0.3622	0.4195	0.4145	0.7869
MIMLNN (Rprop)	**0.5515**	**0.4914**	**0.519**	**0.4818**	**0.8237**

Figure 4-4 shows the AUC (ROC) of MIMLNN trained by Rprop for all the ten labels. We can see that MIMLNN gains the largest AUC value on labels “sun” and “snow”. This could be explained by that the features of “sun” and “snow” are quite unique and easy to distinguish. By contrast, it is relatively difficult for MIMLNN to

recognize “tree”, because the color of “tree” is similar to “grass”, which appears in some images of the other classes.

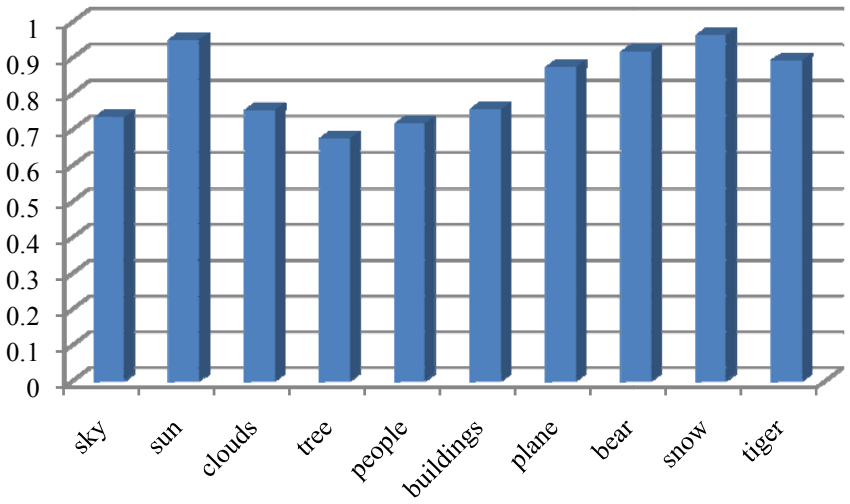


Figure 4-4. AUC (ROC) of MIMLNN for ten labels.

As presented in Section 4.2, the weights/biases are randomly initialized from the zero-mean unit-standard deviation normal distribution. In order to analyze the initialization sensitivity, we record the F-scores of MIMLNN for different initializations. As the value of sigmoid function is between 0 and 1, we only concentrate on small initial values of weights/biases, since it has been proved that large initial values of weights/biases always result in very poor performance. We analyze the initialization sensitivity in two schemes. In the first scheme, all the weights/biases are initialized to be identical, e.g., the initial values of all the weights/biases are set to the same small value. The results are shown in Table 4-3. Obviously, the performance is very bad. The training process is difficult to converge when the weights/biases are initialized to the same values. In light of the result of

scheme 1, in the second scheme, the weights/biases are randomly initialized in different symmetric intervals. The resulting average F-scores are shown in Figure 4-5. Standard deviations are depicted as the error bars. It can be seen that the initial interval that is either too small (such as [-0.1 0.1]) or too large (such as [-2 2]) does not perform well. A possible reason is that for both small and large intervals, the initial weights/biases usually locate in a region that is far from the optimal solution. Other moderate intervals achieve similar good performance. A good choice is to set the initial interval around [-0.6 0.6]. As the value of sigmoid transfer function is between 0 and 1, such a moderate interval would have a higher probability to initialize the weights/biases in a region where a good solution can be finally found.

Table 4-3. F-score of MIMLNN when the weights/biases are initialized to be identical.

Initial value	-1.0	-0.8	-0.4	0.0	0.4	0.8	1.0
F-score	0.069	0.114	0.199	0.095	0.163	0.051	0.051

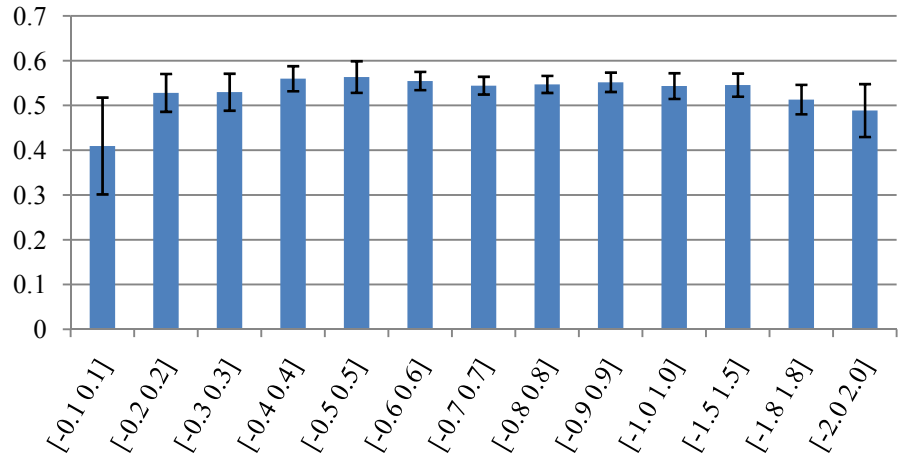


Figure 4-5. Mean F-scores and standard deviations of MIMLNN when the weights/biases are randomly initialized in different symmetric intervals.

We also investigate the performance of MIMLNN when different numbers of hidden nodes are used. The results are shown in Figure 4-6. The figure shows that at around 20 hidden nodes, MIMLNN achieves the satisfactory performance. After that, AUC (ROC) tends to be stable. F-score and AUC (PR) slightly reduce when the number of hidden nodes increases. This is probably because when the number of hidden nodes is too large, MIMLNN tends to overfit the training data, resulting in a relatively low performance on the testing data. Figure 4-7 shows some images that MIMLNN performs better than MIMLBoost and/or MIMLSVM.

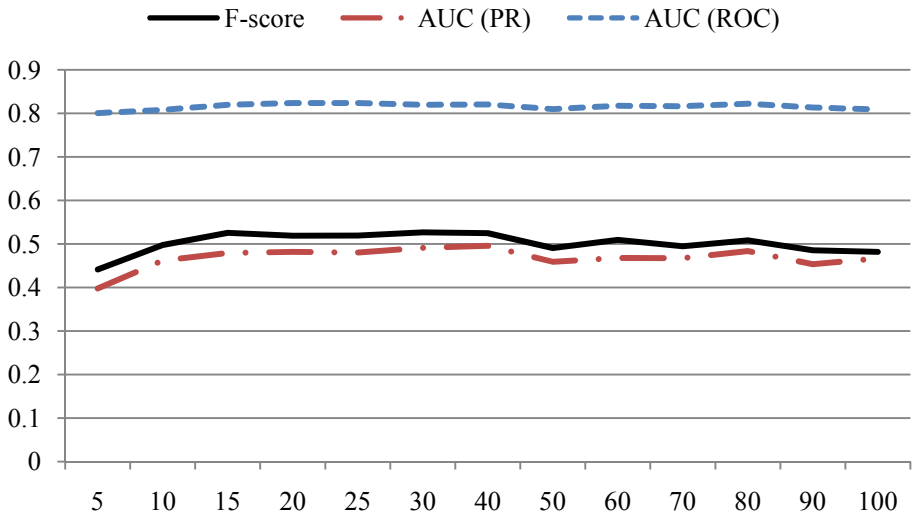


Figure 4-6. Performance of MIMLNN when different numbers of hidden nodes are used.

sky, plane sky, plane sky, plane, bear sky, snow	sky, plane sky, plane sky, plane, bear, snow sky	bear, snow bear, snow bear snow

sky sky sky, sun sky, sun	sky, sun, clouds sky, sun sky sky	people people tree buildings

people people tree tree	tree, people, buildings tree, people, buildings people people	buildings buildings tree, people tree

sky, sun, clouds, tree sky, sun, clouds sky, sun sky, sun	tiger tiger buildings tree	sky, buildings buildings sky people

Figure 4-7. Some image classification results. The first-line labels below each image are the ground truth. The second-line labels are generated by MIMLNN. The third-line and fourth-line labels are generated, respectively, by MIMLBoost and MIMLSVM.

In our approach, the label correlation is extracted by automatic learning using a neural network (MLP^2). Different from many other correlation modeling methods, the main advantage of our approach lies in that the neural network can model nonlinear, asymmetric, not only positive but also negative correlations. To demonstrate the effectiveness of the neural network method, we implement three other methods for a comparison. The first method is Spearman's rank correlation coefficient, a classical correlation coefficient measuring method that calculates statistical dependence between two variables. The second method is a Concept Association Network (CAN) which characterizes label correlation using a linear system (Fu et al., 2010). The third method is based on Graph Learning (GL) (Liu et al., 2009). The authors (Liu et al., 2009) take into account label correlations from both the training data and data collected from the Internet. In this chapter, however, we only consider correlation from the training data. These three methods are respectively termed Spearman, CAN and GL. For a fair comparison, an MLP^1 is first trained, and then the correlations extracted by the three methods are separately employed to enhance the output of MLP^1. In particular, based on the three methods, we can construct three different matrices that consist of correlation values between any two labels. The three correlation matrices are then used to multiply the output vector of MLP^1 so as to generate more accurate classification results. By contrast, our approach is like using a neural network (MLP^2) to automatically learn label correlation to improve the output of MLP^1. The results are tabulated in Table 4-4. The performance of MIMLNN is also listed in Table 4-4 for a convenient comparison. Furthermore, the result of MLP^1

without considering correlation is shown as well. We can see that all the three correlation extraction methods improve the classification performance of MLP[1]. They are, however, outperformed by our proposed MIMLNN. The reason may lie in their limitations in modeling label correlation compared with MIMLNN. Spearman can represent both positive and negative correlations. But its correlation matrix is symmetric, meaning that two labels share the same probability to occur given the other one. This is impractical since the probability of "a" to occur given "b" is usually different from that of "b" given "a", like "sky" and "clouds". CAN extracts asymmetric correlation; however, CAN characterizes linear and positive correlations only. GL can capture nonlinear correlation, but not negative one. By contrast, a neural network is able to model nonlinear, positive and negative correlations via automatic learning. This experiment demonstrates that a neural network is a promising alternative for label correlation modeling.

Table 4-4. Performance comparison among different label correlation modeling methods.

Method	MLP1	MLP1+Spearman	MLP1+CAN	MLP1+GL	MIMLNN
Precision	0.5133	0.5292	0.4802	0.521	**0.5515**
Recall	0.3642	0.3942	0.4748	0.3867	**0.4914**
F-score	0.4254	0.4508	0.4771	0.4431	**0.519**

Both gradient descent and Rprop algorithms are first-order learning algorithms. Therefore, their computational complexities both scale linearly with the number of weights and biases to be optimized, i.e., $O(n)$, with n denoting the number of weights and biases (Igel et al., 2005). Compared with gradient descent, however, Rprop

algorithm takes several more steps to calculate the weight update values. To compare the computational time of gradient decent and Rprop algorithms, we record their training times in terms of per-image per-iteration in Table 4-5. The simulations are run on MATLAB on an Intel Core2 Quad CPU 2.83 GHz, 6 GB RAM PC. Table 4-5 shows that both gradient descent and Rprop take very little time (microsecond level) to complete one iteration per-image. Rprop is slightly slower than gradient descent (time increase by 4.7% only). This is because that Rprop and gradient descent algorithms perform most of the same operations in one update cycle, like signal feedforward, error back-propagation and gradient calculation. The time of extra operations performed by Rprop occupies only a very small proportion of the whole cycle. The validation procedures of Rprop and gradient descent are the same. Thus, we do not compare the validation computational time here. In terms of memory complexity, gradient descent and Rprop algorithms are both of magnitude $O(n)$, with n denoting the number of weights and biases (Igel et al., 2005). Since gradient descent and Rprop algorithms perform most of the same variables and calculations, the memory required by Rprop algorithm is very similar to that by the gradient descent algorithm. The difference between memory storages of gradient descent and Rprop algorithms for our classification task is minor.

Table 4-5. Per-image per-iteration training time of gradient descent and Rprop algorithms.

Training algorithm	GD	Rprop	Increase by Rprop
Time (microsecond)	78.9	82.6	4.7%

We also compare the training time and testing time in terms of per-image of MIMLBoost, MIMLSVM, and MIMLNN using the Rprop training algorithm. The results are shown in Table 4-6. As we can see, MIMLNN and MIMLSVM spend a similar amount of training time. For testing, however, MIMLNN is much faster than MIMLSVM due to its feedforward propagation manner. It is worth mentioning that both the training and testing of MIMLBoost are very slow. The inefficiency of MIMLBoost is mainly caused by its time-consuming clustering and boosting procedures (Li et al., 2009).

Table 4-6. Per-image training and testing time of MIMLBoost, MIMLSVM, and MIMLNN using the Rprop algorithm.

Method	MIMLBoost	MIMLSVM	MIMLNN
Training time (second)	23.26	0.071	0.077
Testing time (second)	16.29	0.032	3.55e-4

As presented in Section 4.2, the image regions can be obtained by performing automatic segmentation or using regular gridding. The experiments reported here are based on automatic segmentation, while in the final experiment, regular gridding is investigated. The image regions are extracted on a regular grid of 3 × 3 regions, resulting in nine regions for each image. The widely-used HOG (Dalal and Triggs, 2005) feature is adopted to represent each region. The performance of MIMLNN is compared with a state-of-the-art approach SPM (Lazebnik et al., 2006) in Table 4-7. SPM achieves higher precision than MIMLNN; however, MIMLNN outperforms

SPM in terms of recall and F-score. The performance of both MIMLNN based on regular gridding and SPM is much lower than that of MIMLNN based on automatic segmentation as shown in Table 4-2. The main reason can be that the regions extracted from regular gridding are usually mixed up with different objects, so it is difficult to distinguish objects from these regions. By contrast, the regions extracted from automatic segmentation are more meaningful and easier to recognize. Furthermore, SPM is originally proposed for holistic single-class scene classification. It may not work very well for multi-label image classification.

Table 4-7. Performance of SPM and MIMLNN based on regular gridding.

Method	Precision	Recall	F-score
SPM	0.5902	0.3052	0.4024
MIMLNN	0.4412	0.4313	0.4362

4.5 Summary

In this chapter, we propose a hierarchical neural approach MIMLNN for multi-instance multi-label image classification. MIMLNN consists of two stages of MLPs. The first-stage MLP is used to establish the relationship between image regions and labels. The second-stage MLP aims at capturing label correlation for classification refinement. To solve the long-term dependency problem encountered in the traditional gradient descent algorithm, a refined back-propagation algorithm Rprop is extended to train MIMLNN. The experiments are conducted on a synthetic dataset and the popular Corel dataset. Two state-of-the-art multi-instance multi-label

learning algorithms MIMLBoost and MIMLSVM are also implemented for performance comparison. Experimental results demonstrate the superior performance of MIMLNN for multi-instance multi-label image classification.

Chapter 5 Combining Holistic and Object-Based Approaches for Scene Classification

5.1 Background

Holistic and object-based are two main strategies for scene classification. Both of them have advantages and disadvantages. When scenes have a small number of objects or simple global visual features, a holistic approach can achieve a superior performance. However, for scenes that contain multiple objects and that objects play important roles in the scene discrimination, a holistic approach may not work well, since it does not take into account the object attribute information. When scenes consist of different objects, the object-based approach is advantageous. However, if scenes are of simple contents, where individual objects may not help in scene classification too much, the object-based approach may be inefficient. Furthermore, the accuracy of regional object recognition significantly influences the final classification performance of an object-based strategy. In light of this, in this chapter, we propose to combine holistic and object-based approaches for scene classification. To our best knowledge, very little work investigates a combination of holistic and object-based approaches.

In this chapter, we first show in Section 5.2 that deep learning, a recent very hot topic of machine learning, is a promising alternative to recognize scenes in a holistic

way. After that, in Section 5.3, we propose to combine holistic and object-based approaches for scene classification.

5.2 Deep Learning for Scene Classification

In recent years, a number of deep learning algorithms have been proposed to automatically learn feature representations for full-size image recognition, e.g., Convolutional Deep Belief Networks (CDBNs) (Lee et al., 2009), Deconvolutional Networks (DN) (Zeiler et al., 2010), Hierarchical Matching Pursuit (HMP) (Bo et al., 2011), and Convolutional-Recursive Neural Networks (CRNN) (Socher et al., 2012). Amongst these algorithms, HMP has been shown to obtain state-of-the-art performance on the recognition of some types of images. Therefore, we would like to apply HMP to our natural scene classification, to see how good performance it can achieve by comparing with several widely-used hand-designed feature representation methods.

5.2.1 Hierarchical Matching Pursuit (HMP)

There are two main components in HMP: dictionary learning with K-SVD, and matching pursuit encoder. The matching pursuit encoder consists of three modules: batch tree orthogonal matching pursuit, spatial pyramid max pooling, and contrast normalization.

1) *Dictionary learning with K-SVD*: K-SVD is a state-of-the-art dictionary learning algorithm that generalizes K-means (Aharon et al., 2006). Given an

h-dimensional observations $\boldsymbol{Y}$, K-SVD aims to learn a dictionary $\boldsymbol{D} = \{\boldsymbol{d}_i\}$, where $\boldsymbol{d}_i$ is called a filter, and an associated sparse code matrix $\boldsymbol{X}$ by minimizing

$$\min_{D,X} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \qquad s.t.\ \forall i,\ \ \|\boldsymbol{x}_i\|_0 \le K \quad , \tag{5-1}$$

where $\|\bullet\|_F$ denotes Frobenius norm. $\boldsymbol{x}_i$ is the *i*th column of $\boldsymbol{X}$. $\|\bullet\|_0$ is zero-norm that counts the non-zero entries in the sparse code $\boldsymbol{x}_i$. K is the sparsity level that bounds the number of non-zero entries. Problem Eq. (5-1) is solved in an alternating way. In the first stage, $\boldsymbol{D}$ is fixed, and only sparse code matrix $\boldsymbol{X}$ is optimized. The problem is then converted into the following *n* sub-problems:

$$\min_{x_i} \|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}_i\|^2 \qquad s.t. \quad \|\boldsymbol{x}_i\|_0 \le K\ . \tag{5-2}$$

Problem Eq. (5-2) is non-convex, but its approximation solution can be found using orthogonal matching pursuit that is introduced later. In the second stage, both $\boldsymbol{D}$ and its associated sparse coefficients are updated simultaneously using Singular Value Decomposition (SVD). It is worth mentioning that if the sparsity level K is set to 1 and the sparse code matrix $\boldsymbol{X}$ is forced to be a 0-1 binary matrix, K-SVD becomes K-means algorithm.

2) *Batch tree orthogonal matching pursuit*: The approximation solution for Eq. (5-2) is computed using orthogonal matching pursuit (OMP) (Pati et al., 1993) in a greedy manner. The basic idea of OMP is as follows. At each step, the filter with the highest correlation to the current residual is selected. Then the observation is orthogonally projected to all the previously selected filters and the residual is recomputed. The procedure is repeated until we gain K filters. The details of OMP can be found in (Pati et al., 1993). A problem with OMP is that the computation becomes

infeasible for a very large dictionary. In light of this, batch tree orthogonal matching pursuit (BTOMP) organizes the dictionary using a tree structure to accelerate the computation. Specifically, BTOMP uses K-means to group the dictionary into a number of sub-dictionaries, and associates the sub-dictionaries with a number of learnt centers. There are two steps for selecting the filters (Bo et al., 2011): (1) select the center that best matches the current residual, and (2) choose the filter within the sub-dictionary associated with this center. BTOMP has been shown to be more efficient than OMP.

3) *Spatial pyramid max pooling*: Spatial pyramid max pooling decomposes an image patch into multiple levels. At each level, the patch is partitioned into different numbers of cells. Level 0 has only one cell, that is, the whole patch. At level 1, the patch is partitioned into four quadrants, and so on. The features of each cell C are the component-wise maxima over all sparse codes within a cell:

$$\boldsymbol{F}(C) = \left[\max_{i \in C} |x_{i1}|, ..., \max_{i \in C} |x_{im}| \right], \tag{5-3}$$

where i ranges over all entries in the cell, and x_{im} is the m-th component of the sparse code vector $\boldsymbol{x}_i$. Therefore, $\boldsymbol{F}(C)$ has the same dimensionality as the sparse codes. Let L be the number of levels, N_l the number of cells at level l. Then we can have the features for a patch P as follows:

$$\boldsymbol{F}(P) = \left[\boldsymbol{F}\left(C_1^1\right), ..., \boldsymbol{F}\left(C_l^{N_l}\right), ..., \boldsymbol{F}\left(C_L^{N_L}\right) \right] . \tag{5-4}$$

The dimensionality of $\boldsymbol{F}(P)$ is equal to the dimensionality of the sparse codes multiplying the total number of cells in the patch.

4) *Contrast normalization*: In order to handle various magnitudes of sparse codes

that are caused by local variations in illumination and foreground-background contrast, contrast normalization is adopted in HMP. For a patch *P*, the contrast normalization can be formulated as follows:

$$\overline{\boldsymbol{F}}(P) = \frac{\boldsymbol{F}(P)}{\sqrt{\|\boldsymbol{F}(P)\|^2 + \varepsilon}} \quad , \tag{5-5}$$

where ε is a small positive number.

HMP consists of multiple layers. The next layer is built on top of the outputs of the match pursuit encoder in the current layer. In the training phase, once a layer is trained, its dictionary is fixed, and its outputs are used as the input to the next layer for the training of the next layer's match pursuit encoder.

5.2.2 Experiments

5.2.2.1 Natural Scene Dataset

The experiments are carried out on the scene dataset used in (Vogel and Schiele, 2007). The dataset has 700 images of six scenes: *coasts*, *rivers/lakes*, *forest*, *plains*, *mountains*, and *sky/clouds*. Figure 5-1 shows sample images of the six scenes. We choose this dataset for experiments mainly because that the scenes are difficult to recognize. The performance of various scene feature representations can be thus well evaluated. As shown in Figure 5-1, for instance, scenes *rivers/lakes* and *coasts* consist of similar objects such as water, rocks, sky etc., and therefore possess similar visual features. If a feature representation method wants to accurately distinguish these scenes, it should be able to describe higher level features like sea water and river

water, rather than capture basic visual features only.

Figure 5-1. Sample images of the scene dataset.

5.2.2.2 Configurations of HMP

We modified the codes of HMP provided by the authors (Bo et al., 2011) to suit our task. The HMP we used has two layers. Only image gray intensity is considered to build the dictionaries. The dictionary size of the first layer is 75, and the size of the second layer is 1000. The sparsity level K for two layers is set to 5 and 10, respectively.

5.2.2.3 Experimental Results

The dataset is divided into ten folds. At each time, one fold is for testing and the remaining nine folds are for training. The average performance of ten testing folds is reported. For a fair comparison, SVM (Chang and Lin, 2011) with RBF kernel is employed to classify the features generated by different feature representation methods. The parameters of SVM are determined by two-fold cross-validation on the

training data.

The classification performance of HMP and the three hand-designed feature representations are shown in Figure 5-2. Evidently, HMP outperforms all the three hand-designed representations, achieving the highest classification accuracy. Among the three hand-designed representations, SPM obtains the best performance.

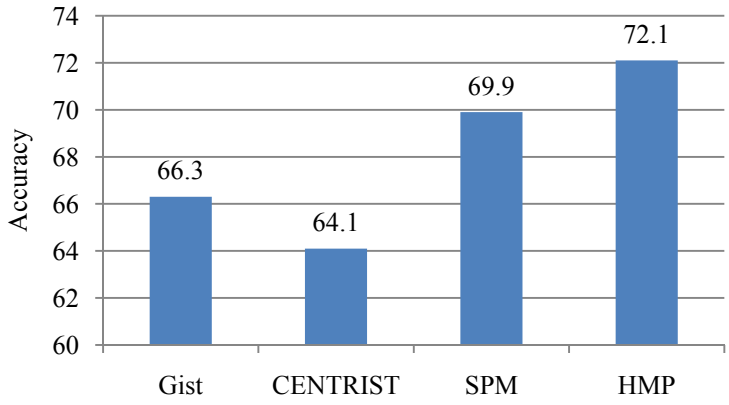


Figure 5-2. Classification accuracy using different feature representations.

To further analyze the performance of different methods on individual scenes, we display the classification confusion tables of four methods in Figure 5-3 to Figure 5-6, respectively. The values larger than ten are tagged in the confusion tables. As shown in the confusion tables, *rivers/lakes* are hard to recognize. A number of *plains* and *rivers/lakes* images are frequently misclassified to *coasts* and *mountains*. This is mainly because that *rivers/lakes* has similar features with *coasts* and *mountains*. Another poorly classified scene is *plains*, which are frequently misclassified to *coasts* and *mountains* as well. Compared to the hand-designed feature representations, HMP performs much better on *plains*. One main difference between *plains* and other scenes is that the contents of *plains* are rather flat and open. Achieving a good accuracy on

plains means that HMP is able to learn more abstract features.

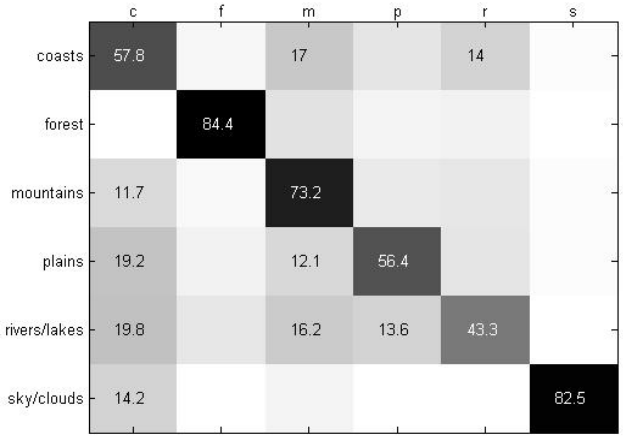


Figure 5-3. Confusion table of Gist.

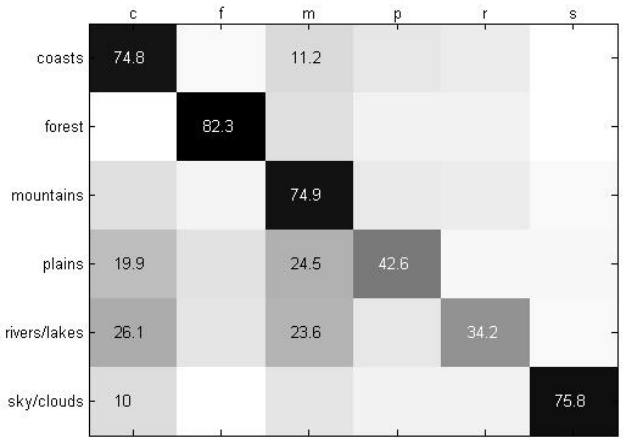


Figure 5-4. Confusion table of CENTRIST.

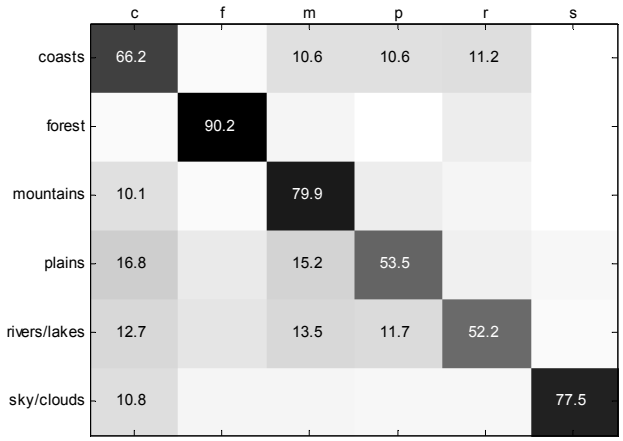


Figure 5-5. Confusion table of SPM.

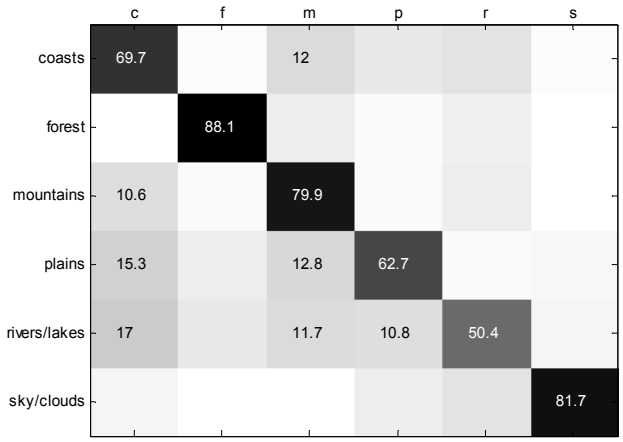


Figure 5-6. Confusion table of HMP.

Figure 5-7 shows some images that are misclassified by HMP. It can be seen that most of these images have great scene ambiguities. It is even difficult for human beings to recognize these scenes without careful examination. This indicates that although HMP performs better than other methods, it is still not able to handle the scene ambiguities very well. It is therefore desirable to explore higher level and more abstract features which can well overcome scene ambiguities.

Figure 5-7. Some images misclassified by HMP. The labels underneath each image specifies the ground truth (left) and the classification result (right).

5.2.3 Remarks

The experimental results demonstrate that deep learning is a promising alternative for obtaining features for scene classification. Most of those images misclassified by HMP have strong scene ambiguities, which are even not easy for human beings to recognize. The experiments also reflect that global features are not powerful enough to tackle scene ambiguities. In order to well solve scene ambiguities, we should take into account not only the objects that would appear in an image, but also the object properties, e.g., object locations, object sizes, etc.

5.3 Improving Scene Classification by Combining Holistic and Object-Based Approaches

In this section, two popular holistic approaches and an object-based approach are introduced. A simple scheme to combine holistic and object-based approaches is discussed. If the decisions of holistic and object-based approaches are the same, the scene class agreed by them is selected as the final decision. Otherwise, a majority voting scheme will be employed to make the final decision based on the results of all the classifiers of both holistic and object-based approaches. Specifically, both the holistic and object-based classification approaches are implemented using an SVM classifier via one-versus-one scheme. In the decision stage, if the holistic and object-based approaches do not agreed with each other, all the SVM classifiers will be involved to vote for the final scene class through the majority voting principle. That is,

the scene class having the maximum number of votes will be selected.

5.3.1 Methodology

5.3.1.1 Holistic Scene Classification

A number of holistic image representations have been proposed (Lazebnik et al., 2006; Wu and Rehg, 2011; Oliva and Torralba, 2001; Dalal and Triggs, 2005). Here we utilize two state-of-the-art approaches: Spatial Pyramid Matching (SPM) (Lazebnik et al., 2006) and CENTRIST (Wu and Rehg, 2011).

SPM characterizes the visual features of an image using a number of SIFT descriptors (Lowe, 1999). Then a vocabulary of visual words is built based on the SIFT descriptors using K-means clustering. After that, each SIFT descriptor can be represented by the visual words. Finally, the visual words are concatenated through spatial pyramid to represent the images. Figure 5-8 shows the spatial pyramid split of an image. There are three levels: level 0, level 1, and level 2, respectively, splitting the image into 1 block, 4 blocks, and 16 blocks. Hence, in total 21 blocks are obtained. If the vocabulary contains w visual words, then the image is finally represented by a $w \times 21$ dimension feature vector.

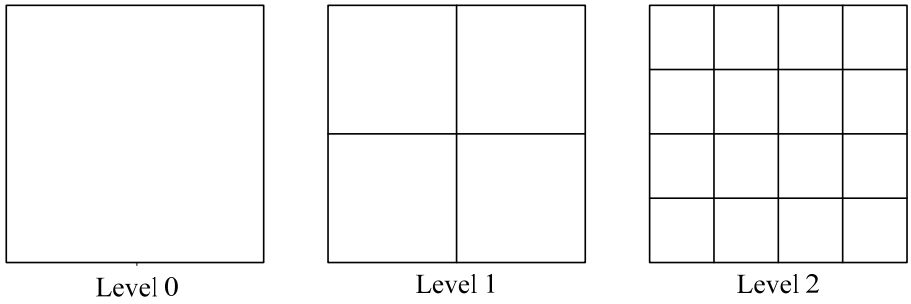


Figure 5-8. Spatial pyramid partition of an image.

CENTRIST represents an image using the histogram of Census Transform (CT). CT compares the intensity value of a pixel with its eight neighboring pixels. If a pixel's intensity is larger than or equal to that of one of its neighbors, a bit 1 is set in the corresponding neighbor location. Otherwise, a bit 0 is set. The eight bits are then collected from left to right, top to bottom to form a binary number, which is consequently converted to a decimal number in [0 255]. The decimal number is the CT value for the center pixel. Figure 5-9 shows an example to calculate the CT value for the center pixel. CENTRIST is the histogram of the CT values of all the pixels.

$$\begin{array}{cccccc} 60 & 58 & 40 & 1 & 1 & 0 \\ 21 & 52 & 92 \Rightarrow 0 & & & 1 \Rightarrow (11001100)_2 \Rightarrow \mathrm{CT} = 204 \\ 88 & 46 & 33 & 1 & 0 & 0 \end{array}$$

Figure 5-9. Calculation procedure of CT value.

5.3.1.2 Object-Based Scene Classification

An object-based approach first recognizes objects appearing in the image, and then classifies the image based on the object co-occurrence distribution. Spatial information and object correlation can be explored to refine the classification

accuracy (Vogel and Schiele, 2007; Cheng and Wang, 2010). In this work, the object-based approach is implemented as follows. (1) Partition an image into 10 × 10 regular regions, which is the same scheme used in (Vogel and Schiele, 2007). For each region, recognize the objects. (2) Consider three parts of the image: top, middle, and bottom. Calculate the area ratio dominated by each object for each image part. (3) concatenate the area ratio features of three parts for the image representation. Use this representation to conduct scene classification. Suppose that there are n semantic objects considered. Then for each image part, an n-dimensional feature vector is attained, and in total an image is represented by a $3n$-dimensional feature vector.

5.3.1.3 Combination Scheme

The combination scheme is shown in Figure 5-10. First of all, an input image is classified by the holistic and object-based approaches separately. The decisions of them are then compared. If the classification results of two approaches are the same, the scene class agreed by them is selected as the final decision. Otherwise, all the classifiers of the holistic and object-based approaches are used to vote for the final decision through the majority voting principle. That is, the class that gains the maximum number of votes is chosen as the scene type of the input image.

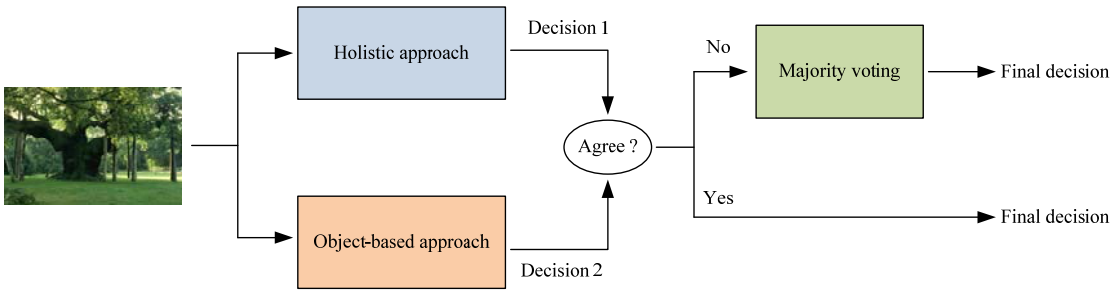


Figure 5-10. Scene classification flowchart of the proposed approach.

In this work, we employ SVM (Chang and Lin, 2011) as the classifier. One-versus-one scheme is used. Suppose that there are in total k scene classes. We have $k(k-1)/2$ SVM classifiers for either holistic or object-based approaches. In the voting phase, $k(k-1)/2\times 2 = k(k-1)$ classifiers will be involved, and the final scene class is

$$c_{final} = \arg\max_{c_j} \sum_{i=1}^{k(k-1)} \delta(y_i, c_j), \tag{5-6}$$

where $1 \le j \le k$, y_i is the decision of classifier i. $\delta(y_i, c_j) = 1$ if $y_i = c_j$; $\delta(y_i, c_j) = 0$ if $y_i \neq c_j$.

5.3.2 Experiments

5.3.2.1 Scene Dataset

The natural scene dataset introduced in Section 5.2.2.1 is used. For the object-based approach, nine semantic objects are involved: *sky*, *water*, *grass*, *trunks*, *foliage*, *fields*, *rocks*, *flowers*, and *sand*. An image region is represented by an 82-dimensional feature vector. The feature vector includes a 50-bin HSV histogram, an 8-bin edge direction histogram, and 24 features of the gray-level co-occurrence

matrix (32 gray levels) (contrast, energy, entropy, homogeneity, inverse different moment and correlation for the displacements $\overrightarrow{1,0}$, $\overrightarrow{1,1}$, $\overrightarrow{0,1}$, $\overrightarrow{-1,1}$).

5.3.2.2 Experimental Results

The dataset is divided into ten folds. Each time one fold is for testing and the remaining nine folds are for training. The average performance of ten folds testing data is reported. The parameters of all the SVM classifiers are determined by two-fold cross-validation on the training data. The final reported performance is the average accuracy of six classes, which is tabulated in Table 5-1.

Table 5-1. Accuracy of different approaches (OB denotes object-based approach).

Method	SPM	CENTRIST	OB	SPM+OB	SPM+CENTRIST+OB	SPM+CENTRIST
Accuracy	0.708	0.628	0.703	0.723	0.741	0.666

As can be seen from Table 5-1, the combination of SPM and object-based (OB) methods achieves a higher accuracy than either SPM or the object-based approach. This means that the combination improves the performance. In addition to the combination of SPM and OB, we also investigate the combination of SPM, CENTRIST and OB, the result of which is also shown in Table 5-1. Evidently, the combination obtains a better performance than all the three individual methods. Furthermore, combining three methods makes a larger improvement than combining SPM and OB only. This indicates that the more methods to combine, the better performance we can achieve. One interesting thing is that the combination of SPM and CENTRIST only achieve an accuracy between SPM and CENTRIST. The reason

could be that both SPM and CENTRIST are holistic methods. They represent images in a similar way. Therefore, they cannot complement each other and no improvement is made. So we can conclude that the combination should include both holistic and object-based methods.

In our combination scheme, if the methods to combine agree with each other, the agreed class will be outputted. This scheme works well only if the methods achieve high accuracy on the agreed images. To analyze this situation, we show the accuracy on the agreed images for different combinations of methods in Figure 5-11. It can be seen that for all the three combinations (SPM+OB, SPM+CENTRIST+OB, SPM+CENTRIST), the accuracy on agreed images is significantly higher than any individual methods shown in Table 5-1. Moreover, the accuracy increase of either SPM+OB or SPM+CENTRIST+OB is much larger than that of SPM+CENTRIST. This is also because that both SPM and CENTRIST are holistic methods. It is more likely that they make the same mistakes in scene classification.

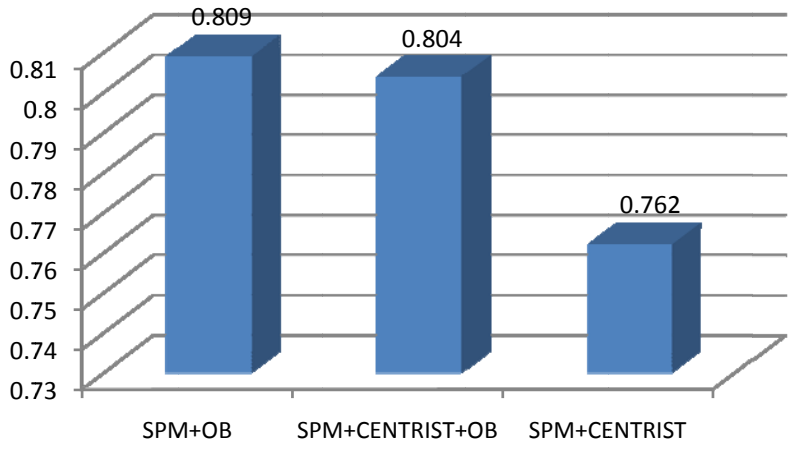


Figure 5-11. Accuracy on agreed images by different methods.

Figure 5-12 and Figure 5-13 show the confusion tables of the combinations SPM+OB and SPM+CENTRIST+OB, respectively. We can see from both figures that, scene *rivers/lakes* has the worst performance. Many images belonging to *rivers/lakes* are classified to *coasts* and *mountains*. This is mainly because that *rivers/lakes* has similar features with *coasts* and *mountains*. For example, *rivers/lakes* usually consists of water, trees, and rocks. These objects are also the main components of *coasts* and *mountains*. As a result, it is difficult for holistic and object-based approaches to distinguish them. Another poorly classified scene is *plains*, which is also frequently misclassified to *coasts* and *mountains*. Again, the misclassification is caused by that they consist of similar objects.

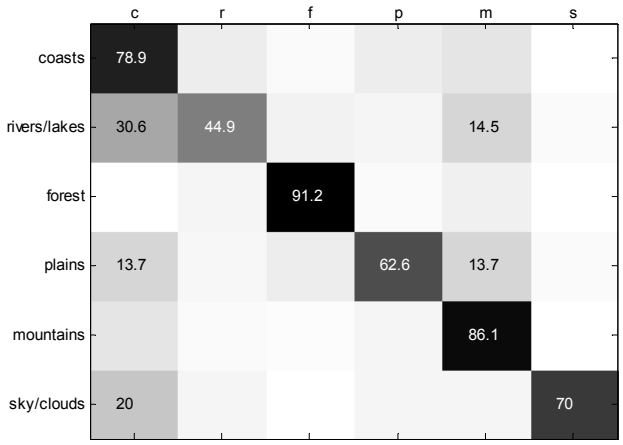


Figure 5-12. Confusion table of the combination SPM+OB.

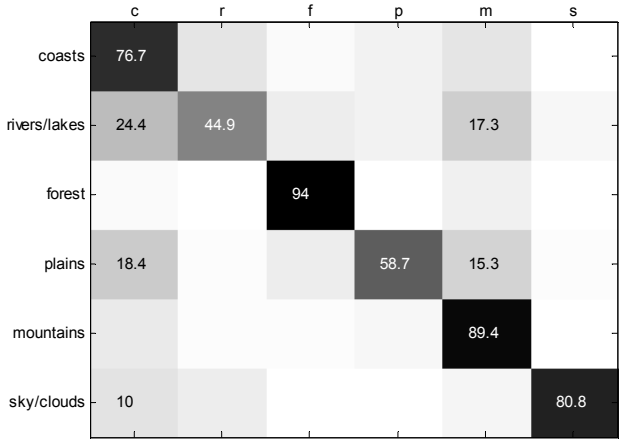


Figure 5-13. Confusion table of the combination SPM+CENTRIST+OB.

5.3.3 Remarks

The experimental results reported in this section demonstrate that the combination of holistic and object-based approaches achieves a better performance than any individual approach. Moreover, the performance improvement would be more significant if more approaches are to be combined.

5.4 Summary

Holistic and object-based approaches are two main strategies to tackle scene classification/categorization. In this chapter, we first demonstrate that Hierarchical Matching Pursuit (HMP), a state-of-the-art deep learning algorithm, is a promising holistic approach for scene classification, by comparing to other widely-used holistic approaches. After that, we propose to combine holistic and object-based approaches for scene classification. In particular, a scene image is classified respectively by holistic and object-based approaches. If holistic and object-based approaches agree

with each other (their results are identical), the scene class agreed by them is selected as the final decision. Otherwise, a majority voting scheme is employed to make the final decision based on the results of all the classifiers of both holistic and object-based approaches. We show that combining holistic and object-based approaches improves the scene classification performance.

Chapter 6 Conclusion and Future Work

6.1 Conclusions of the Thesis

In this thesis, we investigate hierarchical architectures and learning algorithms for multi-label image classification and scene categorization. The thesis can be concluded as follows.

In Chapter 3, we propose an adaptive recognition model (ARM) based on neural networks for image annotation. The proposed model uses a two-layer perceptron to learn label correlation from training images. The label correlation learnt can be nonlinear, asymmetric, both positive and negative. A proposed synthetic image dataset and a real image dataset are employed to evaluate the performance of the ARM. The experimental results demonstrate that the ARM achieves promising performance for image annotation. The ARM can effectively learn the label correlation of training images. Furthermore, the computational time of an ARM is insensitive to the number of regions of the input image and vocabulary size. In addition to that, it can be concluded from the experimental results about label correlation that, 1) the label correlation play an important role in annotation improvement only if the dataset have strong correlative information among different labels; 2) the label correlation extracted from training images is helpful only if the images to be annotated have the similar label correlation patterns; 3) the label correlation plays more important role when a visual classifier has a poorer performance.

The limitation of the model proposed in Chapter 3 is that regional ground truth is required in training. To overcome the limitation, in Chapter 4, we propose a new model (MIMLNN) for multi-instance and multi-label image classification, and develop a training algorithm based on the error back-propagation method to tune the model parameters. The new model can automatically learn the mappings between regional visual features and labels, as well as label correlation using regional visual features and the image ground truth of training images. In light of that a gradient descent algorithm suffers from the long-term dependency problem, we extend an advanced back-propagation algorithm Rprop to effectively train the model. The experiments are conducted on a proposed synthetic image dataset and the popular Corel dataset. The results show that MIMLNN achieves encouraging performance by comparing to state-of-the-art algorithms for multi-instance multi-label image classification. The results suggest that MIMLNN using 20 hidden nodes and setting the initial weights to the interval around [-0.6 0.6] can attain the best performance after training. For label correlation modeling, an MLP outperforms Spearman's rank correlation coefficient measuring method, a graph learning algorithm and a linear system, since an MLP can characterize nonlinear, asymmetric, both positive and negative label correlations. Furthermore, the experimental results also demonstrate that MIMLNN prefers proper segmentation rather than regular gridding, as segmentation can extract more meaningful regions, as a result of which MIMLNN achieves a higher correct classification rate.

In Chapter 5, we propose to combine holistic and object-based approaches for

scene classification. In particular, firstly, we employ a state-of-the-art deep learning algorithm to classify natural scenes in a holistic way. Secondly, we propose to combine holistic and object-based approaches for scene classification. The experimental results demonstrate that the deep learning algorithm is a powerful holistic approach for scene classification. Most of those images misclassified by the deep learning algorithm have strong scene ambiguities, which are even not easy for human beings to recognize. More importantly, the experiments show that combining holistic and object-based approaches improves the performance of scene classification. Moreover, the performance improvement would be more significant if more approaches are to be combined.

6.2 Future Research Directions

More work can be pursued along the line of our research, which is discussed below.

1) The model MIMLNN proposed in Chapter 4 achieves promising performance for small-scale multi-label image classification tasks. In our experiments, however, we find that the accuracy of MIMLNN would dramatically decrease when the dataset is in a large scale, i.e., large numbers of images and labels. The problem lies in the training. Although Rprop is used, the training of MIMLNN would become inefficient because the error still needs to be back-propagated thought many layers. This is a common problem in the training of deep architectures using a traditional error back-propagation method. Thus, we would like to adopt

recently-developed deep learning algorithms to effectively train MIMLNN for large-scale multi-label image classification. There are two possible approaches to realize it. The first approach is to use the training algorithm of deep belief networks. That is, MIMLNN is pretrained by the greedy layer-wise pretraining algorithm, and then a gradient descent algorithm is adopt to fine-tune the model. The second approach is to replace the multi-layer perceptrons in MIMLNN with convolutional neural networks (LeCun et al., 1989). Convolutional neural networks are capable of processing spatial information, since convolutional operators have been used. Then the model could be trained just using the training algorithms of convolutional neural networks.

2) The second direction is to develop an advanced object-based scene classification system. As introduced in Chapter 2, previous object-based approaches classify a region to a label/concept. In our proposed direction, we would like to classify each image pixel to a label to construct a label map for the image. The label map is then quantified, with each pixel label being represented by a real number. Finally deep neural networks are exploited to learn the quantified label map and predict scene classes.

3) The third direction concentrates on scene parsing, which is to label each image pixel. We would like to explore a deep neural network to carry out this task. In particular, the label maps of training images are quantified using the algorithm proposed in Direction 2. The deep neural network has the same number of input nodes and output nodes, with the number equal to the number of pixels in an

image. In the training phase, the deep neural network receives the raw image pixels as inputs and quantified label map as target outputs. In the testing phase, the deep neural network is expected to generate a quantified label map based on an image's raw pixels. The quantified label map is finally converted to a label map, such that each image pixel is assigned a label.

As the research being conducted, we would explore more significant topics.

Appendix A: List of Synthetic Labels

Table A-1. Total 54 synthetic labels and their corresponding entry numbers in the label vector.

Entry number	Label name	Entry number	Label name
1	round_red	28	4-point star_yellow
2	round_green	29	4-point star_cyan
3	round_blue	30	4-point star_magenta
4	round_yellow	31	5-point star_red
5	round_cyan	32	5-point star_green
6	round_magenta	33	5-point star_blue
7	triangle_red	34	5-point star_yellow
8	triangle_green	35	5-point star_cyan
9	triangle_blue	36	5-point star_magenta
10	triangle_yellow	37	moon_red
11	triangle_cyan	38	moon_green
12	triangle_magenta	39	moon_blue
13	rectangle_red	40	moon_yellow
14	rectangle_green	41	moon_cyan
15	rectangle_blue	42	moon_magenta
16	rectangle_yellow	43	heart_red
17	rectangle_cyan	44	heart_green
18	rectangle_magenta	45	heart_blue
19	octagon_red	46	heart_yellow
20	octagon_green	47	heart_cyan
21	octagon_blue	48	heart_magenta
22	octagon_yellow	49	lighting bolt_red
23	octagon_cyan	50	lighting bolt_green
24	octagon_magenta	51	lighting bolt_blue
25	4-point star_red	52	lighting bolt_yellow
26	4-point star_green	53	lighting bolt_cyan
27	4-point star_blue	54	lighting bolt_magenta

Appendix B: Rprop Algorithm

Given a predefined error signal *E*, we can have the Rprop update rule for all weights and biases as follows (Riedmiller and Braun, 1993).

if $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) > 0)$ **then**{

$\Delta_{ij}(t) = \min(\Delta_{ij}(t-1) * \eta^{+}, \Delta_{\max})$

$\Delta w_{ij}(t) = -\text{sign}(\frac{\partial E}{\partial w_{ij}}(t)) * \Delta_{ij}(t)$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

}

else if $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) < 0)$ **then**{

$\Delta_{ij}(t) = \max(\Delta_{ij}(t-1) * \eta^{-}, \Delta_{\min})$

$w_{ij}(t+1) = w_{ij}(t) - \Delta w_{ij}(t-1)$

$\frac{\partial E}{\partial w_{ij}}(t) = 0$

}

else if $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) = 0)$ **then**{

$\Delta w_{ij}(t) = -\text{sign}(\frac{\partial E}{\partial w_{ij}}(t)) * \Delta_{ij}(t)$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

}

In the Rprop algorithm, there are three different weight updating rules depending on three different conditions. Condition 1 indicates that if the derivative of the error signal retains its sign, the update-value is increased by a factor $\eta+$ in order to accelerate the convergence. Condition 2 indicates that if the derivative of the error signal changes its sign, which means that the last update was too big and the algorithm has jumped over a local minimum, the update-value is decreased by a factor

$\eta -$, and the derivative $\partial E/\partial w_{ij}(t)$ is set to zero so that the update-value would not be double punished. Condition 3 is to handle the case when the last derivative $\partial E/\partial w_{ij}(t) = 0$ as set in Condition 2. These conditions ensure that the parameter updating takes into account the derivative sign only. The derivative magnitude does not influence the parameter updating. As a result, the long-term dependency problem can be solved.

References

Aharon, M., Elad, M., and Bruckstein, A. (2006). "K-SVD: An Algorithm for Designing Overcomplete Dictionaries For Sparse Representation," *IEEE Transactions on Signal Processing*, Vol. 54, No. 11, pp. 4311–4322.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). "Speeded-up Robust Features (SURF)," *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346-359.

Bengio, Y. (2009). "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, Vol. 2, No. 1, pp. 1-127.

Bengio, Y. and Glorot, X. (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks," *Proceedings of AISTATS*, May13-15, Sardinia, Italy, pp. 249-256.

Bo, L., Ren, X., and Fox, D. (2011). "Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms," *Proceedings of the Conference on Advances in Neural Information Processing Systems*, December 5-10, Granada, Spain.

Boutell, M., Luo, J., Shen, X., and Brown, C. (2004). "Learning Multi-Label Scene Classification," *Pattern Recognition*, Vol. 37, No. 9, pp. 1757–1771.

Burse, K., Yadav, R. N., and Shrivastava, S. C. (2010). “Channel Equalization Using Neural Networks: A Review,” *IEEE Transactions on Systems, Man, and Cybernetics - Part C:Applications and Reviews*, Vol. 40, No. 3, pp. 352–357.

Chang, C. C. and Lin, C. J. (2011). “LIBSVM : A Library for Support Vector Machines,” *ACM Transactions on Intelligent Systems and Technology*, Vol.2 No.3, pp. 27:1-27:27, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Chang, P. C., Fan, C. Y., and Liu, C. H. (2009). “Integrating A Piecewise Linear Representation Method and A Neural Network Model for Stock Trading Points Prediction,” *IEEE Transactions on Systems, Man, and Cybernetics - Part C:Applications and Reviews*, Vol. 39, No. 1, pp. 80–92.

Chen, C. H., Lin, C. J., and Lin, C. T. (2009). “Nonlinear System Control Using Adaptive Neural Fuzzy Networks Based on A Modified Differential Evolution,” *IEEE Transactions on Systems, Man, and Cybernetics - Part C:Applications and Reviews*, Vol. 39, No. 4, pp. 459–473.

Chen, Y. and Wang, J. Z. (2004). “Image Categorization by Learning and Reasoning with Regions,” *Journal of Machine Learning Research*, Vol. 5, pp. 913–939.

Chen, Y., Bi, J., Wang, J. Z. (2006). “Miles: Multiple-Instance Learning via Embedded Instance Selection,” *IEEE Transactions on Pattern Analysis and Machine*

Intelligence., Vol. 28, No. 12, pp. 1931–1947.

Cheng, H. H. and Wang, R. S. (2010) "Semantic Modeling of Natural Scenes Based on Contextual Bayesian Networks," *Pattern Recognition*, Vol. 43, No. 12, pp. 4042-4054.

Cho, S. Y. and Chi, Z. (2005). "Genetic Evolution Processing of Data Structures for Image Classification, " *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 2, pp. 216 – 231.

Cho, S. Y., Chi, Z., Siu, W. C., and Tsoi, A. C. (2003). "An Improved Algorithm for Learning Long-Term Dependency Problem in Adaptive Processing of Data Structures, " *IEEE Transactions on Neural Networks*, Vol. 14, No. 4, pp. 781–793.

Dalal, N., and Triggs, B. (2005). "Histograms of Oriented Gradients for Human Detection, " *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 20-26, San Diego, U.S.A., pp. 886-893.

Davis, J. and Goadrich, M. (2006). "The Relationship Between Precision-Recall and ROC Curves," *Proceedings of International Conference on Machine Learning*, June 25-29, Pennsylvania, U.S.A., pp. 233-240.

Deng Y. and Manjunath, B. S. (2001). "Unsupervised Segmentation of Color-texture Regions in Images and Video," *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, Vol. 23, No. 8, pp. 800–810.

Duygulu, P., Barnard, K., de Freitas, N., and Forsyth, D. A. (2002). "Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary," *Proceedings of the 7th European Conference on Computer Vision*, May 27 - June 2, Copenhagen, Denmark, pp. 97-112.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). "Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 8, pp. 1915-1929.

Fei-Fei L. and Perona, P. (2005). "A Bayesian Hierarchical Model for Learning Natural Scene Categories," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 20-26, San Diego, U.S.A., pp. 524-531.

Feng, S., Bao, H., Lang, C., and Xu, D. (2011). "Combining Visual Attention Model with Multi-Instance Learning for Tag Ranking," *Neurocomputing*, Vol. 74, No. 17, pp. 3619–3627.

Feng, S. L., Manmatha, R., and Lavrenko, V. (2004). "Multiple Bernoulli Relevance Models for Image and Video Annotation," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 27 - July 2, Washington, U.S.A., pp.1002-1009.

Fu, H., Chi, Z., and Feng, D. (2006). "Attention-Driven Image Interpretation with Application to Image Retrieval," *Pattern Recognition*, Vol. 39, No. 9, pp. 1604–1621.

Fu, H., Chi, Z., and Feng, D. (2009). "An Efficient Algorithm for Attention-Driven Image Interpretation from Segments," *Pattern Recognition*, Vol. 42, No. 1, pp. 126–140.

Fu, H., Chi, Z., and Feng, D. (2010). "Recognition of attentive objects with a concept association network for image annotation," *Pattern Recognition*, Vol. 43, No. 10, pp. 3539-3547.

Ghoshal, A., Ircing, P., and Khudanpur, S. (2005). "Hidden Markov Models for Automatic Annotation and Content-Based Retrieval of Images and Video," *Proceedings of the 28th ACM SIGIR Conference on Research and Development in Information Retrieval*, August 15-19, Salvador, Brazil, pp. 544-551.

Goutte, C. and Gaussier, E. (2005). "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication For Evaluation," *Proceedings of European Conference on Information Retrieval, Lecture Notes in Computer Science*, March 21-23, Santiago de Compostela, Spain, Vol. 3408, pp. 345–359.

Hinton, G. E. and Salakhutdinov, R. R. (2006). "Reducing the Dimensionality of Data with Neural Networks," *Science*, Vol. 313, No. 5786, pp. 504-507.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Computation*, Vol. 18, No. 7, pp. 1527-1554.

Igel, C., Toussaint, M., and Weishui, W. (2005). “Rprop using the natural gradient,” *Trends and Applications in Constructive Approximation, International Series of Numerical Mathematics*, Vol. 151, pp. 259-272.

Jamieson, M., Fazly, A. , Stevenson, S., Dickinson, S., and Wachsmuth, S. (2010). “Using Language to Learn Structured Appearance Models for Image Annotation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 1, pp. 148–164.

Jeon, J., Lavrenko, V., and Manmatha, R. (2003). “Automatic Image Annotation and Retrieval Using Cross-Media Relevance Models,” *Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 28 - August 1, Toronto, Canada, pp. 119-126.

Kang, F., Jin, R., and Sukthankar, R. (2006). “Correlated Label Propagation with Application to Multi-Label Learning,” *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, June 17-22, New York, NY, U.S.A., pp. 1719–1726.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). “ImageNet Classification with

Deep Convolutional Neural Networks," *Proceedings of the Conference on Advances in Neural Information Processing Systems*, December 3-8, Lake Tahoe, Nevada, U.S.A, Vol. 25.

Lavrenko, V. Manmatha, R., and Jeon, J. (2003). "A Model for Learning the Semantics of Pictures," *Proceedings of the Conference on Advances in Neural Information Processing Systems*, December 8-13, British Columbia, Canada, pp. 553-560.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 17-22, New York, NY, U.S.A., pp. 2169-2178.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard,W., and Jackel, L. (1989). "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, Vol. 1, No. 4, pp. 541–551.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations," *Proceedings of the International Conference on Machine Learning*, June 14-18, Montreal, Canada, pp. 609-616.

Li, J. and Wang, J. Z. (2003). "Automatic Linguistic Indexing of Pictures by A Statistical Modeling Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 9, pp. 1075–1088.

Li, J. and Wang, J. Z. (2008). "Real-Time Computerized Annotation of Pictures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 6, pp. 985–1002.

Li, W. and Sun, M. (2006). "Semi-supervised Learning for Image Annotation Based on Conditional Random Fields," *Lecture Notes in Computer Science*, pp. 463-472.

Li, Y. X., Ji, S., Kumar, S., Ye, J., and Zhou, Z. H. (2009). "*Drosophila* Gene Expression Pattern Annotation Through Multi-Instance Multi-Label Learning," *Proceedings of International Joint Conference on Artificial Intelligence*, July 11-17, Pasadena, California, U.S.A., pp. 1445-1450.

Liu, J., Li, M., Liu, Q., Lu, H., and Ma, S. (2009). "Image Annotation via Graph Learning," *Pattern Recognition*, Vol. 42, No. 2, pp. 218-228.

Liu, J., Li, M., Ma, W., Liu, Q., and Lu, H. (2006). "An Adaptive Graph Model for Automatic Image Annotation," *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, October 23-27, Santa Barbara, CA, U.S.A., pp. 61-70.

Lowe, D. G. (1999). “Object Recognition from Local Scale-Invariant Features,” *Proceedings of the IEEE International Conference on Computer Vision*, September 20-25, Kerkyra, Corfu, Greece, pp. 1150-1157.

Luo, J. B., Savakis, A. E. and Singhal, A. (2005). “A Bayesian Network-Based Framework for Semantic Image Understanding,” *Pattern Recognition*, Vol. 38, No. 6, pp. 919-934.

Monay F. and Gatica-Perez, D. (2003). “On Image Auto-Annotation with Latent Space Models,” *Proceedings of the 11th International ACM Conference on Multimedia*, November 02-08, Berkeley, CA, U.S.A. , pp. 275-278.

Monay F. and Gatica-Perez, D. (2004). “PLSA-Based Image Auto-Annotation: Constraining the Latent Space,” *Proceedings of the 12th International ACM Conference on Multimedia*, October 10-16, New York, NY, U.S.A., pp. 348-351.

Oliva, A. and Torralba, A. (2001). “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope,” *International Journal of Computer Vision*, Vol. 42, No. 3, pp. 145-175.

Pati, Y., Rezaiifar, R., and Krishnaprasad, P. (1993). “Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition,” *Proceedings of the Asilomar Conference on Signals, Systems and Computers*,

November 1-3, Pacific Grove, CA , U.S.A., pp. 40–44.

Qi, X. and Han, Y. (2007). “Incorporating Multiple SVMs for Automatic Image Annotation,” *Pattern Recognition*, Vol. 40, No. 2, pp. 728–741.

Riedmiller, M. and Braun, H. (1993). “A Direct Adaptive Method for Faster Backpropagation Learning: the RPROP Algorithm,” *Proceedings of IEEE International Conference on Neural Networks*, March 28 - April 1, San Francisco, CA, U.S.A., pp. 586 – 591.

Serrano, N., Savakis, A. E., Luo, J. B. (2004). “Improved Scene Classification Using Efficient Low-Level Features and Semantic Cues, ” *Pattern Recognition*, Vol. 37, No. 9, pp. 1773-1784.

Shi, J. and Malik, J. (1997). “Normalized Cuts and Image Segmentation,” *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, June 17-19, San Juan, Puerto Rico, pp. 731–737.

Socher, R., Huval, B., Bhat, B., Manning, C. D., and Ng, A. Y. (2012). “Convolutional-Recursive Deep Learning for 3d Object Classification,” *Proceedings of the Conference on Advances in Neural Information Processing Systems*, December 3-8, Lake Tahoe, Nevada, U.S.A,.

Tang J. and Lewis, P. H. (2007). “A Study of Quality Issues for Image

Auto-Annotation with the Corel Dataset," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 17, No. 3, pp. 384-389.

Tang, J. Li, H., Qi, G., and Chua, T. (2010). "Image Annotation by Graph-Based Inference with Integrated Multiple/Single Instance Representations," *IEEE Transactions on. Multimedia*, Vol. 12, No. 2, pp. 131–141.

Tsurugai, Y., Iwasaki, Y., Han, X. H., and Chen, Y. W. (2008). "Region-Based Segmentation and Auto-Annotation for Color Images," *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, August 15-17, Harbin, China, pp. 709–712.

Vogel, J. and Schiele, B. (2007). "Semantic Modeling of Natural Scenes for Content-Based Image Retrieval," *International Journal of Computer Vision*, Vol. 72, No. 2, pp. 0920-5691.

Wang, Y. and Gong, S. (2007). "Refining Image Annotation Using Contextual Relations between Words," *Proceedings of the 6th ACM international conference on Image and video retrieval*, July 09-11, Amsterdam, Netherlands, pp. 425-432.

Wang, Y. Mei, T. Gong, S., and Hua, X. (2009). "Combining Global, Regional and Contextual Features for Automatic Image Annotation," *Pattern Recognition*., Vol. 42, No. 2, pp. 259–266.

Wu, J. and Rehg, J. M. (2011). "CENTRIST: A Visual Descriptor for Scene Categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 8, pp. 1489-1501.

Xu, J. (2011). "An Extended One-Versus-Rest Support Vector Machine for Multi-Label Classification," *Neurocomputing*, Vol. 74, No. 17, pp. 3114–3124.

Xu, X. and Frank, E. (2004). "Logistic Regression and Boosting for Labeled Bags of Instances, " *Lecture Notes in Artificial Intelligence*, Vol. 3056, pp. 272–281.

Yu, Y., Hui, C. L., Choi, T. M., and Au, R. (2010). "Intelligent Fabric Hand Prediction System with Fuzzy Neural Network," *IEEE Transactions on Systems, Man, and Cybernetics - Part C:Applications and Reviews*, Vol. 40, No. 6, pp. 619–629.

Zeiler, M., Krishnan, D., Taylor, G., and Fergus, R. (2010). "Deconvolutional Networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 13-18, San Francisco, U.S.A., pp. 2528 - 2535.

Zha, Z. J., Hua, X. S., Mei, T., Wang, J., Qi, G. J., and Wang, Z. (2008). "Joint Multi-Label Multi-Instance Learning for Image Classification," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, June 23-28, Anchorage, AK, U.S.A., pp.1–8.

Zhang, L. and Ma, J. (2009). "Image Annotation By Incorporating Word Correlation

into Multiclass SVM," *Proceedings of the International Conference on Natural Computation*, August 14-16 , Tianjin, China, pp. 516–520.

Zhang, M. L. and Wang, Z. J. (2009). "MIMLRBF: RBF Neural Networks for Multi-Instance Multi-Label Learning, " *Neurocomputing*, Vol. 72, No. (16–18), pp. 3951–3956.

Zhao, Y., Zhao, Y., and Zhu, Z. (2009). "TSVM-HMM: Transductive SVM Based Hidden Markov Model for Automatic Image Annotation," *Expert Systems with Applications*, Vol. 36, No. 6, pp. 9813-9818.

Zhao, Y. F., Zhao, Y., Zhu, Z. F., and Pan, J. S. (2008). "A Novel Image Annotation Scheme Based on Neural Network," *Proceedings of the International Conference on Intelligent Systems Design and Applications*, November 26-28 , Kaohsiung, Taiwan, Vol. 3, pp. 644–647.

Zhou, Z. H. and Zhang, M. L. (2007). "Multi-Instance Multi-Label Learning with Application to Scene Classification, " *Proceedings of the Conference on Advances in Neural Information Processing Systems*, December 3-8, Vancouver, B.C., Canada, Vol. 19, pp. 1609–1616.

Zhu, Q. X. and Cao, J. D. (2011). "Exponential Stability of Stochastic Neural Networks with Both Markovian Jump Parameters and Mixed Time Delays," *IEEE*

Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 41, No. 2, pp. 341–353.

Zhu, X. L., and Wang, Y. Y. (2011). "Stabilization for Sampled-Data Neural-Network Based Control Systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 41, No. 1, pp. 210–221.