April 2017

Visual Transfer Learning in the Absence of the Source Data

Shuang Ao
The University of Western Ontario

Supervisor
Charles X. Ling
The University of Western Ontario

Graduate Program in Computer Science

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

© Shuang Ao 2017

Recommended Citation

Ao, Shuang, "Visual Transfer Learning in the Absence of the Source Data" (2017). *Electronic Thesis and Dissertation Repository*. 4463.
https://ir.lib.uwo.ca/etd/4463

Abstract

Image recognition has become one of the most popular topics in machine learning. With the development of Deep Convolutional Neural Networks (CNN) and the help of the large scale labeled image database such as ImageNet, modern image recognition models can achieve competitive performance compared to human annotation in some general image recognition tasks. Many IT companies have adopted it to improve their visual related tasks. However, training these large scale deep neural networks requires thousands or even millions of labeled images, which is an obstacle when applying it to a specific visual task with limited training data. Visual transfer learning is proposed to solve this problem. Visual transfer learning aims at transferring the knowledge from a source visual task to a target visual task. Typically, the target task is related to the source task, and the training data in the target task is relatively small. In visual transfer learning, the majority of existing methods assume that the source data is freely available and use the source data to measure the discrepancy between the source and target task to help the transfer process. However, in many real applications, source data are often a subject of legal, technical and contractual constraints between data owners and data customers. Beyond privacy and disclosure obligations, customers are often reluctant to share their data. When operating customer care, collected data may include information on recent technical problems which is a highly sensitive topic that companies are not willing to share. This scenario is often called Hypothesis Transfer Learning (**HTL**) where the source data is absent. Therefore, these previous methods cannot be applied to many real visual transfer learning problems.

In this thesis, we investigate the visual transfer learning problem under HTL setting. Instead of using the source data to measure the discrepancy, we use the source model as the proxy to transfer the knowledge from the source task to the target task. Compared to the source data, the well-trained source model is usually freely accessible in many tasks and contains equivalent source knowledge as well. Specifically, in this thesis, we investigate the visual transfer learning in two scenarios: domain adaptation and learning new categories. In contrast to the previous methods in HTL, our methods can both leverage knowledge from more types of source models and achieve better transfer performance.

In chapter 3, we investigate the visual domain adaptation problem under the setting of Hypothesis Transfer Learning. We propose Effective Multiclass Transfer Learning (**EMTLe**) that can effectively transfer the knowledge when the size of the target set is small. Specifically, EMTLe can effectively transfer the knowledge using the outputs of the source models as the auxiliary bias to adjust the prediction in the target task. Experiment results show that EMTLe can outperform other baselines under the setting of HTL.

In chapter 4, we investigate the semi-supervised domain adaptation scenario under the setting of HTL and propose our framework Generalized Distillation Semi-supervised Domain Adaptation (**GDSDA**). Specifically, we show that GDSDA can effectively transfer the knowledge using the unlabeled data. We also demonstrate that the imitation parameter, the hyperparameter in GDSDA that balances the knowledge from source and target task, is important to the transfer performance. Then we propose GDSDA-SVM which uses SVMs as the base classifier in GDSDA. We show that GDSDA-SVM can determine the imitation parameter in GDSDA autonomously. Compared to previous methods, whose imitation parameter can only be determined by either brute force search or background knowledge, GDSDA-SVM is more effective in real applications.

In chapter 5, we investigate the problem of fine-tuning the deep CNN to learn new food categories using the large ImageNet database as our source. Without accessing to the source data, i.e. the ImageNet dataset, we show that by fine-tuning the parameters of the source model with our target food dataset, we can achieve better performance compared to those previous methods.

To conclude, the main contribution of is that we investigate the visual transfer learning problem under the HTL setting. We propose several methods to transfer the knowledge from the source task in supervised and semi-supervised learning scenarios. Extensive experiments results show that without accessing to any source data, our methods can outperform previous work.

Keywords: Visual Transfer Learning, Hypothesis Transfer Learning, Supervised Learning, Semi-supervised Learning

Acknowlegements

Firstly, I would like to thank my advisor Prof Charles X. Ling for his patience and insightful guidance. He was always eager to discuss new ideas at any time help me understand the research and inspired me to thinking critically for the topic I was working on.

Thanks are also given to my colleagues: Yan Luo, Xiao Li, Bin Gu, Chang Liu, Robin Liu and Jun Wang for their collaboration and valuable discussions. Many thanks to Xiang Li. We worked together with many ideas and he is very helpful in many details of the papers we published together.

The Last gratitude is given to my families: my parents Dingan Ao and Liwen Yang, and my wife Jinglang Hu. Without their encouragements and supports, I am not able to pursuing my PhD degree in Western and finish this dissertation.

My research is supported by NSERC Grants and scholarship from the Universitys the School of Graduate Studies. This thesis would not have been possible without the generous resources provided by the Department of Computer Science.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the explosive image resources people uploaded every day, image recognition becomes a very hot topic and has drawn many attentions in recent years. Every year, there are many inspiring results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). With the development of recognition technology, many IT companies want to use the image recognition techniques to serve their customers and many interesting applications have been developed, such as HowOld from Microsoft and Im2Calories from Google.

To successfully capture the diversity of different objects around us, many recognition models contain thousands or sometimes even millions of parameters and require a large amount of training images to tune these parameters as well. As the visual recognition system become increasingly successful in many general recognition tasks, people expect that it can solve the recognition problems in many new and complicated areas which are paid less attention to before. Unfortunately, for some real applications, it is often difficult and cost to collect a large set of training images. Moreover, most algorithms require that the training examples should be aligned with a prototype, which is commonly done by hand. In many real applications, collecting and fully annotating these images can be extremely expensive and could have a significant impact on the over cost of the whole system. Therefore, the biggest challenge of applying the current recognition algorithms to the new task is lack of training data.

On the other hand, object recognition is one of the most important parts of our human visual system. We can recognize various kinds of materials (apple, orange, grape), objects (vehicles, buildings) and natural scenes (forests, mountain). At the age of six, human can recognize about 10^4 object categories[12]. Our human can learn and recognize a new object with just a glance, which means we can capture the diversity of forms and appearances of an object with just a handful examples. This remarkable ability is obtained by effectively leveraging the learned knowledge and applying it to the new tasks. It could be ideal if there is a visual recognition model can have such ability, which is referred to as **Visual Transfer Learning** (**VTL**).

1

VTL has been increasing popular with the success of modern image recognition algorithms. In VTL, the source domain is referred to the one we have already learned and the target domain is the one we want to learn. In VTL, measuring the relatedness of the source and target tasks is important for the transfer process. In previous studies of VTL, people assume that the source data are always available and can be freely accessed. Therefore, the relatedness of the source and target domain can be effectively measured by comparing the data in two domains. However, this assumption rarely holds in real applications. Source data are often a subject of legal, technical and contractual constraints between data owners and data customers. Beyond privacy and disclosure obligations, customers are often reluctant to share their data. On the other hand, sharing the model trained from the source data, i.e. the source model, instead of the data can avoid these obligations and is more common in real applications. VTL under this situation is usually called **Hypothesis Transfer Learning** (**HTL**) [60] and the source model is called source hypothesis.

In this thesis, we investigate the VTL problem under the HTL setting. Specifically, we investigate the VTL problem under two scenarios: domain adaptation (transductive transfer) and learning the new categories (inductive transfer). The methods proposed in this thesis focus on how to leverage the knowledge from the source model and transfer it to the target task effectively. In this section, we first give an overview for image recognition. Then we illustrate the current approach for VTL tasks and their limitation for the real applications. Finally, we briefly demonstrate the contributions of this thesis.

1.1 Overview for Image Recognition

In this section, we review the major procedures for image recognition. A general image recognition method consists of three parts: image preprocess, feature extraction and classification.

1.1.1 Preprocess

Firstly, the optical property of an object is captured through its optical sensor of a digital camera and then the digital camera generates raw digital data of the image. After receiving the raw data of a image from the sensor, preprocess is to generate a new image from the source image. This new image is similar to the source image, but differs from it considering certain apsects, e.g. the new image has smoother edge, better contrast and less noise. Here, some *pixel operations* and *local operations* are used to improve the contrast and remove the noise.

Another important operation of preprecess is segmentation according to the object, i.e. finding the region of interest. Images used for recognition should be aligned, making the target
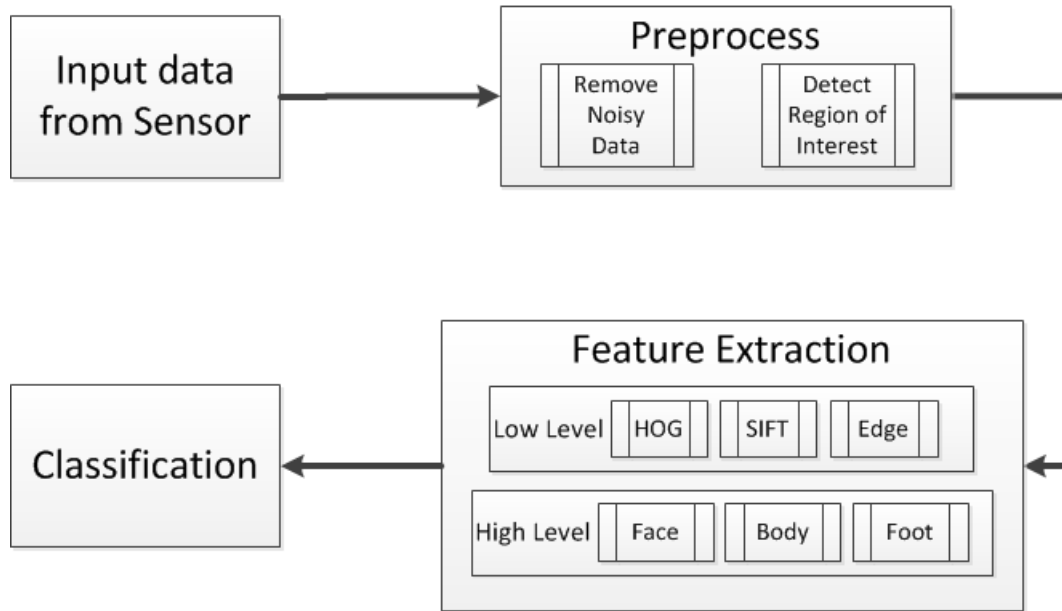
Figure 1.1: Major procedure for image recognition.

object appear in the central of the image and remove those irrelevant area.

The result of preprocess has great impact on the final result of the recognition. Clear and noise free images can make the feature extraction more effective and significantly improve the final classification accuracy.

1.1.2 Feature Extraction

Feature extraction is used to extract the optical properties of an image and represent interesting parts of an image from the raw image data as a compact feature vector. The feature vector is then used for either training the classifier or recognition. Therefore, feature extraction is the most important part for image recognition. The quality of the features extracted from a image have great impact on the recognition result. There are two major streams for feature extraction: the hand engineered method and representation learning method.

1.1.2.1 Hand Engineered Feature

Hand engineered features are typically low level and local features. Low level features are extracted according to some optical properties of an image. These features are low level / local features. There is a widely agreement that local features are an efficient tool for object representation due to their robustness with respect to occlusion and geometrical transformations [105]. Common low level hand engineered features include Histogram of Oriented Gradients (HOG)

[26], Scale Invariant Feature Transform (SIFT) [70], Speeded Up Robust Features (SURF) [8], Local Binary Patterns (LBP) [78], and color histograms [13]. Feature descriptors obtain from these low level features refer to a pattern or distinct structure found in an image, such as a point, the edges, or some small image patches. They are usually associated with an image patch that differs from its immediate surroundings by texture, color, or intensity. What the feature actually represents does not matter. We know that it is distinct from its surroundings. These low level
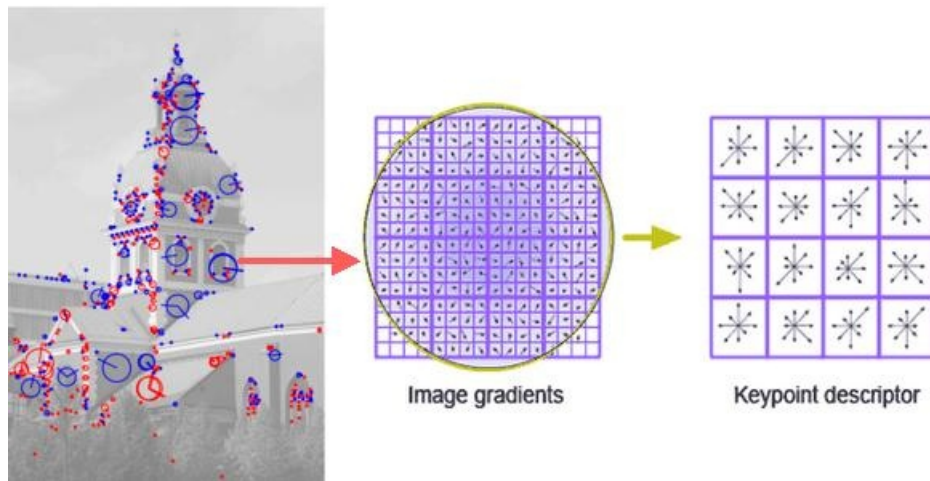
Figure 1.2: Feature extraction using SIFT.

features can be used directly for recognition. However, since they just represent certain local properties of an image and are not discriminative enough for recognition, discriminative high level features can be further learned by combining the low level features using some algorithms such as bag-of-visual words[62].

1.1.2.2 Representation Learning

Representation learning is mainly described by Deep Learning algorithms[58] or Auto Encoders [11]. The ideas is to learn a group of filters that are able to capture various kinds of features to discern one category of images from the another category with some supervised or unsupervised algorithm. Typically in representation learning, features are learned hierarchically from low-level features to high level ones automatically. Learning representation from an image can start from either low level hand-crafted features (for Auto Encoders) or raw pixels of an image (for Deep Learning).

Auto Encoders are widely used to combine different types of low level feature. The outputs of the Auto Encoders are some latent representations. These latent representations are learned from the given images that have lowest possible reconstruction error. Even though the high
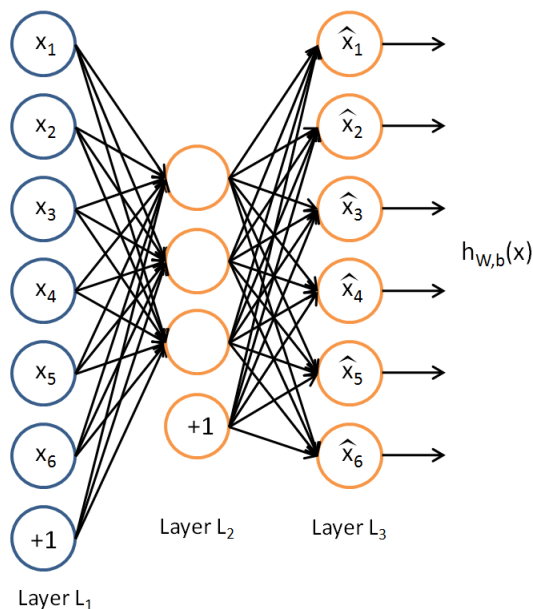
Figure 1.3: General Scheme of Auto Encoders. L1 is the input layer, possibly raw-pixel intensities. L2 is the compressed learned latent representation and L3 is the reconstruction of the given L1 layer from L2 layer. AutoEncoders tries to minimize the difference between L1 and L3 layers with some sparsity constraint.

level representations from Auto Encoders are learned by minimizing the reconstruction errors, they are still not robust enough to handle all kinds of variance of the objects in some tasks.

Deep Learning is the most popular approach for learning representations. It has been widely used for all kinds of image recognition tasks and achieved the state-of-the-art performance on some large scale image recognition tasks, such as ILSVRC and The PASCAL Visual Object Classes Challenge (PASCAL VOC). Convolutional Neural Networks (CNN) is the most popular deep learning model for the image recognition tasks[1]. The first deep CNN that had great success on image recognition is the LeNet proposed by Y.LeCun in 1989 [64]. Backpropagation was applied to Convolutional Neural networks with adaptive connections. This combination, incorporating with Max-Pooling and speeding up on graphics cards has become an important part for many modern, competition-winning, feedforward, visual Deep Learners. Deep CNNs have been widely used as the feature extractor for all kinds of images recognition tasks and proven to be the most powerful method of feature extraction.

---

[1]Deep CNNs are sometimes considered as the end-to-end classifier while learning the feature representation and discriminative classifier simultaneously. However, the feature representation learned from deep CNNs can still achieve good results with other classifiers and here we consider it as a feature extractor rather than a classifier.
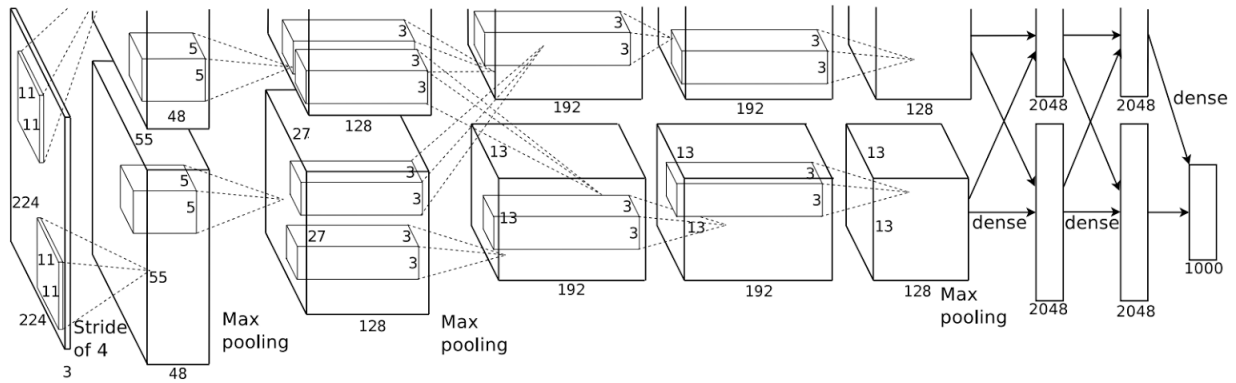
Figure 1.4: The architecture of ALEXNET (adopted from [58]).

1.1.3   Classification

After extracting feature representation from the images, a classifier is used to train a recognition model as well as for predicting the new coming images. A supervised model is always used for training the recognition model. Discriminative Classifier such as Support Vector Machine (SVM) is widely used as the classifier for recognition [24]. As we mentioned before, in order to capture different variances of the images for one category, the size of the feature representation for an image is usually very large. In order to avoid overfitting, the size of the training set should be at least the same size of the feature representation as well. Some classifiers such as Bayesian method or decision tree require to consider the correlations between each feature and the class labels and suffer from the large feature dimension. However, Discriminative Models[15] are more convenient for training. Discriminative Models can be effectively optimized with stochastic gradient descent and are suitable for the large training set.

However, to capture all variance of an object, training a good recognition model requires abundant data when we learn a model from scratch. With the limited training data, it is difficult to achieve a good classification performance. Transfer learning is an effective way to solve this problem by utilizing the knowledge from previous tasks. In this thesis, we focus on how to transfer the knowledge from the source domain for recognition tasks. The methods proposed in chapter 3 and chapter 4 mainly focus on the stage of classification while the method in chapter 5 focuses on both feature representation learning and classification.
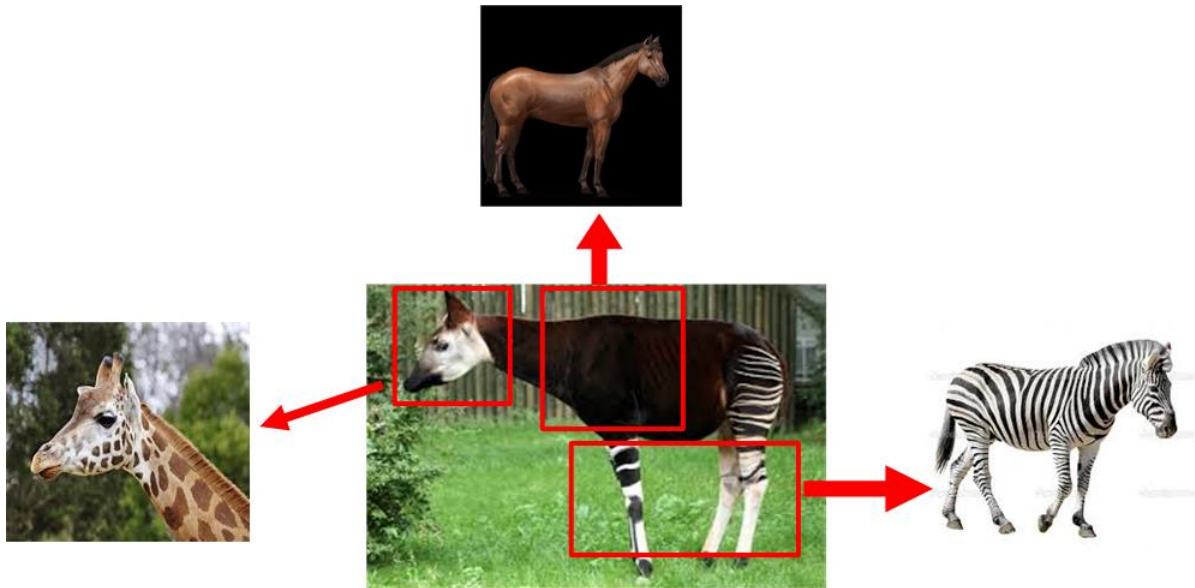
Figure 1.5: An intuitive description for human to learn new concept: an okapi can be roughly described as the combination of a body of a horse, legs of the zebra and a head of giraffe.

1.2 Approaches in Visual Transfer Learning and the Limitations

Previous work of visual transfer learning focuses on designing classifiers that can leverage the source knowledge effectively. In this section, we briefly review methods for the visual transfer learning and show the limitation of previous work.

1.2.1 Intuition for Visual Transfer Learning

The intuition of visual transfer learning comes from human recognition mechanism. For our human, all the information acquired is stored in our memory. This information is organized according to the properties. When we see a new concept, we don't treat it isolated but connect it to certain previous knowledge we stored in our memory. By comparing a new concept with the organized information in our memory, we can capture the property of a new concept effectively. When referring to visual tasks, several examples can be given to show this cognitive ability. For instance, when we describe the animal "okapi" (see Figure 1.5), we would probably say that: okapi has a body of a horse, legs of the zebra and the head of giraffe. People who never see a zebra could instantly have a rough idea of a zebra.

This indicates that to learn a concept effectively, we should be able to make use of the gained knowledge instead of learning it from scratch. This process is commonly referred to

as transfer learning[79]. Traditional machine learning methods work under the common assumption: training data and testing data are drawn from the same feature space and same distribution. In transfer learning, the test data can come from a different distribution. The data from the original distribution is called source data, and data from the new distribution is called target data. Transfer learning is used to utilize the source knowledge from the source data to help to build the new model to classify the target data.

1.2.2   Approaches for Visual Transfer Learning

Successfully leveraging the source knowledge can greatly improve the performance of the target model. In general, the more related the source and target domain are, the more useful the source knowledge is and the more benefit the target model can get. Leveraging unrelated knowledge cannot help to improve the performance of the target model or even hurt it. Therefore, the key issue for visual transfer learning is to identify the relatedness of the source and target domain. The major approach for Visual Transfer Learning consists of two main directions: Distribution Similarity Measurement and Instance Reuse.

- **Distribution Similarity Measurement**. The core idea of transfer learning is to leverage the related source knowledge. The more related the source is, the better transfer performance we can achieve. Thus, measuring the relatedness of the source knowledge is an important part in transfer learning especially where there are multiple sources. A straight-forward approach to identify the relatedness of the source and target domain is to measure their similarity directly. Measuring the data discrepancy through some statistical measurements such as **Maximum Mean Discrepancy** (MMD) [31], has been a popular way to identify the source and target domain. MMD reflects the distance of two data distributions in the Reproducing Kernel Hilbert Space (RKHS) [3].

- **Instance Reuse**[67]. We apply transfer learning under the scenario where the target data is scarce and we are not able to build the target model alone with the target data. A simple solution is to "borrow" some of the data from the source domain and use it to build the target model together with the target data. This approach can directly increase the size of the data in the target domain and effectively improve the performance of the target model. For example, **Feature Transformation** [32] can overcome the data distribution mismatch in different domains and project the data into the same augmented space and thus can increase the training data for the target task as well (see Figure 1.6).
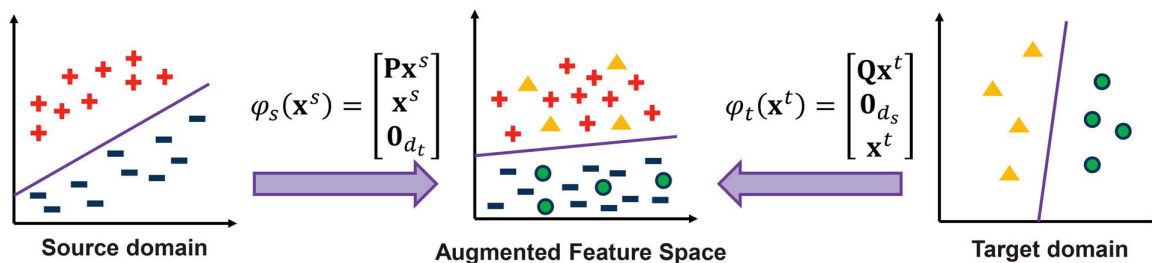
Figure 1.6: Feature transformation. Transform the data in different domains into a augmented feature space.

1.2.3 Limitation of Previous Methods

From the review of the approaches for Visual Transfer Learning we can see that most previous methods require access to the source data to obtain the source knowledge. However, in many practical problems, these previous approaches may not be as convenient as we thought due to the following reasons:

- **Data accessibility**. The source data may not be able to access for some tasks. For example, the clinical database is not allowed to access for general publics due to the privacy. Disclosure obligations and will to share the databases are also two important reasons that make the source data inaccessible.

- **Size of the source data**. Besides data accessibility, many previous methods [27][32] require accessing to each of the individual source instance to obtain the source knowledge which is ineffective for many large source domain. For example, it is almost impossible to measure the MMD for some large source domain which contains hundreds of thousands of instances.

From above we can see that these previous methods can successfully leverage the source knowledge under the assumption that the source data are freely accessible and relatively small. However, this assumption could fail in real applications. In some cases, the source data could be private (such as the clinic data from patients) and therefore, could not be shared with the public. Moreover, those large source dataset contains more knowledge and information compared to the small ones and thus can better improve the transfer performance of the target task. However, obtaining the similarity measurement of these large datasets with those previous methods can be tedious and inefficient. It is important to find a way to leverage the source knowledge without accessing to the source data.

In this thesis, we assume that we can freely access to the source model trained from the source data and thus leverage the knowledge from the model instead. Using the source model

for transfer learning can successfully avoid the two issues discussed above. Source model can contain as much knowledge as the source data while without containing any information regarding the individual instance. Therefore, the owner of the source data does not have to worry about the data privacy issue. For those large source dataset such as ILSVRC containing millions of images, a trained source model is normally a few hundreds megabytes and public available. Therefore, leveraging the source knowledge from source model instead of the source data is more practical for real visual transfer learning applications.

1.3 Main Contribution

In this section, we discuss the challenges that we may encounter when we can only access the source model. Then we demonstrate the learning scenarios and proposed solutions in this thesis.

1.3.1 Challenges

Despite for the general challenges in transfer learning, in our HTL settings, there are some new challenges in this thesis.

The first challenge is **knowledge representation**, i.e. how to obtain the source knowledge when we are not able to access to the source data. When source data is accessible, the source knowledge is relatively explicit and we can easily obtain the source knowledge by either analyzing the source data distribution or make use of the source data to help training the target mode. The knowledge of the source domain is implicit and encoded in the source model using certain learning algorithms. How to effectively extract the source knowledge from the models is challenging.

The second challenge is **knowledge expressiveness**, i.e. how to leverage the source knowledge to help training the target model. As the source knowledge is implicit, how to effectively leverage the source knowledge and improve the transfer performance is also important. We also expect that the source knowledge extracted from the source model should be as general as possible so that the source knowledge can be extracted from different types of source model. Therefore, our transfer learning algorithm can work in many situations.

The last challenge is **knowledge regularization** i.e. how to guarantee the performance of our transfer method. Humans appear to have mechanisms for deciding when to transfer information, selecting appropriate sources of knowledge, and determining the appropriate level of abstraction [100]. A basic criterion for the knowledge transfer process is that leveraging the knowledge from the source model should not hurt the performance of the target model.

Negative transfer [79] happens when the performance of target model degrades after receiving the knowledge from the source domain and how to avoid negative transfer is still an open question to all transfer learning researchers. The absence of the source data makes the situation more complicated.

1.3.2 Two Transfer Learning Scenarios

In this thesis, we mainly focus on two transfer learning scenarios: inductive transfer learning for new classes and domain adaptation. The definition of the two scenarios is as follows:

Definition Domain Adaptation[9] Let X be the input space and Y be output space. Given the source domain D_s and the target learning task D_t with marginal distribution $P(X_s)$ and $P(X_t)$, we assume that D_s and D_t share the same conditional distribution $P(Y|X)$. The goal of Domain Adaptation is to learn the $P(X_t|Y_t)$ for the target task with the help of D_s.

Definition Inductive Transfer Learning[79] Given a source domain D_s and the source learning task T_s, a target domian D_t and the target learning task D_t, inductive transfer learning aims to help improve the learning of the target task function $f_t(\cdot)$ in D_t using the knowledge from D_s where $D_s \neq D_t$.

The major difference between the two transfer learning scenarios is that in inductive transfer learning, the source and target tasks are two different but related tasks (e.g. from sport car to heavy truck) while in domain adaptation, the source and target tasks are the same task but with different marginal distributions (e.g. from the animation dog to the real dog).

(a) Domain Adaptation   (b) Inductive transfer learning the new class
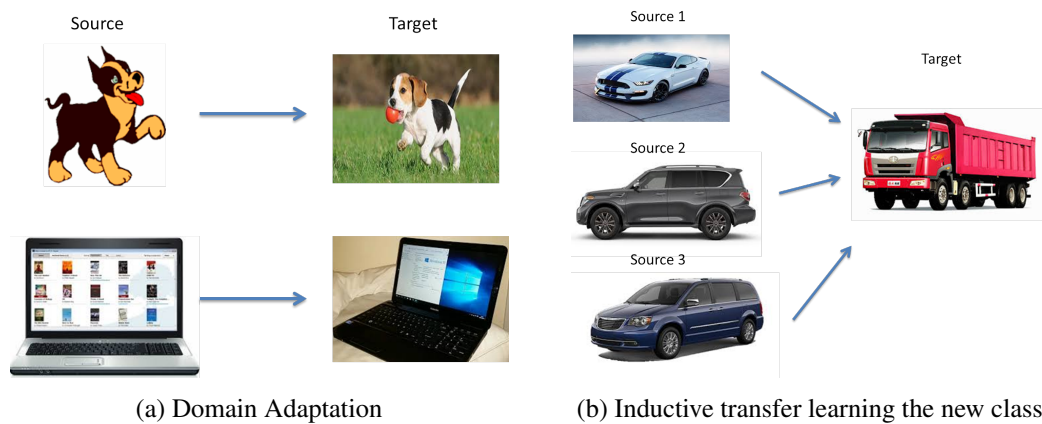
Figure 1.7: Difference between two transfer learning scenarios

1.3.3   Proposed Methods

There are three methods proposed in this thesis, one in inductive transfer learning for new classes and two methods in domain adaptation.

In chapter 3, we extend the previous methods in HTL which are limited to using SVMs as the source model and propose a novel method **Effective Multi-class Transfer Learning** (EMTLe) for supervised domain adaptation. We use the output of the source model as the auxiliary bias to adjust the target model. Here, the output of the source model is used as the prior to adjust the decision for the target model. As long as we can set a proper weight to the prior knowledge, the source model can serve well for the target task. Because we only require the source model to provide its decision, we can treat it as a black-box model and EMTLe can leverage the source model from any source classifier that can generate the class probability of an example.

In chapter 4, we investigate the problem of semi-supervised domain adaptation where most of the data in the target task are unlabeled and propose a novel framework **Generalized Distillation Semi-supervised Domain Adaptation** (GDSDA). In GDSDA, the source model generates "soft labels" for the target data and can improve the performance together with the true labels from target task. We use the imitation parameter to determine the relative importance of the soft label and true label. Then we propose GDSDA-SVM that can determine the imitation parameter autonomously through cross-validation.

In chapter 5, we use the deep neural network to solve the transfer learning problem for food recognition. We use GoogLeNet trained from ImageNet of 1000 classes as our source model and two food databases as our target task, containing 101 classes and 265 classes respectively. In this chapter, we don't treat the source model as a black-box. Instead, we can obtain the parameters of the original GoogLeNet. We re-use the parameters in the original GoogLeNet as the prior and fine-tune it on our target task to achieve the improved performance. By re-using and fine-tuning the parameter, i.e. features from source tasks, we can effectively learn new categories of the target task. We show that without accessing the source data, we can still achieve better performance compared to previous methods.

1.4   Summary

In this chapter, we briefly introduced the problems that are solved in this thesis. We first demonstrate the procedure for image recognition and introduce some previous work for visual transfer learning. Then we pointed out the limitations of the previous work and briefly introduced our methods.

Chapter 2

Related Work

In this chapter, we review some previous work related to ours. We first review the classifiers used in this thesis, providing the general concept and principle of how they work. Then we review the types of transfer learning for visual recognition from two different views and then discuss the previous work of how to alleviate negative transfer. Finally, we review some methods related to the three methods used in this thesis.

2.1 Classifiers for Image Recognition

In our scenario, we have to face the problem of image recognition. Due to the large dimension of the feature representation for each image as well as the size of training image, manual classification is hopeless. As we mentioned in Section 1.1, a recognition model is used to distinguish the objects from different categories automatically trained by supervised learning. In this section, we introduce the classifiers we used in this thesis.

2.1.1 Binary Classification and Multi-class Classification

In image recognition, we train a recognition model from a set of training images along with their labels provided. The labels are predefined in a category space. Thus, the task of image recognition is to classify each image as one predefined category. If there are only two categories, this recognition task is called binary classification. For the task recognizing the objects from more than two categories, the recognition task is called multi-class classification [4] [58].

Here, we give a formal definition of the scenario for binary and multi-class classification. Generally, we can decompose the multi-class learning task into a set of binary scenarios by training a binary classifier for each class, e.g. One-VS-Rest strategy (see figure 2.1)[82] [102].
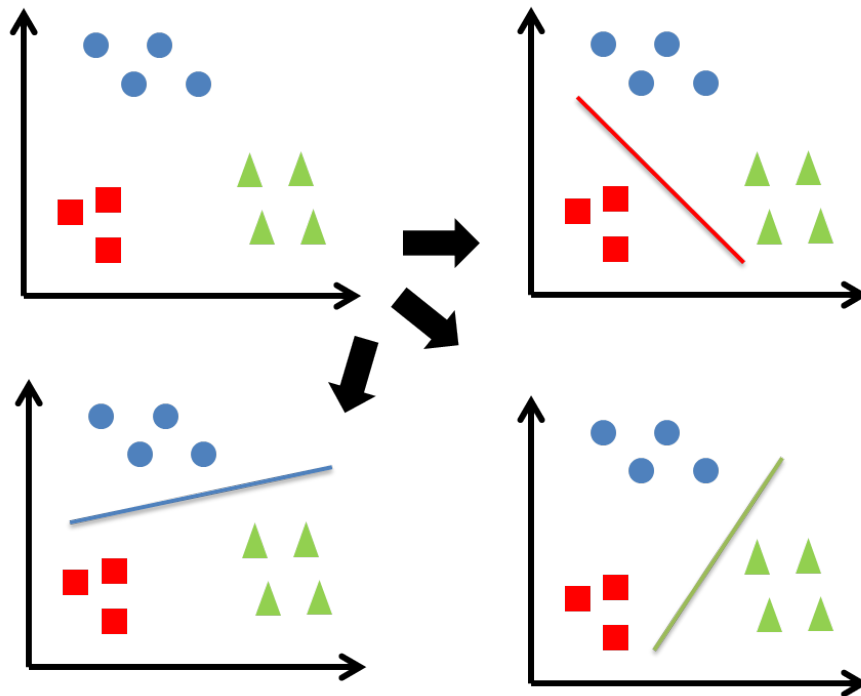
Figure 2.1: One-vs-Rest strategy for multi-class scenario. A three classes problem can be decomposed into 3 binary classification sub-problems.

The binary scenario for classification can be defined as follow: given a dataset from domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the input feature representation and $\mathcal{Y}$ is the binary label set $\{1, -1\}$ (for some classifier, $\{1, 0\}$ is also used). We usually use the label 1 to denote the examples belong to one certain category and -1 to denote examples not belong to that category. We assume that the training image set $D_{train} = \{(x_i, y_i)\} \subset \mathcal{X} \times \mathcal{Y}$ and the test image set $D_{test} = \{(x_i^t, y_i^t)\} \subset \mathcal{X} \times \mathcal{Y}$ are given and separated from each other. Each pair (x_i, y_i) denotes the input feature representation x_i and its corresponding label y_i for the *ith* image in the both set. Our goal of the classification problem is to learn a decision function $f : \mathcal{X} \to \mathcal{Y}$ from the training set D_{train} such that f can achieve good performance on both D_{train} and D_{test}.

2.1.2 Softmax Classifier

In this subsection, we will introduce a widely used linear classifier **Softmax classifier** in image recognition. Linear classifier is commonly used as the classification model for image recognition. Linear classifier achieves this by making a classification decision based on the value of a linear combination of the input feature representations of a image. A linear classifier consists of two parts: a score function and a loss function[107]. The score function maps the input data

into the class scores and the loss function that quantifies the agreement between the predicted scores and the ground truth labels. Linear classifier often works very well when the number of dimensions of the input is large. Therefore, it is widely used as the classifier for image recognition, especially as the classifier for Convolutional Nueral Networks [64].

Typically, Softmax classifier is widely used for multi-class image classification. Softmax classifier (also called multinomial logistic regression) is a generational form of logistic regression for the multi-class scenario. As logistic regression can only handle the binary classification scenario, Softmax classifier adapts the one-vs-rest strategy where several logistic regression models are trained for each class.

Given a training set $\{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, we assume the label $y_i \in \{1, 0\}$ and the input feature $x_i \in \mathcal{R}^n$. For each binary logistic regression model, The score function takes the form:

$$f_w(x) = \frac{1}{1 + \exp(-w^T x)} \tag{2.1}$$

and the parameters w are optimized to minimize the following loss function:

$$l(w) = -[\sum_{i=1}^{m} y_i \log f_w(x_i) + (1 - y_i) \log(1 - f_w(x_i))] \tag{2.2}$$

For Softmax classifier, it is used to handle the multi-class classification problem and suppose there are N classes. Therefore, y can take from N different values $\{1, 2, 3, ..., N\}$ instead of just two. For a given test example x, the score function estimate the probability $P(y = n|x)$ for each value of $n = 1, 2, 3, ..., N$, i.e. estimate the probability that each score assigns the input x to the N classes associated to the N different possible values. Thus, the score function will output a N-dimensional vector providing N estimated probabilities whose sum of elements is 1. The N dimensional output of the score function can be generated according to the following form:

$$f_w(x) = \begin{bmatrix} P(y = 1|x; w) \\ P(y = 2|x; w) \\ ... \\ P(y = n|x; w) \end{bmatrix} = \frac{1}{\sum_{j=1}^{N} \exp(w^{(j)T} x)} \begin{bmatrix} \exp(w^{(1)T} x) \\ \exp(w^{(2)T} x) \\ ... \\ \exp(w^{(N)T} x) \end{bmatrix} \tag{2.3}$$

Here $w^{(1)T}, w^{(2)T}, ..., w^{(N)T}$ are the parameters of the Softmax classifier model. The term $\frac{1}{\sum_{j=1}^{N} \exp(w^{(j)T} x)}$ is called the normalization term so that the N distributions sum to 1.

To optimize the parameters $w^{(1)T}, w^{(2)T}, ..., w^{(N)T}$, the cross-entropy loss used for the Softmax classifier is defined as:

$$J(w) = -\left[\sum_{i=1}^{l} \sum_{k=1}^{N} \ell\{y_i = k\} \log \frac{\exp(w^{(k)T} x_i)}{\sum_{j=1}^{N} \exp(w^{(j)T} x_i)} \right] \tag{2.4}$$

Here ℓx is the 0-1 loss function:

$$\ell\{x\} = \begin{cases} 1 & x \text{ is true} \\ 0 & x \text{ is false} \end{cases} \tag{2.5}$$

It is noted that Eq (2.4) is a generalized form of Eq (2.2). Minimizing Eq (2.4) can be interpreted as minimizing the negative log likelihood of the correct class, which is equivalent to performing Maximum Likelihood Estimation (MLE) [53].

The minimum of $J(w)$ can be obtained by gradient descent method while taking the gradient:

$$\nabla J(w^{(n)}) = -\sum_{i=1}^{l} \left[x_i \left(\ell\{y_i = n\} - \frac{\exp\left(w^{(k)T} x_i\right)}{\sum_{j=1}^{N} \exp\left(w^{(j)T} x_i\right)} \right) \right] \tag{2.6}$$

2.1.3   Support Vector Machines

In this subsection, we will review another widely used discriminant classifier, **Support Vector Machine** (SVM) [24]. SVM is another classifier that has been adoptive in many image recognition tasks [22] [89] [110]. In this thesis, we also use a classifier based on SVM. We will give a detailed description of SVM.

As we mentioned before, the linear classifier consists of two parts: the score function and loss function. SVM classifier can be divided into two categories based on their score function: linear SVM that uses a linear discriminant function and kernel SVM that uses kernel function. Kernel SVM can be considered as a extension version of linear SVM where kernels are used for calculate the scores of the inputs. Another difference for between linear and kernel SVM is linear SVM can be solved on the primal problem while kernel SVM is mostly optimized on its dual [24] [90].

First, we will introduce the linear SVM. Linear SVM uses the simplest representation of a score function by taking the linear combination of the input vector:

$$f(x) = w^T x + b \tag{2.7}$$

where w is called the weight vector and b is called the bias. In a binary scenario, for the input example x and the class labels $y \in \{c1, c2\}$, x is assigned to class $c1$ if $f(x) \geq 0$ and $c2$ otherwise. Therefore, the corresponding decision surface is defined by $f(x) = 0$. For two points x_1 and x_2 lie on the decision surface, we have $f(x_1) = f(x_2) = 0$. Then we can have $w^T(x_1 - x_2) = 0$ and hence the weight vector w is orthogonal to every point lying within the decision surface, i.e w determines the orientation of the decision surface. Similarly, if x lies on the decision surface, the normal distance from the origin to the decision surface is given by:

$$\frac{w^T x}{\|w\|} = -\frac{b}{\|w\|} \tag{2.8}$$

(a) Different sparating hyperplanes   (b) Max-Margin hyperplanes
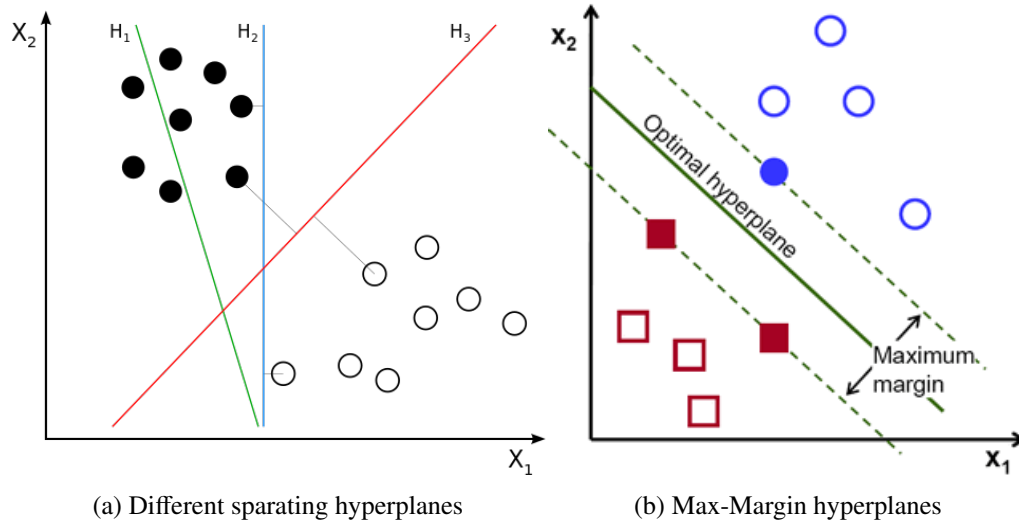
Figure 2.2: Support Vector Machine

Therefore, we can see that, the location of the decision surface is determined by the bias b.

2.1.3.1 Hard Margin SVM

Hard margin SVM is used to find the optimal solution for the data sets that are linear separable. Given a set of n training points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where $y_i \in \{1, -1\}$, we expect to find a decision surface that can separate the data from two classes. However, when the data from these two classes can be linearly separated, there could be several decision surfaces that can separate the data. The idea of SVM is to choose the maximal margin decision surface so that the distance between these two classes is as large as possible (called maximal margin hyperplane). For example, in figure 2.2(a), $H2$ and $H3$ are two candidate decision surface that can separate the data. However, $H3$ is has the largest distance to all the data from two classes and SVM will choose $H3$ as the optimal hyperplane.

The optimal hyperplane can be found by minimizing the following objective function:

$$\begin{aligned} \min \quad & \|w\|^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 \quad \text{for all } 1 \leq i \leq n \end{aligned} \tag{2.9}$$

2.1.3.2 Soft Margin SVM

In real world application, most data are not linear separable. Therefore, SVM introduce the concept of slack variable to handle this situation. Slack variable is defined as:

$$\xi_i = hinge(x_i) = \max(0, 1 - y_i(w^T x_i + b)) \tag{2.10}$$
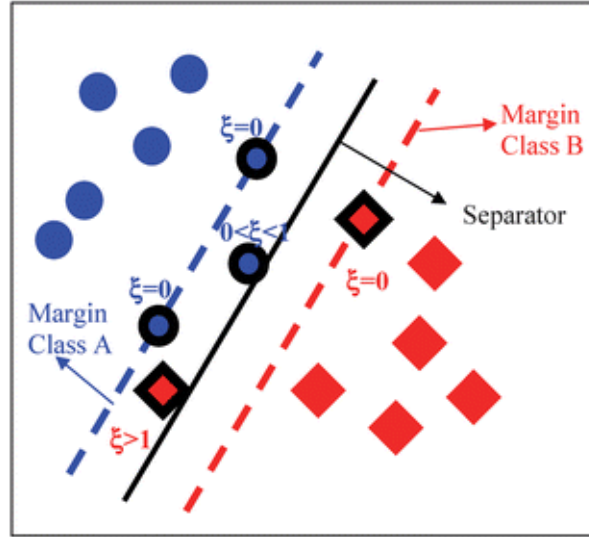
Figure 2.3: Slack variables for soft-margin SVM

The value of slack variable is 0 if the example x_i lies on the correct side of the margin. For those data on the wrong side, its value is proportional to the distance from the correct margin.

Therefore, to find the parameters of hyperplane, i.e. weight vector w and bias b, soft-margin SVM minimize the following loss function:

$$
\begin{aligned}
\min \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_i^n \xi_i \\
\text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\
& \xi_i \geq 0 \quad \text{for all } 1 \leq i \leq n
\end{aligned}
\tag{2.11}
$$

The objective function is called primal of SVM. A stochastic sub-gradient descent can be used to find the optimal solution for eq. (2.11) effectively [90].

2.1.3.3 Kernel SVM

Linear soft-margin SVM works well when number of features is larger than number of training examples. However, when the size of the training example is larger than the features, kernel SVM (such as Gaussian Kernel) with proper parameters outperforms linear SVM [56].

The idea of kernel was first introduced into pattern recognition by Aizerman *et. al.* [2]. When the size of the training examples are significantly larger than the dimension of the input features and the distribution become more complex, these data can not be easily separated by a straight line in the feature space. Instead of obtaining the optimal hyperspace in the input feature space, kernel SVM tries to map the inputs into a high-dimensional feature spaces and
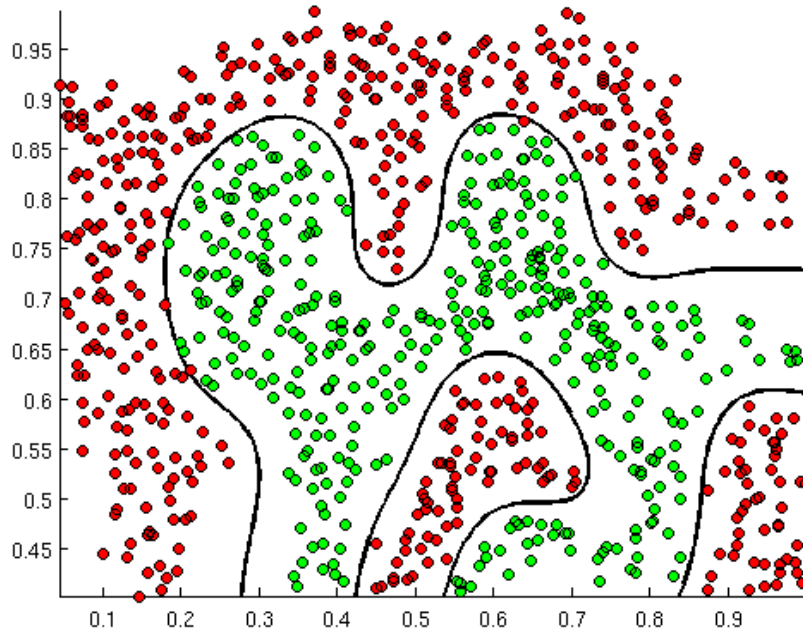
Figure 2.4: The hyperplane of SVM with RBF kernel for non-linear separable data.

find the optimal hyperplane in the high-dimensional feature spaces. Given a feature space mapping $\phi(x)$, the score function for kernel SVM can be re-written as:

$$f(x) = w^T \phi(x) + b \tag{2.12}$$

From eq (2.12) we can see that, when we take the identity mapping $\phi(x) = x$, the kernel SVM becomes a linear SVM. Therefore, the linear SVM can be considered as a special case of kernel SVM where identity mapping is used as the kernel. The loss function of kernel SVM is almost identical to eq. (2.11) except for replacing the term x with $\phi(x)$:

$$
\begin{aligned}
\min \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_i^n \xi_i \\
\text{s.t.} \quad & y_i(w^T\phi(x)_i + b) \geq 1 - \xi_i \\
& \xi_i \geq 0 \quad \text{for all } 1 \leq i \leq n
\end{aligned}
\tag{2.13}
$$

By introducing the Lagrangian term to the primal (2.13) and some transformation, we obtain

the dual of kernel SVM function:

$$\max \quad \sum_i^n \alpha_i - \sum_i^n \sum_j^n y_i y_j \alpha_i \alpha_j \phi(x_i)\phi(x_j)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq \frac{1}{\lambda} \tag{2.14}$$

$$\sum_i^n y_i \alpha_i = 0 \quad \text{for all } 1 \leq i \leq n$$

We can obtain the solution of (2.14) by Sequential Minimal Optimization (SMO) [81] or Dual Coordinate Descent [46].

There are several advantages of using SVM as the classifier:

- **Generalization ability.** SVM provides good generalization ability by maximizing the margin between the examples of the two classes. By setting the proper parameters and generalization grade, SVM can overcome some bias from the training set. Therefore, SVM is able to make correct prediction for unseen data. This ability can be very useful for image recognition as there is no image dataset that can cover all the transformation of the objects. Moreover,the idea of soft-margin makes it robust against noisy data.

- **Kernel transformation.** By introducing the non-linear transformation of the input, SVM can model complex non-linear distributed data. The kernel trick can greatly improve the computational efficiency.

- **Unique solution.** The objective function of SVM is convex. Compared to other methods, such as Neural Networks, which are non-convex and have many local minima, SVM can deliver a unique solution for any given training set and can be solved with efficient methods, like sub-SGD [90] or SMO [81].

2.1.4   Convolutional Neural Networks

Convolutional Neural Networks [65] is the most popular and powerful method for image recognition task. In this part, we introduce the history of Convolutional Neural Networks. The detail description of the Convolutional Neural Networks layers will be included in chapter 5.

2.1.4.1   Early Work with Convolutional Neural Networks

The first simple version of Neural Networks (NNs) trained with supervised learning was proposed in 1960s [85][86]. Networks trained by the Group Method of Data Handling (GMDH)

could be the first DL systems of the Feedforward Multilayer Perceptron type [48][88]. Later, there have been many applications of GMDH-style nets [38] [47] [57] [108].

Apart from deep GMDH networks, the Neocognitron, a hierarchical, multilayered artificial neural network, was perhaps the first artificial NN to incorporate the neurophysiological insights [40]. Inspired by Neocognitron, Convolutional NNs (CNNs) was proposed where the rectangular receptive field of a convolutional unit with given weight vector is shifted step by step across a 2-dimensional array of input values, such as the pixels of an image (usually there are several such filters). The results of the previous unit can provide inputs to higher-level units, and so on. Because of its massive weight replication, relatively few parameters may be necessary to describe its behavior.

In 1989, backpropagation [64][65] was applied to Convolutional Neural networks with adaptive connections [64]. This combination, incorporating with Max-Pooling and speeding up on graphics cards has become an important part for many modern, competition-winning, feedforward, visual Deep Learners. Later, CNNs achieved good performance on many practical tasks such as MNIST and fingerprint recognition and was commercially used in these fields in 1990s [7] [63].

In the early 2000s, even though GPU-MPCNNs wons several official contests, many practical and commercial pattern recognition applications were dominated by non-neural machine learning methods such as Support Vector Machines (SVMs).

2.1.4.2   Recent Achievements with Convolutional Neural Networks

In 2006, CNN trained with backpropagation set a new MNIST record of 0.39% without using unsupervised pre-training [75]. Also in 2006, an early GPU-based CNN implementation was introduced which was up to 4 times faster than CPU-CNNs [20]. Since then, GPUs or graphics cards have become more and more essential for CNNs in recent years. In 2012, a GPU implemented Max-Pooling CNNs (GPU-MPCNNs) was also the first method to achieve human-competitive performance (around 0.2%) on MNIST [21].

In 2012, an ensemble of GPU-MPCNNs (called AlexNet) achieved best results (top-5 accuracy at 83%) on the ImageNet classification benchmark (ILSVRC2012), which contains 1000 classes and 1.2 million images [58]. After that, excellent results have been achieved by GPU-MPCNNs in image recognition and classification. Many attempts have been made to improve the architecture of AlexNet. With the help of high performance computing systems, such as GPUs and large scale distributed cluster, some improvements have been made by either making the network deeper or increasing the size of the training data (with extra training example and data argumentation). By reducing the size of the receptive field and stride, Zeiler and Fer-

gus improve AlexNet by 1.7% on top 5 accuracy [115]. By both adding extra convolutional layers between two pooling layers and reducing the receptive field size, Simonyan and Zisserman built a 19 layer very deep CNN and achieved 92.5% top-5 accuracy [92]. After the AlexNet-like deep CNNs won ILSVRC2012 and ILSVRC2013, Szegedy et al. built a 22-layer deep network, called GoogLeNet and won the 1st prize on ILSVRC2014 for 93.33% top-5 accuracy, almost as good as human annotation[96]. Different from AlexNet-like architecture, GoogLeNet shows another trend of design, utilizing many 1×1 receptive field. Recently, Wu et. al present an image recognition system by aggressive data augmentation on the training data, achieving a top-5 error rate of 5.33% on ImageNet dataset[109]. Searchers from Google successfully trained an ingredient detector system based on GoogLeNet with 220 million images harvested from Google Images and Flickr [74].

Besides its impressive performance on those huge datasets, MPCNNs shows some impressive results by fine-tuning the existing models on small datasets.Zeiler et al. applied their pre-trained model on Caltech-256 with just 15 instances per class and improved the previous state-of-the-art in which about 60 instances were used, by almost 10% [115]. Chatfield et al. used their pre-trained model on VOC2007 dataset and outperformed the previous state-of-the-art by 0.9% [19]. Zhou et al. trained AlexNet for Scene Recognition across two datasets with identical categories and provided the state-of-the-art performance using our deep features on all the current scene benchmarks [116]. Hoffman et al. fine-tuned the MPCNNs trained from ImageNet with one example per class, showing that it is possible to use a hybrid approach where one uses different feature representations for the various domains and produces a combined adapted model [45].

To summarize, in this section, we have briefly reviewed three types of classifier for image recognition, namely Softmax classifier, SVM and CNNs. Softmax classifier is typically used as the last layer of CNNs. Linear SVM is a general method for the image recognition. Moreover, The generalization ability of SVM classifier can reduce the bias from the training data.

2.2   An Overview of Visual Transfer Learning

Traditional machine learning algorithms try to build the classifiers from a set of training data and apply to the test data with the same distribution to the training data. In contrast, transfer learning attempts to change this by transfer the learned knowledge from one or several tasks (called **source tasks**) to improve a related new task (called **target task**). According to the situations of the source and target tasks, transfer learning can be categorized as 3 types: inductive transfer learning, transductive transfer learning and unsupervised transfer learning. On the other hand, from the types of the source knowledge, transfer learning can be classified as:
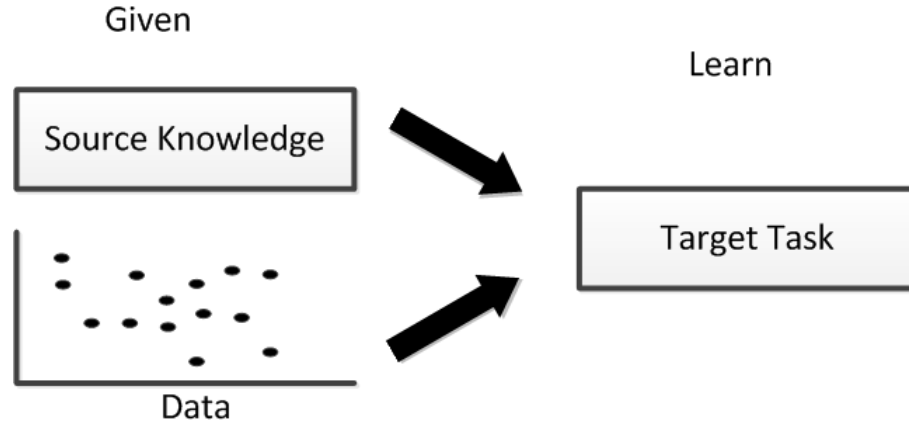
Figure 2.5: Apart from the standard machine learning, transfer learning can leverage the information from an additional source: knoweldge from one or more related tasks.

instance transfer, feature representation transfer and parameter transfer.

Table 2.1: Categories of our learning scenarios

	Situation of Task	Source Knowledge
Learning New Categories	Inductive Transfer	Parameter Transfer
Domain Adaptation	Transductive Transfer	

2.2.1 Types of Transfer Learning from the Situations of Tasks

Transfer learning can be categorized into 3 sub-settings: **inductive transfer learning, transductive transfer learning** and **unsupervised transfer learning** based on the different situations of the source and target domains and tasks. [79]. We compared the differences of these three sub-categories and show them in Table 2.3.

Table 2.2: Relationship between traditional machine learning and different transfer learning settings

Learning settings		Source target domain	Source target task
Traditional machine learning		the same	the same
Transfer learning	**Inductive transfer learning**	the same	different but related
	Unsupervised transfer learning	different but related	different but related
	Transductive transfer learning	different but related	the same

Table 2.3: Various settings of transfer learning

	Related areas	Source data	Target data
Inductive transfer learning	self-taught learning	unlabeled	labeled
	multi-task learning	labeled	labeled
Transductive transfer learning	**domain adaptation**, Sample selection bias	labeled	labeled/unlabeled
Unsupervised transfer learning		unlabeled	unlabeled

2.2.2 Types of Transfer Learning from the Aspect of Source Knowledge

According to the type of the source knowledge comes from, transfer learning can be split into 3 major streams: instance transfer, feature representation transfer and parameter transfer.

The core idea of instance transfer learning is to select some useful data from the source task to help learning the target task. Dai et al. [25] propose a method (called TrAdaBoost) that can select the most useful examples from the source task as the additional training examples for the target task. These useful examples are iteratively re-weighted according to the classification results of some base classifiers. Jiang et al. [51] proposed a method that can ignore the "misleading" examples from the source data based on the conditional probabilities on the source task $P(y_t|x_t)$ and target task $P(y_s|x_s)$. Liao et al. [66] proposed a active learning method that selects and labels the unlabeled data from the target data with the help of the source data. Ben-David et al. [9] provided a theoretical analysis the lowest target test error for different source data combination strategies when the source data is large and target training set is small.

Feature representation transfer aims to find a good feature representations to reduce the gap between the source and target domains. According to the size of labeled examples in the source data, feature representation transfer consists of two approaches: supervised feature construction and unsupervised feature construction. When the source data are labeled, supervised feature transfer learning is used to find the feature representations shared in related tasks to reduce the difference between the source and target tasks. Evgeniou et al. [34] proposed a method that can learn sparse low-dimension feature representations that can share between different tasks. Jie et al. [52] reconstructed the feature representations for the target data by using the outputs of the source models as the auxiliary feature representations. In unsupervised feature representation transfer learning, Daume III [27] proposed a simple feature reconstruction method for both source and target data so that source and target data are triple augmented and a SVM model is trained on both source and target data.

Parameter transfer assumes that there should be some parameters or prior distribution of
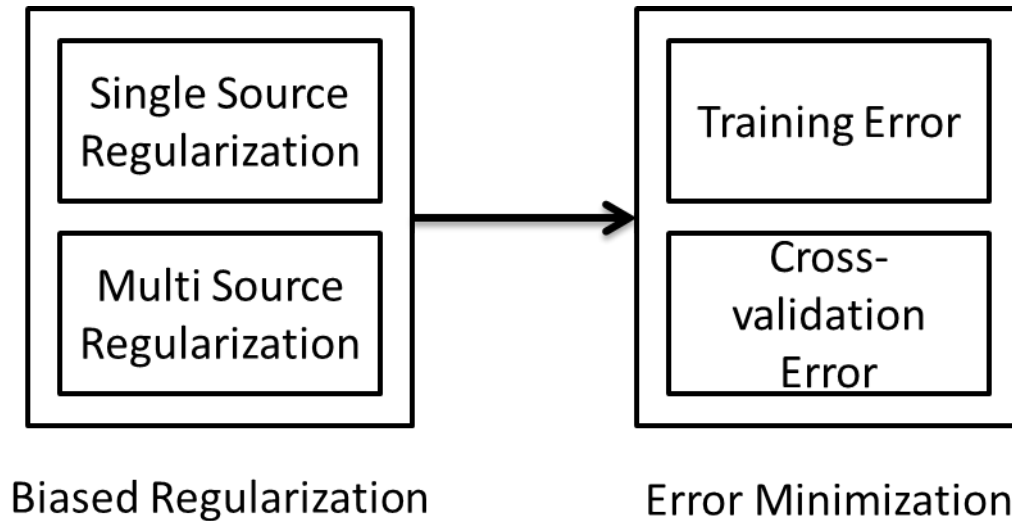
Figure 2.6: Two steps for parameter transfer learning. In the first step multi-source and single source combination are usually used to generate the regularization term. The hyperplane for the transfer model can be obtained by either minimizing training error or cross-validation error on the target training data.

the hyperparameters in the individual models of related tasks. Most of the approaches are designed under the multi-task learning scenario. Therefore, in this thesis, we also focus on the parameter transfer approach to leverage the knowledge from the source data. In parameter transfer learning, there are three major frameworks: a regularization framework, a Bayesian framework and a neural network framework.

- Regularization framework: In the regularization framework, some researchers propose to transfer the parameters of the SVM following the assumption that the hyperplane for the target task should be related to the hyperplane of the source models. Evgeniou et al. [35] proposed an idea that the hyperplane of the SVM for the target task should be separate into two terms: a common term shared over tasks and a specific term related to the individual task. Inspired by this idea, some researchers propose different strategies to combine these two terms for transfer learning [4] [98] [112]. Most of these work contains two steps. In the first step, a SVM objective function with a biased regularization term for the target model is built. Then another objective function is built to reduce the empirical error of the target model on the target data.

- Neural network framework. In neural network framework, the idea is to use the parameters of a CNN pre-trained from a very large dataset as an initialization to reduce the target data bias when it is small. Yosinski et al. [114] show that the high level layer parameters

are more related to a specific task while the low level layer ones are more general and transferable. This framework is widely used for image recognition task. By re-using and fine-tuning the parameters of some layers in the pre-trained model, the bias of the target task can be greatly alleviated [19] [45] [115] [116].

- Bayesian framework. In Bayesian framework, one or several posterior probabilities of the source data or parameters of the source model can be used to generate a prior probability for the target task. With this prior probability, a posterior probability for the target task can be obtained with the target data. Li et al. [39] used a prior probability density function to model the knowledge from the source and modify it with the data from target to generate posterior density for detection and recognition. Rosenstein et al. [87] used hierarchical Bayesian method to estimate the posterior distribution for all the parameters and the overall model can decide the similarities of the source and target tasks.

2.2.3 Special Issues in Avoiding Negative Transfer

In transfer learning, for a given target task, the performance of a transfer method depends on two aspects: the quality of the source task and the transfer ability of the transfer algorithm. The quality of the source task refers to how the source and target tasks are related. If there exists a strong relationship between the source and target, with a proper transfer method, the performance in the target task can be significantly improved. However, if the source and target tasks are not sufficiently related, despite of the transfer ability of the transfer algorithm, the performance in the target task may fail to be improved or even decrease. In transfer learning, negative transfer refers to the degraded performance compare to a method without using any knowledge from the source [79]. How to avoid negative transfer is still an open question for researchers. For example, we can use a teacher-student diagram to illustrate the procedure of transfer learning. The student (target model) would like to learn the new knowledge (target task) with the assistance of a teacher (source knowledge). If the teacher can provide helpful knowledge (related knowledge), the student can learn the new knowledge very quickly (positive transfer). If the teacher can only provide useless knowledge, the student could not learn the new knowledge effectively or even get confused (negative transfer).

Another important aspect that affect the learning performance is the transfer ability of the algorithm used for the target task. An ideal transfer algorithm would be able to produce positive transfer on related tasks while avoiding negative transfer on unrelated tasks. However, in practice, it is not easy to achieve these two goals simultaneously. Approaches that can avoid negative transfer often bring some affects on positive transfer due to their caution. On the other hand, approaches using aggressive transfer strategies often have little or no protection against
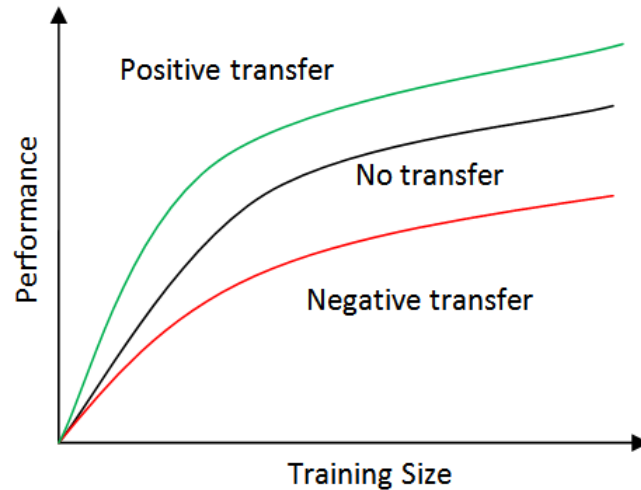
Figure 2.7: Positive transfer VS Negative transfer.

negative transfer [100]. Even though voiding negative transfer is an important issue in transfer learning, how to avoid negative transfer has not been widely addressed [71] [79]. Previous work show suggest that negative transfer can be alleviated through 3 approaches [100]:

- Rejecting unrelated source information. A important approach to avoid negative transfer is to recognize and reject unrelated and harmful source knowledge. The goal of this approach is to minimize the impact of the unrelated source, so that the transfer model performs no worse than the learned model without transfer. Therefore, in some extreme situation, the transfer model is allowed to completely ignore the source knowledge. Torrey et al. [101] proposed a method using advice-taking algorithm to reject the unrelated source knowledge. Rosenstein et al. [87] presented an approach that use naive Bayes classifier to detect and reject the unrelated source.

- Choosing correct source task. When the source knowledge come from more than one candidate source, it is possible for the transfer model to select the best source knowledge of the candidates. In this scenario, leverage the knowledge from the best candidate may be effective against negative transfer as long as the best source knowledge is sufficiently related. Talvitie et al. [97] proposed a method that can iteratively evaluate the candidate sources through a trail-and-error approach and select the best one to transfer. Kuhlmann et al. [59] constructed a kernel function from certain sources for the target task by estimating the bias from a set of candidate sources whose relationship to the target task is unknown.

- Measuring task similarity. To achieve a better transfer performance, it is reasonable for a transfer method to transfer the knowledge from multiple sources instead of just choosing

a single source. In this approach, some methods try to involve all the source knowledge without considering the explicit relationship between the source and target. The other methods try to model the relationships between the source and target tasks and use the information as a part of their transfer methods which can significantly reduce the affect of negative transfer. Bakker et al. [6] proposed a method to provide guidance on how to avoid negative transfer by using clustering and Bayesian approach to estimate the similarities between the target task and multiple source tasks. Tommasi et al. [99] constructed the transfer model by using some transfer parameters to measure the relationships between each source and the target tasks and the transfer parameters are optimized by minimizing the cross-validation error of the transfer model. Similar approaches can be found in [52] [61]. Kuzborskij et al. [60] provided some theoretical analysis of transfer learning and show that regularized least square SVM with truncation function and leave-one-out cross-validation for source task measurement can reduce negative transfer even though the training data of the target task is relatively small.

Here we can see that most of the previous approaches focus on measuring the similarity of the source and target tasks, i.e. try to assign the most related source tasks to the target one through various of metrics and use aggressive transfer algorithm to exploit the source knowledge. Just a few work [60] [98] addressed the problem that a sophisticated transfer algorithm should be designed to better exploit the source knowledge as well as avoid negative transfer. Therefore, in this thesis, we mainly focus on how to design a better transfer algorithm for transfer learning while certain source knowledge is assigned.

To summarize, in this section, we provided an overview of the categories of transfer learning from two different views. From the relationships of the tasks, our two transfer learning scenarios belong to inductive transfer and transductive transfer learning respectively. In this thesis, we assume that we are not able to access to the source data, therefore, from the aspect of source knowledge, we use parameter transfer for both scenarios. In this thesis, without observing any source data, it is difficult to measure the relationship between the source and target tasks. Therefore, avoiding negative transfer is also an important part we have to consider. Finally, we reviewed some methods that cab alleviate negative transfer.

2.3   Related Work in Hypothesis Transfer Learning

The main topic of this thesis is to investigate the problem of visual transfer learning under the HTL setting. Therefore, in this section, we show some work related to this topic and this thesis. We first review some work in fine-tuning the deep neural networks for inductive transfer

learning related to our work in chapter 5. Then we discuss some methods in hypothesis transfer learning, which is related to our work in chapter 3. The related work in chapter 4 is reviewed at last.

2.3.1 Fine-tuning the Deep Net

Since Deep Convolutional Neural Networks (CNNs) became the most powerful algorithm in object recognition task, fine-tuning the deep CNNs has become a popular and effective way to transfer the knowledge between different visual recognition tasks. The intuition of fine-tuning the deep CNNs for transfer learning is that low-level features, such as edges and lines, are universal for object recognition while high-level features, which are the combinations of the low-level features, are more specific for the designed task. Because deep CNNs can learn hierarchical features, from abstract low-level features to detailed high-level ones, by changing the combinations of the low-level features in the pre-trained deep CNNs, the high-level features can be learned effectively for the new recognition task [37].

Applying the pre-trained model from ImageNet dataset on other object recognition benchmark datasets shows some impressive results. Zeiler et al. [115] applied their pre-trained model on Caltech-256 with just 15 instances per class and improved the previous state-of-the-art in which about 60 instances were used, by almost 10%. Chatfield et al. [19] used their pre-trained model on the VOC2007 dataset and outperformed the previous state-of-the-art by 0.9%. Agrawal et al. [1] show that even in the mid-level features, there are some grandmother cells, which can capture the high-level features of specific objects. Hoffman et al. [45] show that even with one labeled example per class, it is possible to fine-tune the pre-trained deep CNNs and obtain a good classifier for some new recognition tasks. Zhou et al. [116] provided the state-of-the-art performance using the deep features on some scene benchmarks by fine-tuning the deep CNNs. Yosinski et al. [114] investigated the transferability of the layers in deep CNNs and show that the target task can be benefited from pre-training even though the source and target tasks are distant.

In this thesis, we also use the pre-trained deep CNNs to learn new categories for food recognition and investigate the affects of each layer in deep CNNs for knowledge transfer in chapter 5.

2.3.2 Hypothesis Transfer Learning with SVMs

In this part, We will introduce the framework of **Hypothesis Transfer Learning** (HTL). HTL is proposed to effective utilize the source knowledge, especially the source model, while we are not able to visit the source data. We first introduce the basic principle of LS-SVM. Based on
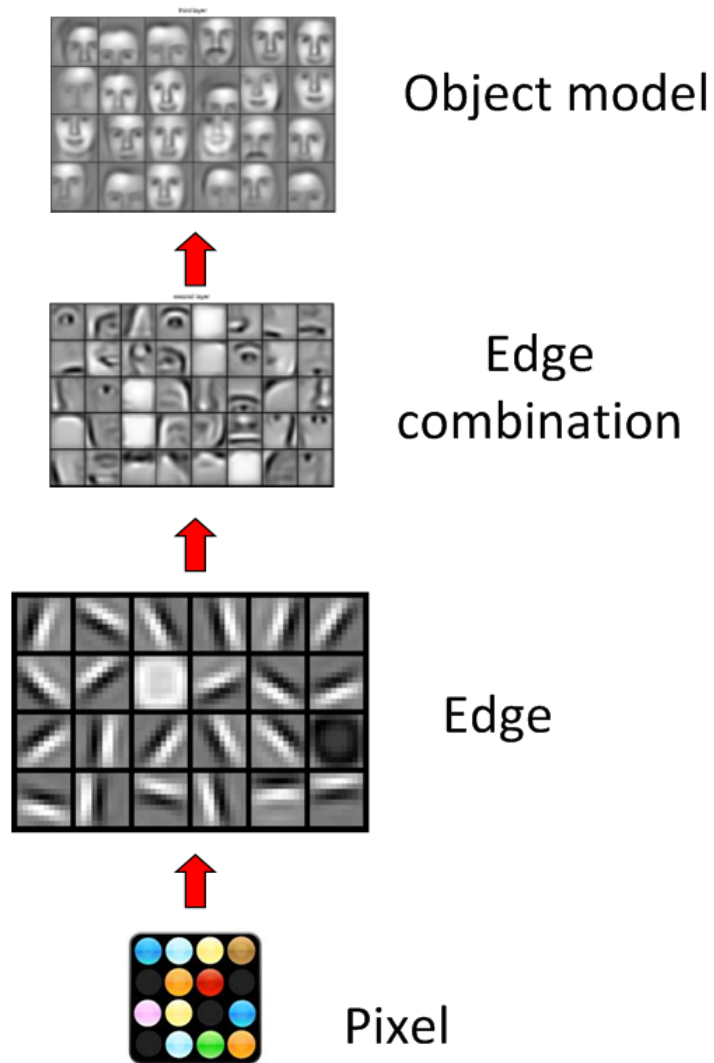
Figure 2.8: Hierarchical Features of Deep Convolutional Neural Networks for face recognition.

LS-SVM, several transfer methods are introduced, including A-SVM, PMT-SVM, Multi-KT and MULTIpLe. In these methods, the first two can only adopt the knowledge from single source model and the rest can adopt the knowledge from multiple ones.

2.3.2.1 LS-SVM Classifier

Least Square SVM is proposed is least squares versions of support vector machines (SVM), which are a set of related supervised learning methods that analyze data and recognize patterns [95]. By replacing the hinge loss in classical SVM with L2 loss, LS-SVM classifier is obtained by reformulating the minimization problem as:

$$\min \quad L_{LSSVM} = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}\varepsilon_i^2$$

$$\text{s.t.} \quad y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\}$$

(2.15)

The primal Lagrangian for this optimization problem given the unconstrained minimization problem can be written as:

$$L(w, b, \alpha, \varepsilon) = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}\varepsilon_i^2 - \sum_{i=1}^{l}\alpha_i\{wx_i + b + \varepsilon_i - y_i\}$$

(2.16)

Where $\alpha = [\alpha_1, ..., \alpha_l]^T$ is the vector of Lagrange multipliers. The solution to minimise this problem is give by:

$$\begin{bmatrix} K + \frac{1}{C}I & 1 \\ 1^T & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

(2.17)

Where $K \in R^{l \times l}, K_{i,j} = x_i \times x_j^T$. I is the identity matrix and **1** is a column vector with all its elements equal to 1. With:

$$\psi^{-1} = \begin{bmatrix} K + \frac{1}{C}I & 1 \\ 1^T & 0 \end{bmatrix}^{-1}$$

(2.18)

Problem (2.16) can be solved by:

$$\begin{bmatrix} \alpha \\ b \end{bmatrix} = \psi^{-1}\begin{bmatrix} y \\ 0 \end{bmatrix}$$

(2.19)

2.3.2.2 ASVM & PMT-SVM

Adaptive SVM (ASVM) is the first work using LS-SVM for transfer learning for vision related tasks [111]. The goal of ASVM is to minimize the distance between the target hyperplane w and source one w' incorporating with the transfer parameter γ. The objective function is defined as follow:
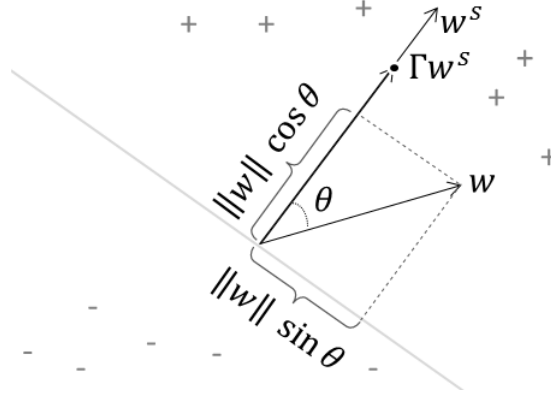
Figure 2.9: Projecting w to w' in PMT-SVM (adapted from [4]).

$$\min \quad L_{ASVM} = \frac{1}{2}\|w - \gamma w'\|^2 + \frac{C}{2}\sum_{i=1}^{l} \varepsilon_i^2 \tag{2.20}$$

$$\text{s.t.} \quad y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\}$$

Here, γ controls the amount of transfer regularization. Intuitively, the regularization term of ASVM is like a spring between w and $\gamma w'$. Equivalently, assume $\|w'\|^2 = 1$ and the regularization term can be expended as:

$$\|w - \gamma w'\|^2 = \|w\|^2 - 2\gamma \|w\| \cos\theta + \gamma^2$$

Where θ is the angle between w and w'. However, the term $-\gamma \|w\| \cos\theta$ also encourages $\|w\|$ to be larger (as this reduces the cost) which prevents margin maximization. Thus γ, which defines the amount of transfer regularization, becomes a trade-off parameter between margin maximization and knowledge transfer.

Based on this, Projective Model Transfer SVM (PMT-SVM) is proposed to solve the transfer problem by optimizing the following objective function [4]:

$$\min \quad L_{PMT} = \frac{1}{2}\|w\|^2 + \gamma\|Pw'\|^2 + \frac{C}{2}\sum_{i=1}^{l} \varepsilon_i^2$$

$$\text{s.t.} \quad y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \tag{2.21}$$

$$w^T w' \geq 0$$

Where P is the the projection matrix $P = I - \frac{w'^T \times w'}{w' \times w'^T}$. Therefore, $\|Pw\|^2 = \|w\|^2 \sin^2\theta$ is the squared norm of the projection of the w onto the source hyperplane (see Figure 2.9). As $\gamma \to 0$, the loss function (2.21) becomes a classic LS-SVM loss function. Because (2.21) is convex, it can be solved effectively by quadratic optimization.

In summary, both ASVM and PMT-SVM are designed to answer the question: how to transfer by solving some convex objective function. However, they both require a pre-defined parameter γ, which controls the amount of the knowledge to be transfered, to complete the objective function. In most cases, this parameter can only be set according to the background knowledge. Also, both methods can only adopt the knowledge from single source model which limits the their performance. **Multi-KT** To learning from many sources, Multi Model Knowledge Transfer (Multi-KT) is proposed to reply on multiple sources by assigning different weight for each of them [99]. Similar to the objective function (2.20), its objective function is defined as follow:

$$
\min \quad L_{Multi-KT} = \frac{1}{2}\left\|w - \sum_k \beta_k w'_k\right\|^2 + \frac{C}{2}\sum_{i=1}^{l}\zeta_i \varepsilon_i^2 \tag{2.22}
$$

$$
\text{s.t.} \quad y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\}
$$

Here, β_k is the weight assigned for the kth source model and ζ_i is defined as:

$$
\zeta_i = \begin{cases} \frac{N}{2N^+} & \text{if } y_i = 1 \\ \frac{N}{2N^-} & \text{if } y_i = -1 \end{cases}
$$

where N^+ and N^- are number of positive and negative examples respectively and N is the total number of examples.

The primal Lagrangian for optimization problem (2.22) can be written as:

$$
L_{Multi-KT}(w, \beta, \varepsilon, \alpha) = \frac{1}{2}\left\|w - \sum_k \beta_k w'_k\right\|^2 + \frac{C}{2}\sum_{i=1}^{l}\zeta_i \varepsilon_i^2 + \sum_{i=1}^{l}\alpha_i\left[wx_i + b + \varepsilon_i - y_i\right] \tag{2.23}
$$

So problem (2.22) can be solved once β is set. Different from ASVM and PMT-SVM which require background knowledge to select proper transfer parameter, Multi-KT can estimate the transfer parameter β itself by using the closed-form Leave-One-Out (LOO) error. According to [18], the closed-form LOO error is defined as:

$$
y_i - \hat{y}_i = \frac{\alpha_i}{\psi_{ii}^{-1}} \quad \text{for} \quad i = 1, ..., l \tag{2.24}
$$

Here ψ_{ii}^{-1} is its ith diagonal element of ψ^{-1} in (2.18).

To estimate β, a loss function, similar to hinge loss, is defined as:

$$
\min \quad \mathcal{L} = \sum_{i}^{l}\zeta_i|1 - y_i\hat{y}_i|_+ \tag{2.25}
$$

$$
\text{s.t.} \quad \|\beta\| \leq 1
$$

Here $|x|_+ = max(x, 0)$. Intuitively, if β is properly set, $y_i \hat{y}_i$ should be positive for each i. However, focusing only on the sign of those quantities would result in a non-convex formulation with many local minima. By adding the $|\cdot|_+$ function, formula (2.27) becomes convex and can be solved by gradient descent method.

2.3.2.3 MULTIpLE

MULticlass Transfer Incremental LEarning (MULTIpLE) focuses on adding a new class to an existing N class source problem while preserving the performance on the old classes [61]. To preserving the overall performance of the classifier among $N + 1$ classes, MULTIpLE contains two parts: incremental learning for existing N classes and transfer learning for the new class.

For the existing N classes, MULTIpLE uses the similar strategy as ASVM, setting the transfer parameter γ to 1. For the new class, it adopts the strategy of Multi-KT, combining knowledge from existing N-class models. As a result, the objective function for the hyperplanes in MULTIpLE is defined as:

$$\min \quad L_{MULTIpLE} = \frac{1}{2} \sum_{n=1}^{N} \|w_n - w'_n\|^2 + \frac{1}{2} \left\| w_{N+1} - \sum_{k=1}^{N} w'_k \beta_k \right\|^2 + \frac{C}{2} \sum_{n=1}^{N+1} \sum_{i=1}^{l} \varepsilon_{i,n}^2 \quad (2.26)$$

$$\text{s.t.} \quad \varepsilon_{i,n} = Y_{in} - x_i w_n - b_n$$

Similar like Multi-KT, MULTIpLE uses LOO error in (2.24) to estimate the transfer parameter β for the new class. The objective function for β estimation is defined by [23]:

$$\min \quad \mathcal{L}(\beta, i) = \begin{cases} \max_{n \neq y_i} \left| 1 - \hat{Y}_{in}(\beta) - \hat{Y}_{iy_i}(\beta) \right|_+ & : y_i \neq N + 1 \\ \left| 1 - \hat{Y}_{in}(\beta) - \hat{Y}_{iy_i}(\beta) \right| & : y_i = N + 1 \end{cases} \quad (2.27)$$

$$\text{s.t.} \quad \|\beta\| \leq 1$$

We can find the optimal β with projected subgradient descent [17].

These previous methods provided an approach to transfer the knowledge under the HTL setting. The previous methods try to use the hyperplane parameter as the transferable knowledge and thus limited to leverage the source knowledge from SVMs. In chapter 3 of this thesis, we extend their methods and propose a novel method EMTLe under the HTL setting which can leverage the knowledge from more types of source models.

2.3.3 Distillation for Knowledge Transfer

Distillation [43] was developed frameworks for knowledge transfer and addressed the problem how to effectively transfer the knowledge from the source model directly. In chapter 4, we

propose a framework called GDSDA based on it to solve semi-supervised domain adaptation problem. In this part, we review the principle of Distillation and some related work using this framework. The technical details will be introduced in chapter 4.

Hinton et al. proposed Distillation to transfer the knowledge from a source neural network (or a whole ensemble of neural networks) to a single target one. In this setting, the capacity of the source neural network is large while the capacity of the target one is small. The capacity reflects the expressive power of a model and a model with larger capacity can fit complex data better. In statistical learning theory [107], the relationship of the generalization error e_{te} and training error e_{tr} of a model can be bounded as follow:

$$e_{te} \leq e_{tr} + 2\sqrt{\frac{\log h + \log \frac{2}{\eta}}{2N}} \tag{2.28}$$

Here, h is the capacity (VC dimension) of the model and N is the training set size. Inequation 2.28 holds with a probability of $1 - \eta$. When we train the target model, we can let it mimick the output of the source one on the training set. If both source and target models can achieve similar training error on the training set, the small target model can typically do much better on the test data due to its lower capacity. In this process, we don't need the true label of the training data. Instead, we only require the outputs of the source model of the training data. The output of the source model is usually called the "soft label". However, introducing the true label of the training data can further improve the performance of the target model. In distillation, imitation parameter is used to balance the importance between the soft label and true label.

Distillation is typically used for training the deep neural network for knowledge transfer between different models and tasks. Tzeng et al. [103] proposed a CNN architecture for domain adaptation to leverage the knowledge from limited or no labeled data using the soft label. Urban et al. [104] use a small shallow net to mimick the output of a large deep net while using layer-wised distillation with ℓ_2 loss of the outputs of student and teacher net. Similarly, Luo et al. [72] use ℓ_2 loss to train a compressed student model from the teacher model for face recognition. Gupta et al. [42] use supervision transfer to distil the knowledge from a trained CNN with unlabeled data or just a few labeled data.

In this thesis, we propose a novel framework that uses distillation for domain adaptation. The limitation of the previous work in Distillation is that it is difficult to determine the value of the imitation parameter. Previous studies avoid this problem by either using brute force searching or domain knowledge. To better solve this problem, we proposed a novel method that can autonomously balance the importance and extend it to the semi-supervised domain adaptation.

2.4   Summary

In this chapter, we reviewed some concepts and work related to this thesis. We demonstrate the main methods related to our work and their limitations. In the following chapters, we will provide the technical details of this thesis.

Chapter 3

Effective Multiclass Transfer For Hypothesis Transfer Learning

3.1 Introduction

Domain adaptation for image recognition tries to exploit the knowledge from a source domain with plentiful data to help learn a classifier for the target domain with a different distribution and little labeled training data. In domain adaptation, the source and target domains share the same label but their data are drawn from different distributions.

Previous research [9, 10] shows that without carefully measuring the distribution similarity between the source and target data, the source knowledge could not be exploited effectively or even hurt the learning process (called *negative transfer*)[79]. However, as we are not able to access the source data in the Hypothesis Transfer Learning (HTL) [60] setting, how to effectively and safely exploit the knowledge from the source model could be an important issue in HTL, especially when target data is relatively small (Effectiveness issue). Moreover, the source models from different domains can be trained with different kinds of classifiers. For example, most models trained from ImageNet are deep convolutional neural networks while some models of the VOC recognition task could be SVMs or ensemble models. Therefore, a practical HTL algorithm should be compatible with different types of source classifiers (Compatibility issue). Previous work is limited to either leveraging the knowledge from a certain type of source classifiers [99, 39] or low transfer efficiency in a small training set[52]. To the best of our knowledge, none of the previous work in HTL is able to solve these two issues at the same time.

In this chapter, we propose our method, called **Effective Multiclass Transfer Learning** (EMTLe), that can solve these two issues simultaneously. We perform comprehensive ex-
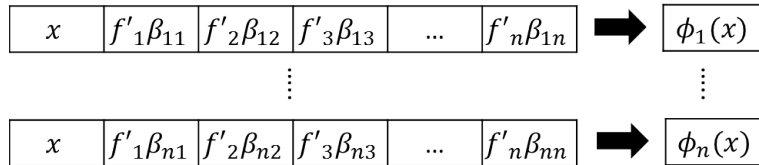
Figure 3.1: Illustration of feature augmentation in MKTL. f_i' is the output of the i-th source model and β_{in} is the hyperparameter (need to be estimated) to weigh the augmented feature. $\phi_n(x)$ is augmented feature for the n-th binary model.

periments on 4 real-world datasets from two benchmark datasets (3 from Office and 1 from Caltech256). We show that EMTLe can effectively transfer the knowledge with different types of source models and outperforms the baseline methods under the HTL setting.

This work has been accepted by *Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2017*.

3.2   Using the Source Knowledge as the Auxiliary Bias

Some previous work such as MKTL[52] suggests that using the prediction from the source model as the source knowledge can greatly release the constraint of the type of the source model. However, with complex feature augmentation method, there are many hyperparameters to be estimated which makes it inefficient with small training set. In this paper, we adopt the idea of using the source model prediction as the transferable knowledge and propose our transfer strategy.

Suppose we have to recognize an image from one of the N visual classes and there are N experts each of who can only provide the probability of this image for one certain class (binary source model). After we make our decision for one example (prediction from target model), the experts provide their decisions as well (probabilities from the source models). Their decisions can provide extra information regarding this example as the auxiliary bias and adjust our final prediction. As each of the experts is a specialist in one class, we should weigh their decisions as well due to the bias of their predictions.

Suppose our target is to distinguish the bottles of 4 pop while our source model can distinguish the same 4 pops but in cans (see Figure 3.2). Apparently, there are some connections between the source and target (such as the color information). In our solution, we treat the knowledge from source (the prediction of the source model) as a prior to affect the output of our final model. Due to the domain shift, we have to reweigh the source knowledge. As long as we can choose the proper weight, we are able to obtain a good target model. It is important
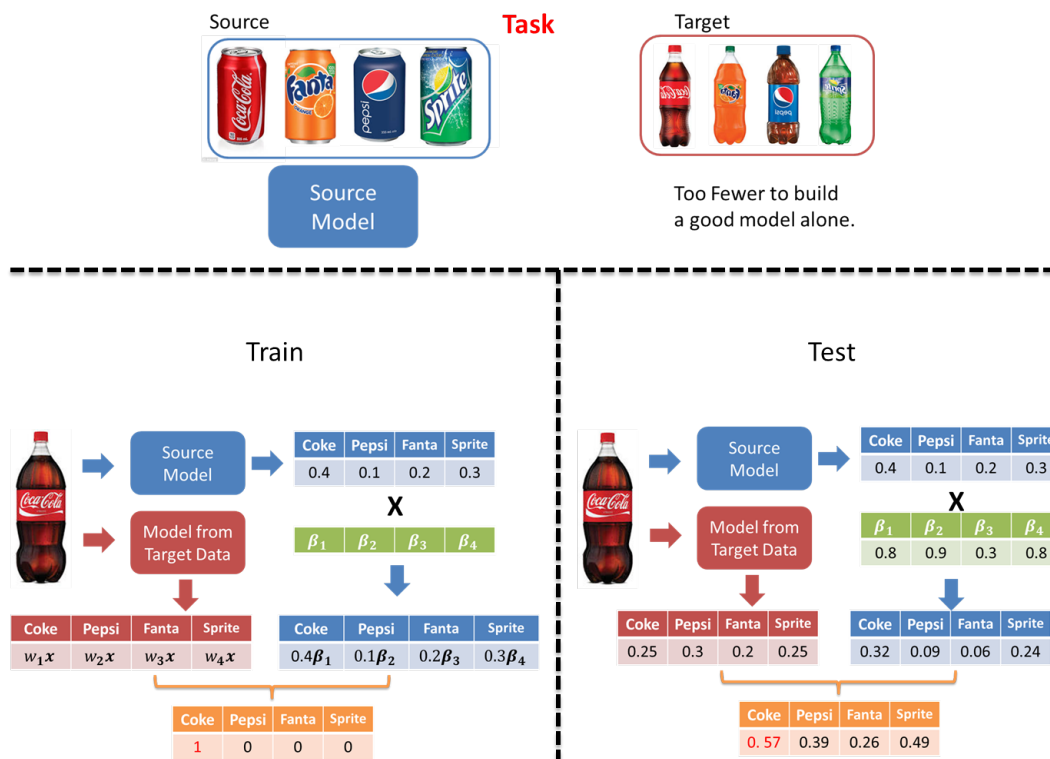
Figure 3.2: Demonstration of using the source class probability as the auxiliary bias to adjust the output of the target model. The source task is to distinguish the 4 pop cans while the target one is to distinguish the 4 pop bottles

to choose the transfer parameter in our solution.

Here, the weight of each source model reflects the relatedness between the source model and our target domain. The more related they are, the better decision the source model can make and the larger weight we should apply to it. Specifically, in this paper, we call the weight *transfer parameter*. Therefore, for any target data $D = \{x, y\}$ and the given source models $f' = \{f'_1, ..., f'_N\}$, our goal is to find the target model f:

$$f = \arg\min_{f \in \mathcal{F}} \ell \left(f + \beta f' | D, \beta \right) \tag{3.1}$$

where $\beta = [\beta_1, ..., \beta_N]$ is the transfer parameter and $\ell(\cdot, \cdot)$ is the loss function to learn the target model. It is obvious that assigning the proper transfer parameter to the source model can significantly improve the performance of our final prediction. From Eq. (3.1) we can see that, once we have determined the value of the transfer parameter β, we are able to find the target model f and solve the learning problem. However, the transfer parameter in Eq.(3.1) is a hyperparameter and we cannot solve it directly. Therefore, we introduce our bi-level optimization method for transfer parameter estimation in the next section.

Unlike previous work[4, 99, 111] which has to use the specific parameter of the source model as the source knowledge, our strategy is more compatible with different types of classifiers. Compared to MKTL[52], we only have to estimate N hyperparameters for the N-class problem while there are $N \times N$ hyperparameters in MKTL (see Figure 3.1). Therefore, it is easier to estimate the transfer parameters with our strategy and EMTLe can perform better especially when the size of the training set is small. In addition, there are two advantages of our strategy: (1) It is a practical and easy way to align the knowledge from different types of source classifiers. (2) The auxiliary bias term is naturally normalized in the same dimension as the class probabilities are always in the interval [0, 1]. As EMTLe can select more types of source classifiers, this makes it more practical in a real HTL scenario.

3.3 Bi-level Optimization for Transfer Parameter Estimation

As we discussed before, the transfer parameter in Eq. (3.1) is a hyperparameter that cannot be solved directly. Here we use bi-level optimization (**BO**)[80], a popular method that is used in hyperparameter optimization to estimate the transfer parameter. In BO, the low-level optimization problem is to learn the target model and the high-level problem is another cross-validation (CV) hyperparameter optimization problem corresponding to the model learned at the low-level.
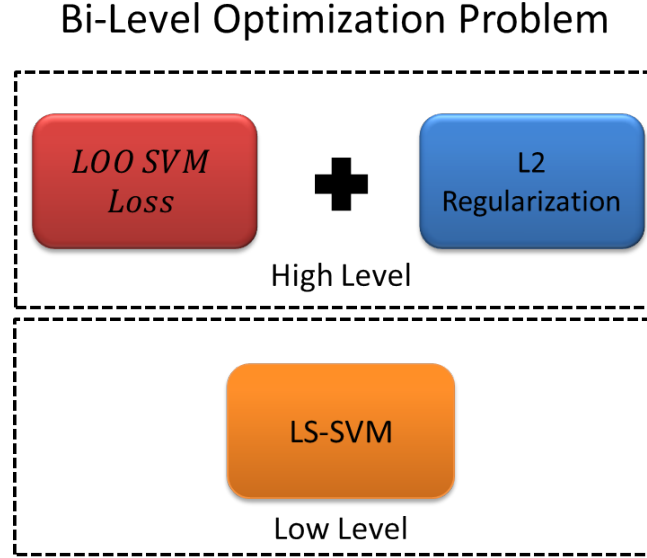
Figure 3.3: Bi-level Optimization problem for EMTLe.

Suppose we use K-fold CV on the high-level problem. For the i-th fold CV, the target set D is split into training set D_i^{tr} and validation set D_i^{val}. The transfer parameter can be optimized with the following BO function:

$$
\begin{aligned}
\text{High level} \quad & \beta = \arg\min_{\beta} \sum_{i}^{K} \mathcal{L}(f^i(\beta)|D_i^{val}) \\
\text{Low level} \quad & f^i(\beta) = \arg\min_{f \in \mathcal{F}} \ell \left(f + \beta f' | D_i^{tr}, \beta \right)
\end{aligned}
\tag{3.2}
$$

Here, $\ell(\cdot, \cdot)$ and $\mathcal{L}(\cdot, \cdot)$ are our low-level and high-level objective functions respectively. We can use any convex loss functions in Eq.(3.2) for optimization (e.g. SVM objective function). In this paper, we use the leave-one-out cross-validation (**LOOCV**) in the high-level problem. Previous research [60] suggests that LOOCV can increase the robustness of the estimated hyperparameter especially on the small dataset. In previous studies[73, 80], BO is a non-convex problem and can only obtain the local optimal solution. However, we will show that problem (3.2) is strongly convex and we are able to obtain its optimal solution.

3.3.1 Low-level optimization problem

To better illustrate our learning scenario, we define our learning process as follows. Suppose we have N visual categories and can obtain N source binary classifiers $f' = \{f'_1, ..., f'_N\}$ from the source domain. We want to train a target function f consisting of N binary classifiers $f = \{f_1, ..., f_N\}$ using the target training set D and the source models f'. Specifically, in our

BO problem Eq. (3.2), for the low-level optimization, we consider the scenario where we have to train N binary linear target models $f_i = w_i x + b_i$ so that for any $\{x_i, y_i\}_{i=1}^{l} \in D$, the adjusted result satisfies $f(x) + f'(x)\beta = y$. Let $D^{\backslash i} = D \backslash \{x_i, y_i\}$. Then, we use mean square loss in the low-level objective function to optimize each target model f_n with any given transfer parameter β:

$$\text{Low-level:} \quad f^{\backslash i}(\beta) : \min_{w,b} \sum_{n}^{N} \frac{1}{2} \|w_n\|^2 + \frac{C}{2} \sum_{j} e_{jn}^2$$

$$\text{s.t.} \qquad f_n(x) = w_n x + b_n; \quad x_j \in D^{\backslash i}$$

$$e_{jn} = Y_{jn} - f_n(x_j) - \beta_n f_n'(x_j)$$

(3.3)

Here, Y is an encoded matrix of y using the one-hot strategy where $Y_{in} = 1$ if $y_i = n$ and 0 otherwise.

The reason why we use the objective function (3.3) is that it can provide an unbiased closed form Leave-one-out error estimation for each binary model f_n[18]. As a result, the high-level problem becomes a convex problem and we are able to estimate our transfer parameter easier.

Let $K(X, X)$ be the kernel matrix and C be the penalty parameter in Eq.(3.3). We have:

$$\psi = \left[K(X, X) + \frac{1}{C} I \right]$$

(3.4)

Let ψ^{-1} be the inverse of matrix ψ and ψ_{ii}^{-1} is the ith diagonal element of ψ^{-1}. $\hat{Y}_{in}$, the LOO estimation of binary model $f_n^{\backslash i}$ for sample x_i, can be written as[18]:

$$\hat{Y}_{in} = Y_{in} - \frac{\alpha_{in}}{\psi_{ii}^{-1}} \quad \text{for} \quad n = 1, ..., N$$

(3.5)

where the matrix $\alpha = \{\alpha_{in} | i = 1, ...l; n = 1, ..., N\}$ can be calculated as:

$$\alpha = \psi^{-1} Y - \psi^{-1} f'(X) diag(\beta)$$

(3.6)

3.3.2 High-level optimization problem

For the high level optimization problem, we use multi-class hinge loss [23] with ℓ_2 penalty in our objective function.

$$\text{High-level:} \quad \beta : \min \frac{\lambda}{2} \sum_{n}^{N} \|\beta_n\|^2 + \sum_{i} \xi_i$$

$$\text{s.t.} \qquad 1 - \varepsilon_{ny_i} + \hat{Y}_{in} - \hat{Y}_{iy_i} \leq \xi_i$$

(3.7)

Here, $\varepsilon_{ny_i} = 1$ if $n = y_i$ otherwise 0. λ is used to balance the ℓ_2 penalty and our multi-class hinge loss. Compared to the previous work [61, 99] which uses the multi-class hinge loss without the ℓ_2 penalty, there are two main advantages for our high-level objective function:

1. When the training set is small, our LOOCV estimation could have a large variance. It is important to add the ℓ_2 penalty to reduce the variance and improve the generalization ability of the estimated transfer parameter.

2. It is clear that $\hat{Y}$ is a linear function w.r.t. β. With the ℓ_2 penalty, the high-level optimization problem (3.7) becomes a strongly convex optimization problem w.r.t. the transfer parameter β.Therefore, we can obtain an $O(\log(t)/t)$ optimal solution with t iterations using Algorithm 1 (see proof of Theorem A.2.1 in Appendix).

Algorithm 1 EMTLe

Input: $\lambda, \psi, Y, f', T,$
Output: $\beta = \left\{\beta^1, ..., \beta^n\right\}$
1: $\beta^0 = 1$,$\alpha' = \psi^{-1}Y, \alpha'' = \psi^{-1}f'$
2: **for** $t = 1$ to T **do**
3:     $\hat{Y} \leftarrow Y - \left(\psi^{-1} \circ I\right)^{-1} (\alpha' - \alpha'' diag(\beta))$
4:     **for** $i = 1$ to l **do**
5:         $\Delta_\beta = \lambda\beta$
6:         $l_{ir} = \max(1 - \varepsilon_{y_i r} + \hat{Y}_{ir} - \hat{Y}_{iy_i})$
7:         **if** $l_{ir} > 0$ **then**
8:             $\Delta_\beta^{y_i} \leftarrow \Delta_\beta^{y_i} - \frac{\alpha''_{iy_i}}{\psi_{ii}^{-1}}, \Delta_\beta^r \leftarrow \Delta_\beta^r + \frac{\alpha''_{ir}}{\psi_{ii}^{-1}}$
9:         **end if**
10:     **end for**
11:     $\beta^t \leftarrow \beta^{(t-1)} - \frac{\Delta_\beta}{\lambda \times t}$
12: **end for**

3.4 Experiments

In this section, we show empirical results of our algorithm for different transferring situations on two image benchmark datasets: Office and Caltech.

3.4.1 Dataset & Baseline methods

Office contains 31 classes from 3 subsets (Amazon, Dslr and Webcam) and Caltech contains 256 classes. We select 13 shared classes from two datasets[1]. The input features of all exam-

---

[1]13 classes include: backpack, bike, helmet, bottle, calculator, headphone, keyboard, laptop, monitor, mouse, mug, phone and projector

ples are extracted using AlexNet[58]. We compare our algorithm EMTLe with two kinds of

Table 3.1: Statistics of the datasets and subsets

Dataset	Subsets	# classes	# examples	# features
	Amazon (A)	13	1173	4096
Office	Dslr (D)	13	224	4096
	Webcam (W)	13	369	4096
Caltech256	Caltech (C)	13	1582	4096

baselines. The first one is the methods without leveraging any source knowledge (no transfer baselines), including two methods. **No transfer:** SVMs trained only on target data. Any transfer algorithm that performs worse than it suffers from negative transfer. **Batch:** We combine the source and target data, assuming that we have full access to all data, to train the SVMs. The result of the Batch method is expected to outperform other methods under the HTL setting as it can access the source data. The second kind of baseline consists of two previous transfer methods in HTL, **MKTL[52]** and **Multi-KT[99]**. Similar to EMTLe, both of them use the LOOCV method to estimate the relatedness of the source model and target domain, but they use their convex objective function without the ℓ_2 penalty terms. We use linear kernel for all methods in all our experiments.

3.4.2 Transfer from Single Source Domain

In this subsection, following the experiment protocol in [52, 99] for a fair comparison, we perform 12 groups of experiments under the setting of HTL. For each experiment, one of the 4 (sub)datasets is selected as the source, while another dataset is used as the target. We evaluate the performance of EMTLe when all source models are of the same type. As Multi-KT can only leverage knowledge when the source model is SVM, All source models are trained with linear SVMs. The size of each target dataset is varied from 1 to 5 to see how EMTLe and other baselines behave under the extremely small dataset. We use a heuristic way to set the value of λ in Eq. (3.7):

$$\lambda = 2e^{err_n - err_s} \tag{3.8}$$

where err_n and err_s denote the performance of "No transfer" and the source model on the training set. We perform each experiment 10 times and report the average result in Figure 3.4.

Observation & discussion: EMTLe can significantly outperform other baselines especially with a small training set. As we have discussed above, when the training set is small, with the transfer parameter estimated by our ℓ_2 penalty in our high-level objective functions,

EMTLe has a strong generalization ability and performs better on the test data. As the training size increases, the variance of training data decreases and the affect of the ℓ_2 penalty term become less significant. Therefore, EMTLe and the other two HTL baselines show similar performance. It is interesting to see that MKTL even falls into negative transfer even with 5 training examples per class in some experiments. We found that, MKTL is more sensitive to the variance of the training data. Its performance is not as stable as Multi-KT and EMTLe over the 10 experiments. Because MKTL needs to learn more hyperparameters than Multi-KT and EMTLe, even though the training size increases, it may not be able to obtain a good model. In some experiments, we can see that EMTLe can even outperform the Batch method which can access more information and is expected to outperform the other methods under the setting of HTL.

3.4.3 Transfer from Multiple Source Domains

As we mentioned, EMTLe can exploit knowledge from different types of source classifiers which could greatly extend our choice of the source domain under the HTL setting. In this subsection, we show that EMTLe can successfully transfer the knowledge from two different types of source classifiers. Meanwhile, MKTL and "No Transfer" are used as our baseline.

In this experiment, we assume that there is no single source domain that can cover all 13 classes in our target domain and we have to select source models from different source domains. Specifically, the 13 classes are selected from two different domains separately (6 from DSLR and 7 from Webcam) according to Table 3.2. Similar to our previous experiment configurations, we only use Caltech and Amazon as the target domains. We show the experiment results in Figure 3.6.

Table 3.2: The selected classes of the two source domains and the classifier type of the source model.

	class	classifier
DSRL	monitor,bike, helmet,calcu,headphone,projector	Logistic
Webcam	keyboard,mouse,phone,backpack,mug,bottle,laptop	SVMs

Observation & discussion: In our multi-source scenario, it is more difficult to leverage the knowledge from the source models as the models are trained from different domains separately. From the results we can see that, EMTLe can still exploit the knowledge from the source models despite the types of the source classifiers while MKTL can hardly leverage the source knowledge. EMTLe uses a simple way to leverage the source models and BO can help us better

estimate the transfer parameter. However, MKTL uses a sophisticated feature augmentation and has more hyperparameters to estimate. Without sufficient training data, it is difficult for MKTL to measure the importance of each source model and exploit the knowledge from the models.

3.5   Summary

In this chapter, we propose a method, EMTLe that can effectively transfer the knowledge under the HTL setting. We focus on the effectiveness and compatibility issues for HTL problems. We propose our auxiliary bias strategy to let our model exploit the knowledge from different types of source classifiers. The transfer parameter of EMTLe is estimated by bi-level optimization method using our novel high-level objective function which allows our model to better exploit the knowledge from source models. Experiment results demonstrate that EMTLe can effectively transfer the knowledge even though the size of training data is extremely small.

(a) A→D

(b) C→D

(c) W→D

(d) A→W

(e) C→W

(f) D→W
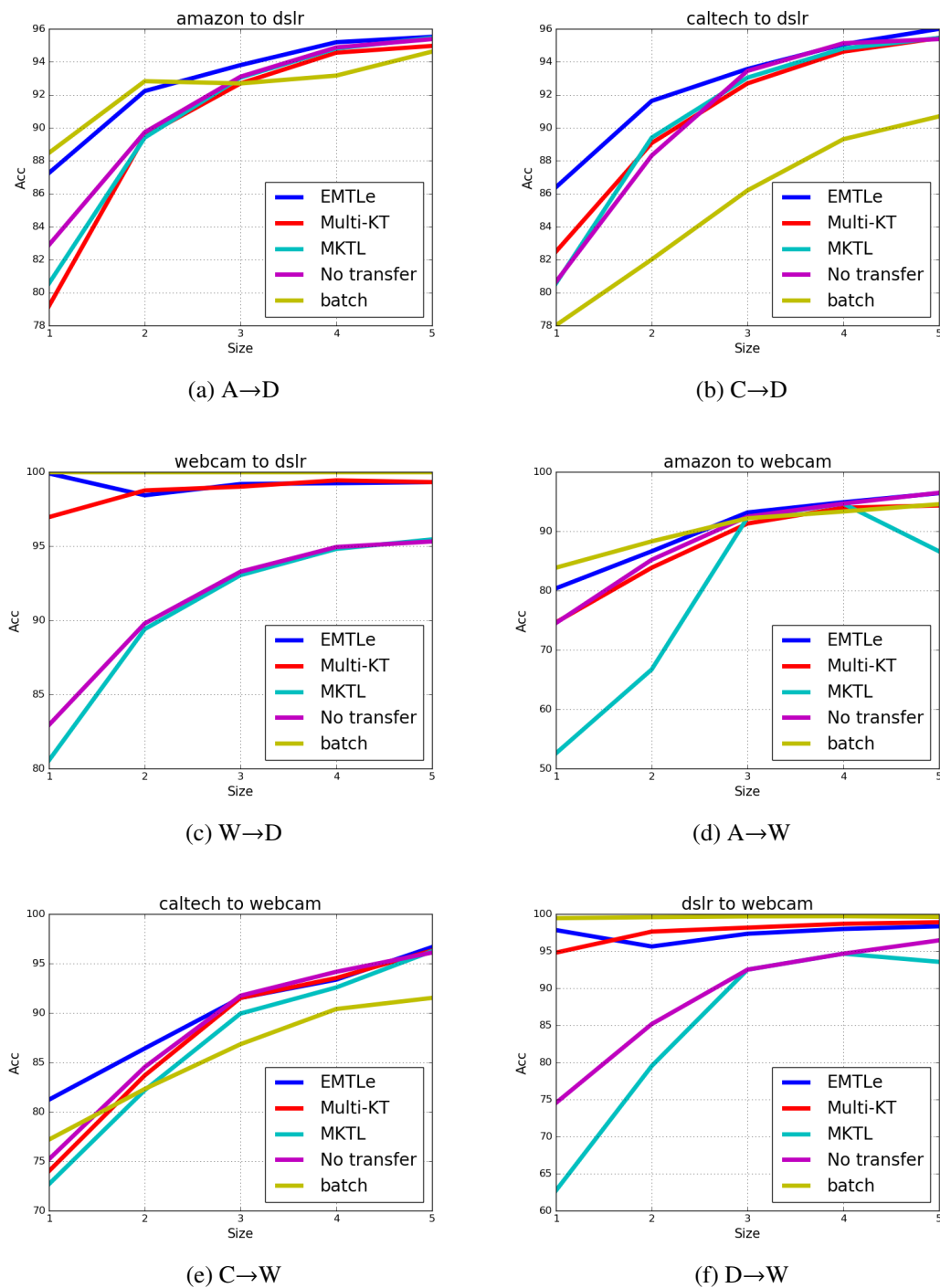
Figure 3.4: Recognition accuracy for HTL domain adaptation from a single source (Part1). 5 different sizes of target training sets are used in each group of experiments. A, D, W and C denote the 4 subsets in Table 3.1 respectively.
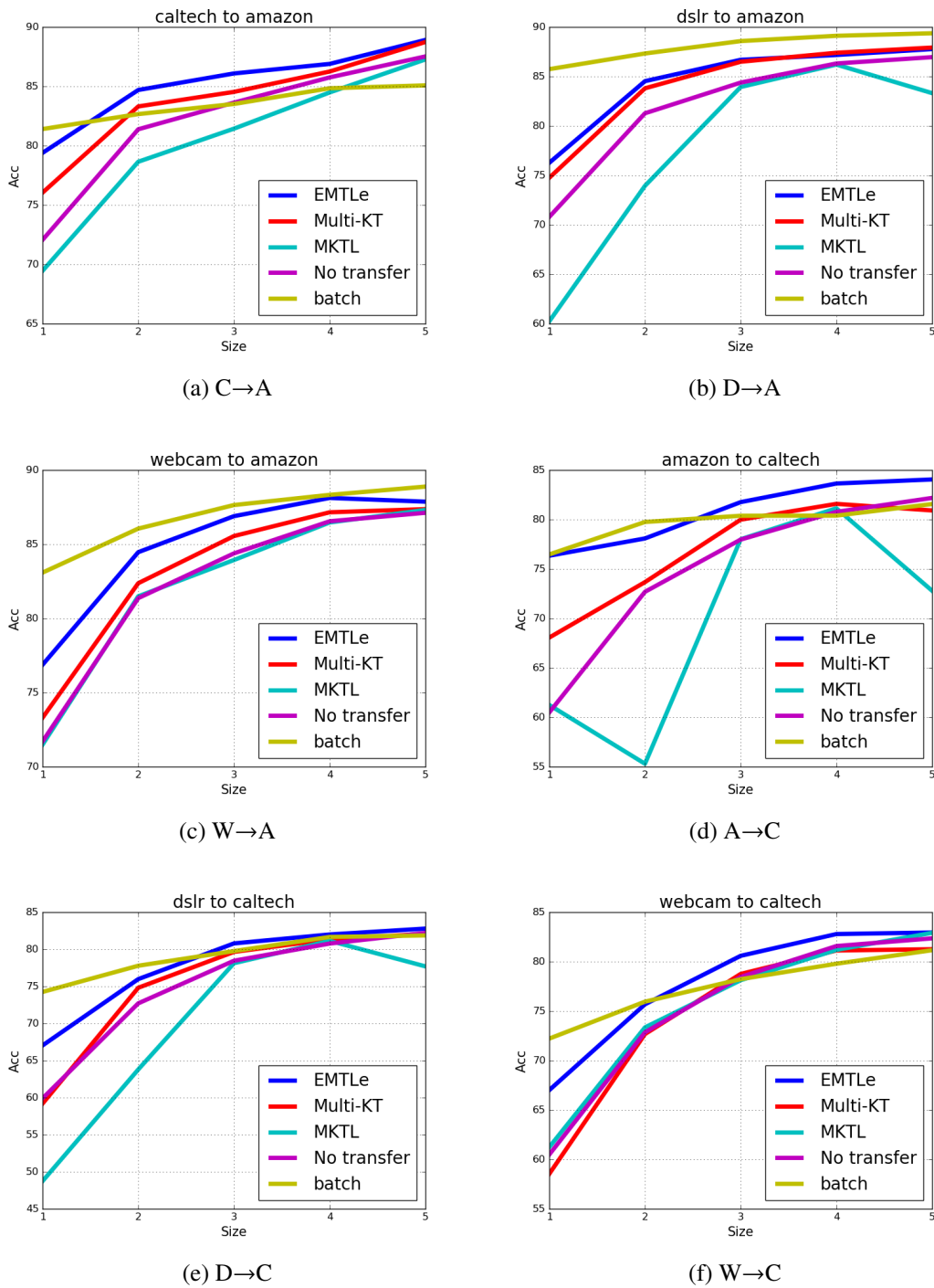
Figure 3.5: Recognition accuracy for HTL domain adaptation from a single source (Part2). 5 different sizes of target training sets are used in each group of experiments. A, D, W and C denote the 4 subsets in Table 3.1 respectively.

(a) D+W → A   (b) D+W → C
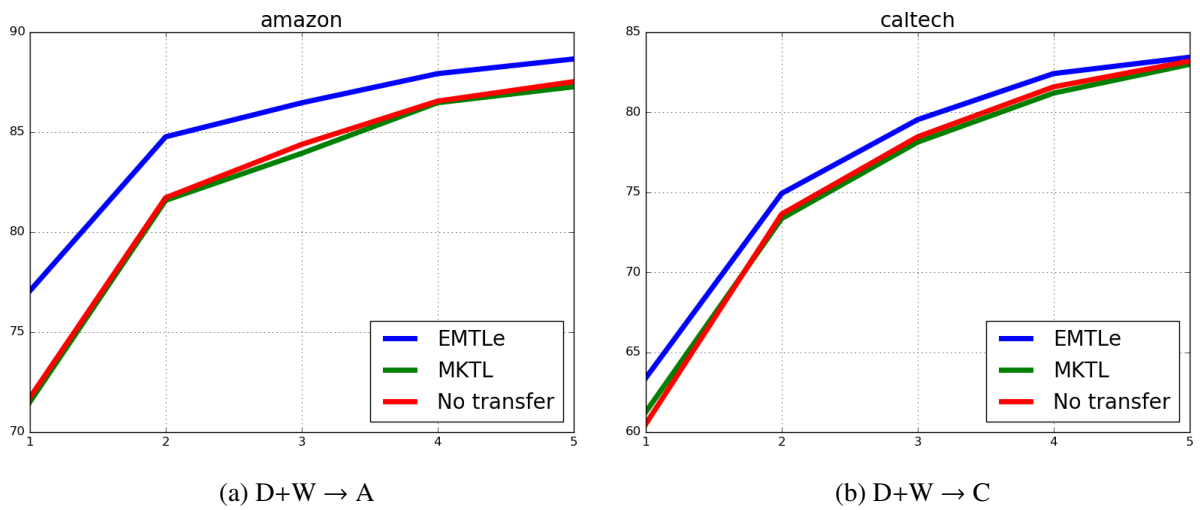
Figure 3.6: Recognition Accuracy for Multi-Model & Multi-Source experiment on two target datasets.

Chapter 4

Fast Generalized Distillation for Semi-supervised Domain Adaptation

4.1 Introduction

Domain adaptation can be used in many real applications, which addresses the problem of learning a target domain with the help of a different but related source domain. In real applications, it can be very expensive to obtain sufficient labeled examples while there are abundant unlabeled ones. *Semi-supervised domain adaptation* (SDA) tries to exploit the knowledge from the source domain and use a certain amount of unlabeled examples and a few labeled ones from the target domain to learn a target model. Typically, the labeled examples in the target domain are too few to construct a good classifier alone. Therefore, an important issue in SDA is how to effectively utilize the unlabeled examples.

The previous work in SDA requires access to the source data to measure the data distribution mismatch between the source and target domain[31, 30, 28, 113]. It is difficult to apply these methods in some real applications where we are not able to access the source data. In this chapter, we try to use the framework of *Generalized Distillation* (**GD**)[69] to solve the SDA problem. We try to explore these two questions: (1) Why GD can solve the SDA problem? (2) How can we improve its effectiveness when we apply GD to real SDA applications?

To answer these two questions, in this chapter, we first propose a new paradigm, called *Generalized Distillation Semi-supervised Domain Adaptation* (**GDSDA**), to solve the SDA problem. We show that, with GDSDA the knowledge of the source models can be effectively transferred to the target domain using the unlabeled data. Then we propose a novel imitation parameter estimation method for GDSDA, called GDSDA-SVM for real SDA problems. Experimental results show that GDSDA-SVM can effectively find the optimal imitation parameter

and achieve competitive performance compared to methods using brute force search but with faster speed.

This work has been published in *Thirtieth-First AAAI Conference on Artificial Intelligence, AAAI 2017*.

4.2 Previous Work

As we use GD to solve SDA problem, we introduce related work in both GD and SDA areas.

In SDA, previous work tried to utilize the unlabeled data to improve the performance. [113] introduced a framework named Semi-supervised Domain Adaptation with Subspace Learning (SDASL) to correct data distribution mismatch and leverage unlabeled data. [30] proposed a framework for adapting classifiers by "borrowing" the source data to the target domain using a combination of available labeled and unlabeled examples. [28] show that augmenting the feature space of the data can compensate the domain shift. [31] proposed a method using the unlabeled data to measure the mismatch between different domains based on the maximum mean discrepancy.

There are also many studies related to GD for computer vision tasks. [91] proposed a Rank Transfer method that uses attributes, annotator rationales, object bounding boxes, and textual descriptions as the privileged information for object recognition. [76] proposed the information bottleneck method with privileged information (IBPI) that leverage the auxiliary information such as supplemental visual features, bounding box annotations and 3D skeleton tracking data to improve visual recognition performance. [103] proposed a CNN architecture for domain adaptation to leverage the knowledge from limited or no labeled data using the soft label. [104] used a small shallow net to mimick the output of a large deep net while using layer-wised distillation with ℓ_2 loss of the outputs of the student and teacher net. Similarly, [72] used ℓ_2 loss to train a compressed student model from the teacher model for face recognition.

Compared to previous work on SDA, our method only requires the output of the source models, which is more effective when the size of the source domain is relatively large and the source model is well-trained. Compared to other work in GD, our method GDSDA-SVM can effectively estimate the imitation parameter while previous work was limited to using either a brute force search or domain knowledge.
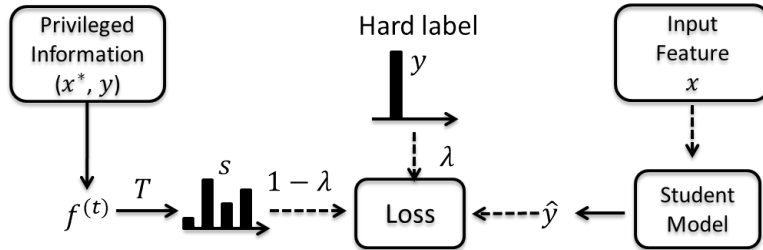
Figure 4.1: Illustration of Generalized Distillation training process.

4.3   Generalized Distillation for Semi-supervised Domain Adaptation

As previously mentioned, GDSDA is a paradigm using generalized distillation for semi-supervised domain adaptation. In this section, we first give a brief review of generalized distillation. Then we show the process of GDSDA and demonstrate the reason why GDSDA can work for the SDA problem. Finally, we show the importance of the imitation parameter.

4.3.1   An Overview of Generalized Distillation and GDSDA

Distillation [43] and *Learning Using Privileged Information* (LUPI) [106] are two paradigms that enable machines to learn from other machines. Both methods address the problem of how to build a student model that can learn from the advanced teacher models. Recently, [69] proposed a framework called *generalized distillation* that unifies both methods and show that it can be applied in many scenarios.

In GD, the training data can be represented as a collection of the triples:

$$\{(x_1, x_1^*, y_1), (x_2, x_2^*, y_2) \dots (x_n, x_n^*, y_n)\}$$

x^* is the privileged information for data x, which is only available in the training set and y is the corresponding label. Therefore, the goal of GD is to train a model, called student model with the guidance of the privileged information to predict the unseen example pair (x, y).

The process of generalized distillation is as follows: in step 1, a teacher model $f^{(t)}$ is trained using the input-output pairs $\{x_i^*, y_i\}_{i=1}^n$. In step 2, use $f^{(t)}$ to generate the soft label s_i for each training example x_i using the softmax function σ:

$$s_i = \sigma(f^{(t)}(x_i)/T) \tag{4.1}$$

where T is a parameter called temperature to control the smoothness of the soft label. In step
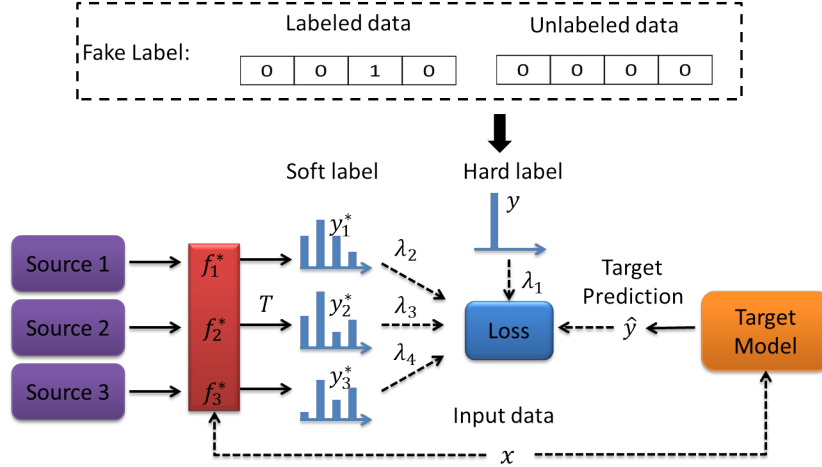
Figure 4.2: Illustration of GDSDA training process and our "fake label" strategy.

3, learn the student $f^{(s)}$ from the pairs $\{(x_i, y_i), (x_i, s_i)\}_{i=1}^n$ using:

$$f^{(s)} = \underset{f^{(s)} \in \mathcal{F}^{(s)}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left[\lambda \ell \left(y_i, \sigma(f^{(s)}(x_i)) \right) + (1 - \lambda) \ell \left(s_i, \sigma(f^{(s)}(x_i)) \right) \right] \qquad (4.2)$$

Here, $\ell(\cdot, \cdot)$ is the loss function and λ is the imitation parameter to balance the importance between the hard label y_i and the soft label s_i.

GD can be used in many scenarios such as multi-task learning, semi-supervised learning, and reinforcement learning. In domain adaptation, when we consider the source model as the teacher and the predictions of the target data given by the source models as the privileged information, GD can be naturally applied to SDA. This leads to *Generalized Distillation Semi-supervised Domain Adaptation* (**GDSDA**). Moreover, in GDSDA, we also consider the multi-source scenario and extend the GD paradigm to fit this scenario. To be consistent with other work of domain adaptation, we use the *source model* and the *target model* to denote the teacher model and the student model.

Technically, when we apply GD to SDA, according to Eq. (4.2), each example is assigned with a hard label *y* (true label) and a soft label *s* (class probabilities from the teacher). However, in SDA, we are not able to obtain the hard labels of the unlabeled data. Here we follow the GD work[69] and use the "fake label" strategy to label the unlabeled data: for the labeled examples, we use *one-hot* strategy to encode their labels while using all 0s as the label of the unlabeled examples (see Fig 4.2). Thus, each example in the target domain is assigned with a label. It is arguable that the "fake label" strategy would introduce extra noise and degrade the performance. However, we will show in our experiment that this noise can be well controlled by setting a proper value to the imitation parameter and we can still achieve improved performance (See the single source experiment).

Suppose we have $M - 1$ source domains denoted as $D_s^{(j)} = \{X^{(j)}, Y^{(j)}\}_{j=1}^{M-1}$ and the target domain $D_t = \{X, Y\}$ encoded with the "fake label" strategy. The process of GDSDA is as follows:

1. Train the source models f_j^* for each of the $M - 1$ domains with $\{X^{(j)}, Y^{(j)}\}$.

2. For each of the training example x_i in the target domain, generate the corresponding soft label y_{ij}^* with each of the source model f_j^* and the temperature $T > 0$.

3. Learn the target model f_t using the $(M + 1)$-tuples $\{x_i, y_i, y_{i1}^*, \ldots, y_{i(M-1)}^*\}_{i=1}^L$ with the imitation parameters $\{\lambda_i\}_{i=1}^M$ using (4.3):

$$f_t(\lambda) = \arg\min_{f_t \in \mathcal{F}} \frac{1}{L} \sum_{i=1}^L \left[\lambda_1 \ell\left(y_i, f_t(x_i)\right) + \sum_{j=1}^{M-1} \lambda_{j+1} \ell\left(y_{ij}^*, f_t(x_i)\right) \right]$$

$$\text{s.t.} \quad \sum_i \lambda_i = 1 \tag{4.3}$$

In Figure 4.3 we show an example of how to use GDSDA to train the target model when there is one source model available. The distilled label consists two parts: the label information from the source model and from the ground truth. For each part, we use the imitation parameters to reweigh them. As long as we can find the good imitation parameters, we can successfully label all data and therefore, can build a target model from the target data.

Compared to other studies on SDA where each example of the source domain was used, by either re-weighting [30, 33] or augmentation [28], GDSDA only requires the trained model from the source domain to generate the soft labels. Considering that it is more convenient to access the source model than each of the examples of the source domain, GDSDA can be more useful than those previous methods. For example, if we want to use ImageNet [29] as the source domain, it is almost impossible to access each of the millions of examples while there are many well trained models publicly available online that can be used for GDSDA. Also, GDSDA is able to handle the multi-class scenario while previous methods, such as SHFA[32] only solved the binary classification problem of SDA. Moreover, GDSDA is compatible with any type of source model that is able to output the soft label (i.e., the class probabilities).

4.3.2   Why does GDSDA work

In this section, we demonstrate the scenarios where GDSDA can work. Before we provide our analysis, we first introduce two basic assumptions for GDSDA: the *assumption of distillation* and the *assumption of the source model*.

Assumption of Distillation: The capacity (or VC dimension) of the target model f_t is smaller than the capacity of source model f^*. This assumption is inherited from distillation
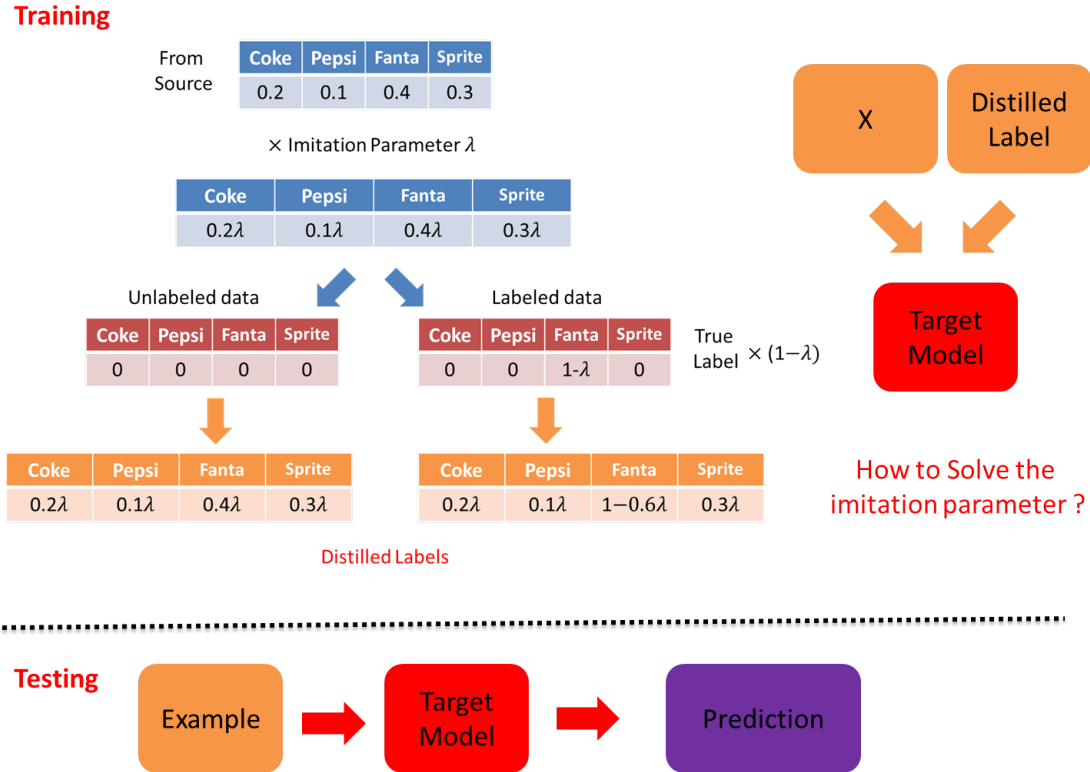
Figure 4.3: An example of using GDSDA to generate distilled labels for the target data.

[69]. **Assumption of the source model:** The source model f^* should work better than a target model f'_t trained only with the hard labels. This assumption is common, especially in SDA where the labeled examples are often too few to build a good target model. For example, when we only have one labeled example from each class in the target training set, it is reasonable to assume that the source model trained from another domain can perform better than the model trained only with the target training data on the target task. Based on these two assumptions, we will show that GDSDA can effectively leverage the source model and transfer the knowledge between different domains under the SDA setting.

In GDSDA, we actually have two steps to get improved performance through knowledge transfer: *Unsupervised distillation* and *Label fusion with target label*.

- **Distillation with unlabeled data**. According to the ERM principle[107], a simple model has better generalization ability than the complex one, if they both have the same training error. As long as the target model f_t can achieve similar training error to that of the source model f^* on the target domain, considering the fact that the VC dimension of f_t is smaller than f^*, we can expect that the target model has better generalization ability. This process can be achieved by letting the target model mimic the output of the source model on the training data. Here, we call the output of the source model "**soft label**". It

is worthy to notice that in this process, the target model only has to mimick the soft label without considering the true labels of the examples. In another word, GDSDA provide an effective way to utilize the unlabeled data.

- **Label fusion with target label**. Arguably, because of the domain shift, the source model is biased towards the source domain when we apply it to the target task. However, as suggested in [43], we can use labeled information from the target domain to compensate for the domain shift and achieve a better performance on the target task with Eq. (4.3). Specifically, we use the imitation parameter λ to control the relative importance between the soft label and the hard label, which in turn reflects the similarity between the source and target tasks. For example, in Figure 4.3, when we set $\lambda_1 = 0$, we actually ignore the knowledge from source model and thus can avoid negative transfer if we notice the source knowledge may hurt the transfer process. As a result, GDSDA can compensate for the domain shift under the setting of SDA (for more details, please see the experiment section).

4.3.3 Key Parameter: the Imitation Parameter

From the analysis above, we can see that in each of the step, the target model can get improved performance through a simple but effective method. However, in the label fusion part, an important issue is to determine the value(s) of the imitation parameter. As we show before, the imitation parameter balance the importance of the source and target knowledge which is essential in GDSDA. In this section, we demonstrate that the imitation parameter can greatly affect the performance of the target model.

In GDSDA, we must decide the values of 2 parameters, the temperature T and the imitation parameter λ. The temperature T controls the smoothness of the soft label and the imitation parameter λ controls how much knowledge can be transferred from the source model. Previous work has addressed the importance of knowledge control in domain adaptation [32, 33]. Without carefully controlling the amount of knowledge transferred from the source domain, the target model may suffer from degraded performance or even negative transfer [79]. How to choose the imitation parameter is crucial for GDSDA. In previous work, however, the imitation parameter was determined by either a brute force search [69] or background knowledge [103]. Meanwhile, in real applications, it is common that multiple source domains can be exploited. As suggested by [99], learning from multiple related sources simultaneously can significantly improve the performance of the target model. However, these previous methods become more difficult to apply when there are multiple sources and imitation parameters to be determined. For these reasons, it is ideal to find an approach that can determine the imitation parameter

automatically.

4.4 GDSDA-SVM

As previously mentioned, it is important to find an approach that can effectively determine the imitation parameter. In this section, we propose our method GDSDA-SVM which uses SVM as the base classifier and can effectively estimate the imitation parameter by minimizing the cross-validation error on the target domain.

4.4.1 Distillation with Multiple Sources

As suggested in [106], the optimal imitation parameter should be the one that can minimize the training error on the target domain. Based on that, we propose our method GDSDA-SVM which can effectively estimate the imitation parameter.

Instead of using hinge loss in our GDSDA-SVM, we use Mean Squared Error (MSE) as our loss function for the following two reasons: (1) several recently studies [5, 72, 84, 104] show that MSE is also an efficient measurement for the target model to mimick the output of the source model. (2) MSE can provide a closed form cross-validation error estimation which allows us to estimate the imitation parameter effectively.

Suppose we have L examples $\{\mathbf{x}_j, \mathbf{y}_j\}_{j=1}^{L}$ from N classes in the target domain where $X \in R^{L \times d}, Y \in R^{L \times N}$. Meanwhile, there are $M - 1$ source (teacher) models providing the soft labels $Y^* = \{\mathbf{y}_{ij}^* | j = 1, ..., L; i = 1, ..., M-1\}$ for each of the L examples. For simplicity, we concatenate the hard label Y and soft label Y^* into a new label matrix S, where:

$$S = [Y; Y^*] = [S_1; ...; S_M]; S_i \in R^{L \times N}$$

To solve this N-class classification problem, we adopt the One-vs-All strategy to build N binary SVMs. To build the nth binary SVM, we have to solve the following optimization problem:

$$\min_{w_n} \quad \frac{1}{2}\|\mathbf{w}_n\|^2 + C \sum_j e_{jn}^2$$

$$s.t. \quad e_{jn} = \sum_i \lambda_i S_{ijn} - \mathbf{w}_n \mathbf{x}_j \tag{4.4}$$

We use the KKT theorem [24] and add dual sets of variables to the Lagrangian of the optimization problem:

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}_n\|^2 + C \sum_j e_{jn}^2$$

$$+ \sum_j \eta_{jn} \left(\sum_i \lambda_i S_{ijn} - \mathbf{w}_n \mathbf{x}_j - e_{jn} \right) \tag{4.5}$$

To find the saddle point,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_n} = 0 \rightarrow \mathbf{w}_n = \sum_j \eta_{jn}\mathbf{x}_j; \quad \frac{\partial \mathcal{L}}{\partial e_{jn}} = 0 \rightarrow \eta_{jn} = 2Ce_{jn} \tag{4.6}$$

For each example $\mathbf{x}_j$ and its constraint of label S_{ijn}, we have $e_{jn} + \mathbf{w}_n\mathbf{x}_j = \sum_i \lambda_i S_{ijn}$. Replacing $\mathbf{w}_n$ and e_{jn}, we have:

$$\sum_j \eta_{jn}\mathbf{x}_j\mathbf{x}_i + \frac{\eta_{in}}{2C} = \sum_i \lambda_i S_{ijn} \tag{4.7}$$

Here we use Ω to denote the matrix $\Omega = [K + \frac{\mathbf{I}}{2C}]$ where K is the linear kernel matrix $K = \{\mathbf{x}_i\mathbf{x}_j | i, j \in 1\ldots L\}$. Let Ω^{-1} be the inverse of matrix Ω and Ω_{jj}^{-1} be the jth diagonal element of Ω^{-1}. We have $\eta = \sum_i \lambda_i \Omega^{-1} S_i = \sum_i \lambda_i \eta'_i$. According to [18], the Leave-one-out (LOO) estimation of the example $\mathbf{x}_j$ for the nth binary SVM can be written as:

$$\hat{y}_{jn} = \sum_i \lambda_i \left(S_{ijn} - \frac{\eta'_{ijn}}{\Omega_{jj}^{-1}} \right) \tag{4.8}$$

Now for any given λ, we have found an efficient way to estimate the LOO prediction of each binary target model for example $\mathbf{x}_j$. In the following section, we will introduce how to find the optimal λ_i for each of the source models.

4.4.2   Cross-entropy Loss for Imitation Parameter Estimation

From the previous section, we have already found an effective solution to estimate the output of the SVM. The optimal imitation parameters can be found by solving the following optimization problem[1]:

$$\min \quad L_c(\lambda) = \frac{1}{2}\sum_i^M \|\lambda_i\|^2 + \frac{1}{L}\sum_{j,n} \ell\left(y_{in}, \hat{y}_{jn}(\lambda)\right) \tag{4.9}$$

$$s.t. \quad \sum \lambda_i = 1$$

Here we use the ℓ-2 regularization term to control the complexity of λs so that the target model can achieve better generalization performance. For the loss function $\ell(\cdot, \cdot)$, We choose the cross-entropy loss function.

$$\ell\left(y_{in}, \hat{y}_{jn}(\lambda)\right) = y_{in} \log(P_{jn}) \qquad P_{jn} = \frac{e^{\hat{y}_{jn}}}{\sum_h e^{\hat{y}_{jh}}} \tag{4.10}$$

Cross-entropy pays less attention to a single incorrect prediction which reduces the affect of the outliers in the training data. Moreover, cross-entropy works better for the unlabeled data with

---

[1]In Eq.(4.9), it is reasonable to use a hyperparameter β to regularize λ_i, i.e. $\frac{\beta}{2}\sum_i^M \|\lambda_i\|^2$. Empirically, we found that $\beta = 1$ can achieve good results in our experiment and ignore this hyperparameter β here.

Algorithm 2 GDSDA-SVM

Input: Input examples $X = \{\mathbf{x}_1, ..., \mathbf{x}_L\}$, number of classes N, number of sources M, 3D label
matrix, $S = [Y_1, Y_2, ..., Y_M]$ with size $L \times M \times N$, temperature T

Output: Target model $f_t = Wx$

$\quad \Omega = [K + \frac{\mathbf{I}}{2C}]$

$\quad$ Find the imitation parameter λ with Algorithm 3

$\quad$ Generate new label $Y_{new} = \sum_i \lambda_i S_i$

$\quad$ Calculate $\eta = \Omega^{-1} Y_{new}$

$\quad$ Calculate $w_n = \sum_j \eta_{jn} x_j$

---

our "fake label" strategy. As we mentioned in our "fake label" strategy, we use 0s to encode
the hard labels of the unlabeled examples. From (4.10) we can see that cross-entropy loss can
automatically ignore penalties of the unlabeled examples and reduce the noise introduced by
our "fake label" strategy. Let:

$$\mu_{ijn} := S_{ijn} - \frac{\eta'_{ijn}}{\Omega^{-1}_{jj}} \tag{4.11}$$

The derivative can be written as:

$$\frac{\partial \ell(\lambda)}{\partial \lambda_i} = \sum_n \mu_{ijn} \left(P_{jn} - y_{jn} \right) \tag{4.12}$$

We describe GDSDA-SVM in Algorithm 2. As the optimization problem (4.9) is strongly
convex, it is easy to prove that Algorithm 3 can converge to the optimal λ with the rate of
$O(\log(t)/t)$ where t is the optimization iteration (see Appendix A.2).

4.5 Experiments

In this section, we show the empirical performance of our algorithm GDSDA-SVM on the
Office benchmark dataset. Specifically, we provide the empirical results under two transfer
scenarios: single source and multi-source transfer scenarios for GDSDA-SVM.

Dataset: We use the domain adaptation benchmark dataset Office as our experiment dataset.
There are 3 subsets in Office dataset, Webcam (795 examples), Amazon (2817 examples) and
DSLR (498 examples), sharing 31 classes. We denote them as W, A and D respectively. In
our experiments, we use DSLR and Webcam as the source domains and Amazon as the target
domain. We use the features extracted from Alexnet [58] FC7 as the input feature for both
source and target domain. The source models are trained with multi-layer perception (MLP)
on the whole source dataset.

Algorithm 3 λ Optimization

Input: Input examples X, number of classes N, size of sources M, 3D label matrix S, temperature T, optimization iteration $iter$, Kernel matrix Ω

Output: Imitation parameter λ

    Initialize $\lambda = \frac{1}{M}$,

    Let S_n be the label matrix of S for class n

    for Each label S_n **do**

        Calculate $\eta'_n = \Omega^{-1} S_n$

    end for

    Calculate μ using (4.11)

    for $it \in \{1, ..., iter\}$ **do**

        Compute $\hat{y}_{jn}$ and P_{jn} with (4.8) and (4.10)

        $\Delta_\lambda \leftarrow 0$

        for each $\mathbf{x}_j$ in X **do**

            $\Delta_\lambda = \Delta_\lambda + \sum_n \mu_{ijn} \left(P_{jn} - y_{jn} \right)$

        end for

        $\Delta_\lambda = \Delta_\lambda / L$, $\lambda = \lambda - \frac{1}{it}(\Delta_\lambda + \lambda)$

        $\lambda = \lambda / \sum \lambda_i$

    end for

4.5.1   Single Source for Office datasets

In this experiment, we compare our algorithm under the scenario where the source model is trained from a single source dataset. Specifically, we have two groups of experiments, transferring from Webcam to Amazon and from DSLR to Amazon. As we mentioned, there are significantly fewer labeled examples than unlabeled ones in real SDA applications. Therefore, in each group of experiment, there are only 31 labeled examples (1 per class) and some unlabeled examples (10, 15 and 20 per class) in the target domain.

To demonstrate the effectiveness of GDSDA-SVM, we show the performance of GDSDA using brute force to search the imitation parameter as the baseline. As there are two imitation parameters in this experiment, we use λ_1 and $1 - \lambda_1$ to denote the imitation parameter for hard and soft label respectively. Specifically, we search the imitation parameter λ_1 in the range $[0, 0.1, ..., 1]$ with different temperature T. Meanwhile, we show the performance of the source model (denoted as "Source") and the performance of a target model (denoted as "No transfer" using LIBLINEAR[36]) trained with only labeled examples of the target domain on the target task. We run each experiment 10 times and report the average result. For GDSDA-SVM, as

we are not able to tune the temperature T, we empirically set $T = 20$ for all experiments in this subsection. The experimental results are shown in Figure 4.5.

From the results of the brute force search we can see that, the value of imitation parameter can greatly affect the performance of the target model. As we expected, without using any true label information of the target data, i.e. $\lambda_1 = 0$, GDSDA can still slightly outperform the source model. This means GDSDA can effectively transfer the knowledge between different domains with the unlabeled data. As we increase the value of imitation parameter, i.e. considering the hard labels from the target domain, the performance of GDSDA can be further improved. As we mentioned before, even though our "fake label" strategy would introduce extra noise, the noise can be limited by setting a proper value to imitation parameter and the target model can still achieve improved performance compared to the baselines.
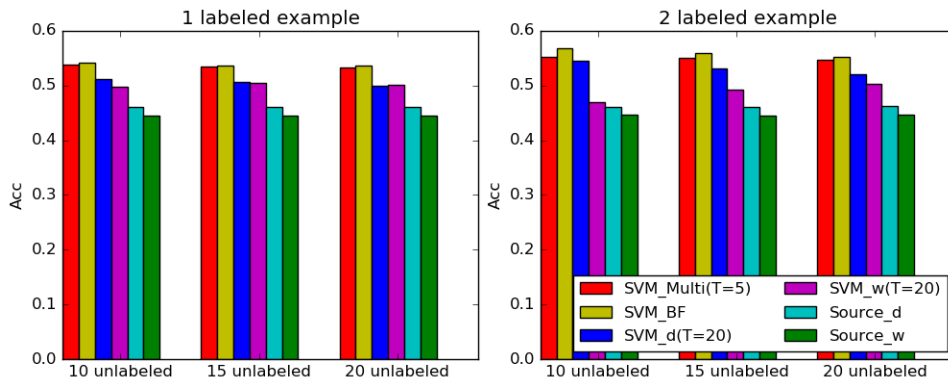
Figure 4.4: D+W→A, Multi-source results comparison.

Moreover, we can see that GDSDA-SVM can achieve competitive results compared to baselines using brute force search in D→A experiments. In W→A experiments, it achieves the best performances on all 3 different unlabeled sizes. This indicates that we can efficiently (about 6 times faster than the brute force search) obtain a good target model with GDSDA-SVM.

4.5.2 Multi-Source for Office datasets

In this experiment, we show the performance of GDSDA-SVM under the multi-source SDA scenario. Specifically, we use Amazon as the target domain which can leverage the knowledge of two source models trained from Webcam and DSLR. We use the similar settings as our single source experiment and perform 2 groups of experiments using 1 labeled and 2 labeled examples per class respectively. We use temperature $T = 5$. The results of multi-source GDSDA-SVM are denoted as SVM_Multi. Here we also include two single source GDSDA-

SVMs obtained from the experiments above (SVM_w and SVM_d trained using Webcam and DSLR as the source respectively) as the baselines. Moreover, we show the best performance of the brute force search model (SVM_BF). For SVM_BF, we search temperature in range $T = [1, 2, 5, 10, 20, 50]$ and each imitation parameter in range $[0, 0.1, ..., 1]$. The experiment results are shown in Figure 4.4.

From the results, we can see that, given 2 source models, SVM_Multi can outperform any single source model trained with GDSDA. This indicates GDSDA-SVM can still exploit the knowledge even in the complex multi-source scenario. Even though SVM_Multi performs slightly worse than the best result found by brute force search in some experiments, considering their time consumption (GDSDA-SVM is around 30 times faster than brute force search), SVM_Multi still has its advantage in real applications.

4.6 Summary

In this chapter, we propose a novel framework called *Generalized Distillation Semi-supervised Domain Adaptation* (GDSDA) that can effectively leverage the knowledge from the source domain for SDA problem without accessing to the source data. To make GDSDA more effective in real applications, we proposed our method called GDSDA-SVM and show that GDSDA-SVM can effectively determine the imitation parameter for GDSDA. In our future work, we plan to use a more advanced hyperparameter optimization method, which can optimize the imitation parameter λ and the temperature T in GDSDA simultaneously and expect further performance improvement.

(a) D → A, 10 unlabeled

(b) D → A, 15 unlabeled

(c) D → A, 20 unlabeled

(d) W → A, 10 unlabeled

(e) W → A, 15 unlabeled

(f) W → A, 20 unlabeled
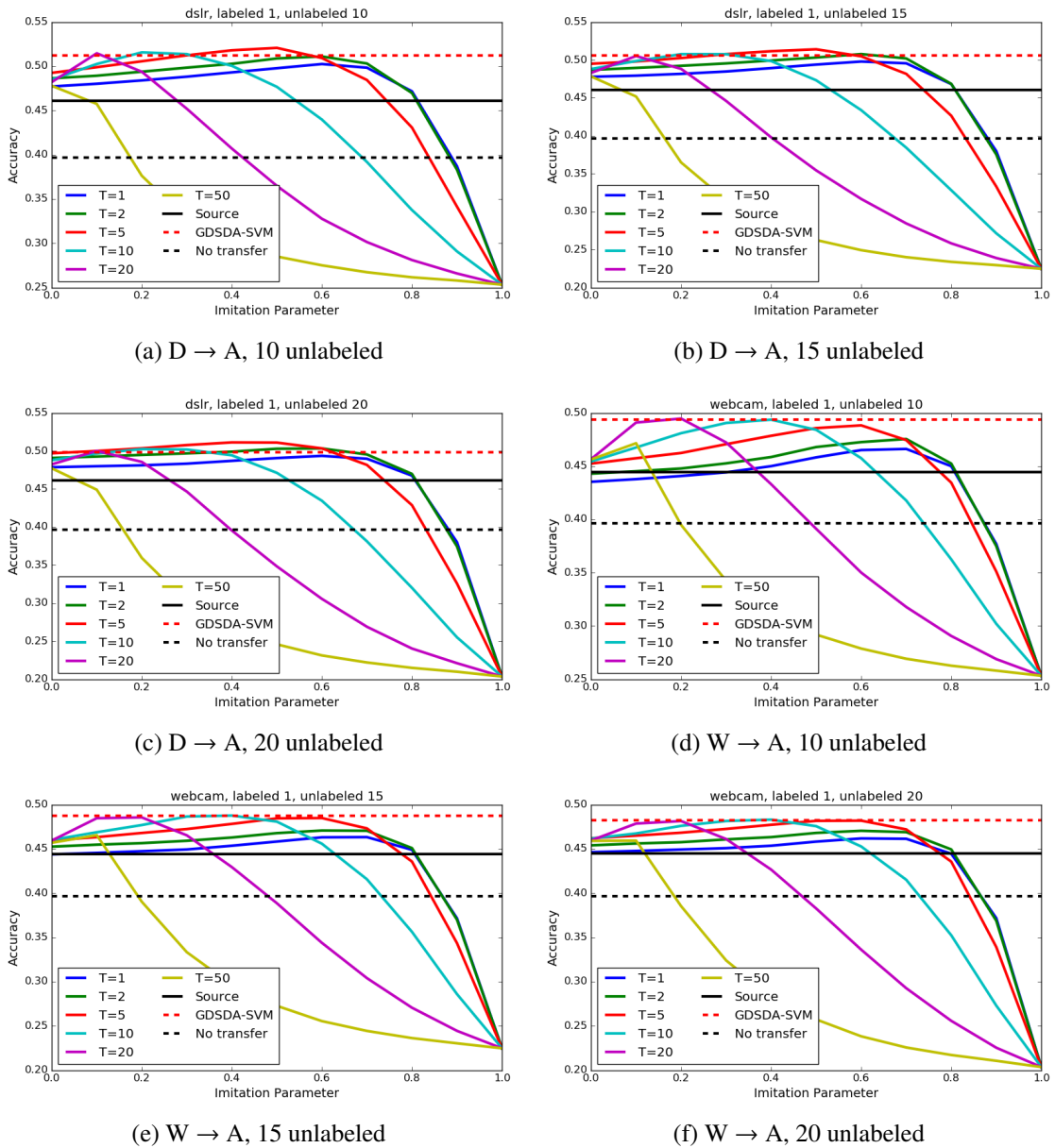
Figure 4.5: Experiment results on DSLR→Amazon and Webcam→Amazon when there are just one labeled examples per class. The X-axis denotes the imitation parameter of the hard label (i.e. λ_1 in Fig 4.2) and the corresponding imitation parameter of the soft label is set to $1 - \lambda_1$.

Chapter 5

Learning Food Recognition Model with Deep Representation

5.1 Introduction

Deep Learning with Convolutional Neural Networks (CNNs) is the most popular method for image recognition and has been applied to solve many real problems. Different than the two methods we proposed, which use the output of the source model for knowledge transfer, in this chapter, we use the parameters of Deep CNNs as transferable knowledge and use it to initialize the Deep CNNs for our recognition task.

In this chapter, we investigate the problem of using the pre-trained CNNs for transfer learning. In particular, we fine-tune the parameters of the pre-trained CNNs on two food image database and achieve the improved results. We also investigate the changes of the parameters after fine-tuning and try to obtain some important experience on fine-tuning the deep CNNs.

This work has been published by *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*.

5.2 Tuning the Deep CNNs

The basic fact that fine-tuning can work for transfer learning is that Deep CNN can learn hierarchical features from bottom to top and some of the features are task-independent, especially most low-level and some mid-level ones.

There are two major strategies to fine-tune the deep CNNs: fine-tune the last layer (i.e. the classifier layer) and fine-tune the whole net.

- **Fine-tune the last layer.** In the first strategy, we consider the deep CNNs as a fixed

64
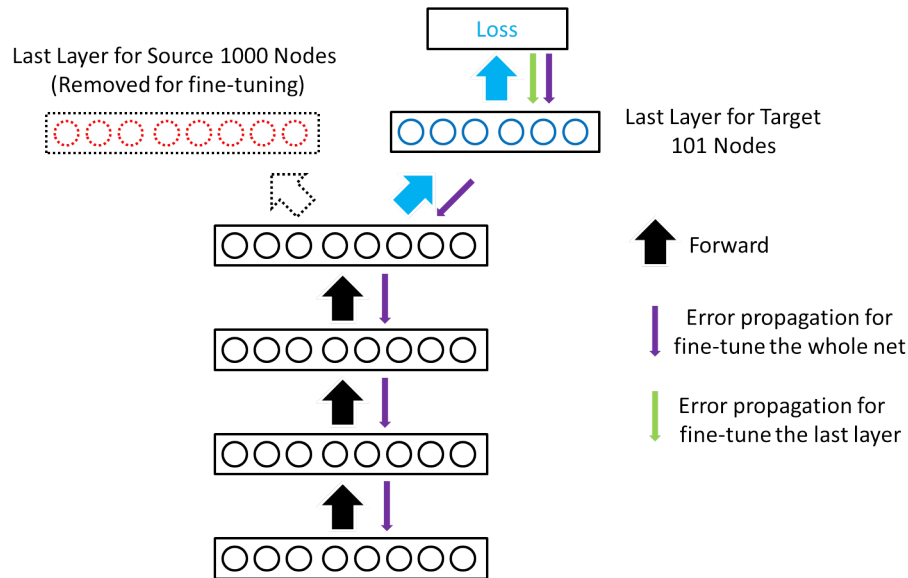
Figure 5.1: Demonstration of Fine-tuning from ImageNet 1000 classes to Food-101 datasets.

feature extractor and only modify the last layer of the net. This assumes that the features extracted by the pre-trained deep CNN can be directly applied to the new tasks. Therefore, by changing the last layer of the model, e.g. the combination of the features, the model can adopt new classes effectively. For example, when we introduce the mythical creature Sphinx to someone, we usually say it has the head of a human, the haunches of a lion, and the wings of a bird and people could obtain a rough idea of how Sphinx looks like without even see its picture. Here, the features such as head and haunches to describe Sphinx are high-level features that can be adopted directly from learned knowledge. Without learning from scratch, we can adopt the new concept effectively.

- **Fine-tune the whole net.** When fine-tuning the whole net, typically, we assume that some of the features extracted from the pre-trained deep CNN are not good enough for the new tasks. Therefore, we have to learn some new high-level features again. In fact, the high-level features are task-dependent and can be learned from those task-independent low-level features with certain combination. The purpose of fine-tuning the whole net is to changing the combination of the low-level features to reconstruct the task-dependent high-level features for the new task.

For both strategies, the last layer, i.e. the layer of a classifier is removed. For different fine-tuning strategies, the error would back propagate to different layers (see Figure 5.1). Typically, a small learning rate is used to for fine-tuning the model to make sure that the model will not overfit the data of the target task.

5.3   Layers in Deep CNN

A CNN consists of some convolutional and subsampling layers optionally followed by fully connected layers. In this part, we introduce the layers used in our work.

5.3.1   Convolutional Layer

Convolutional Layer is the core building block of a Convolutional Network, and its output volume can be interpreted as holding neurons arranged in a 3D volume. Natural images have the property of being "stationary" meaning that the statistics of one part of the image are the same as any other part. This suggests that the features that we learn at one part of the image can also be applied to other parts of the image, and we can use the same features at all locations. For example, some low-level features such as some special points or edges exist in many objects and can be used as the features for different types of objects.

Formally, given some original $h \times w$ input images I, we can train a small autoencoder from $a \times b$ kernel matrix. Also, we have to set other hyperparameters, stride s and padding p. Stride defines the number of pixels the kernel should be moved in each step around the image I and padding defines the number of rows/columns padded to the height and width of the original input (see Figure 5.2). Given a $a \times b$ kernel matrix $W^{(1)}$, bias $b^{(1)}$, padding p and stride s, we can encode the original image I as $f_{conv} = sigmod(W^{(1)}I_p + b^{(1)})$ for $I_p \in I$, giving us f_{conv} (called feature map of $W^{(1)}$), a $\left\lceil \frac{(h-a+2p)}{s} + 1 \right\rceil \times \left\lceil \frac{(w-b+2p)}{s} + 1 \right\rceil$ array of feature map. In general, for any specific input I ($h \times w \times c$ array matrix) of a convolutional layer L, assuming we have k such $a \times b \times c$ kernel matrix, its feature maps f should be a $\left\lceil \frac{(h-a+2p)}{s} + 1 \right\rceil \times \left\lceil \frac{(w-b+2p)}{s} + 1 \right\rceil \times k$ array matrix with $f_{conv}^{(i)} = sigmod(W^{(i)}I_p + b^{(i)})$ for $I_p \in I$ and $i \in 1, \ldots k$.

In real applications, small kernels (3×3, 5×5 and 7×7) are preferred by many different CNN architectures [58] [65] [92] [115]. Recent development of tiny 1×1 kernel shows an improvement on both accuracy and computational efficiency [96].

5.3.2   Pooling Layer

Pooling layer is widely used in all kinds of CNN architecture for dimensional reduction and computational efficiency. After obtaining features maps using convolutional layer, we need to use them for classification. However, applying the feature maps from the convolutional layer for classification would be a computational challenge. Consider for instance images of size 96×96 pixels, and suppose we have learned 400 features over 8×8 inputs. Each convolution results in an output of size $(96 - 8 + 1) \times (96 - 8 + 1) = 7921$, and since we have 400 features,

$$1\times1+2\times1+2\times1+0\times0+3\times0+5\times0+2\times1+5\times1+1\times1=13$$
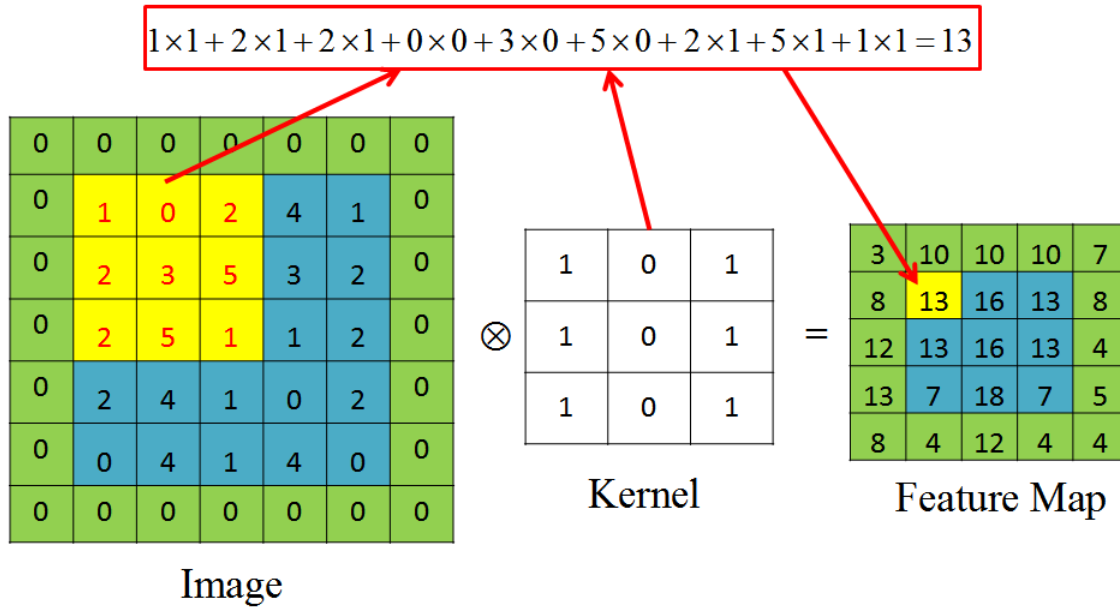
Figure 5.2: Convolution operation with 3×3 kernel, stride 1 and padding 1. $\otimes$ denotes the convolutional operator.

this results in a vector of $892 \times 400 = 3,168,400$ features per example. Learning a classifier from over 3 million features could lead to severe over-fitting.

Therefore, it is common to periodically insert a (Max) Pooling layer in-between successive convolutional layers in CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The pooling layer works independently on the channel dimension and resizes the feature map spatially. For certain $h \times w \times c$ input array matrix, a $a \times b$ Pooling layer with stride s and p padding would output a $\left\lceil \frac{(h-a+2p)}{s} + 1 \right\rceil \times \left\lceil \frac{(w-b+2p)}{s} + 1 \right\rceil \times c$ matrix array.

In general, two kinds of pooling strategy, Max Pooling and Average Pooling, are commonly used in CNN architecture (see Figure 5.3). Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice [74] [96]. Max Pooling is been widely used in all kinds CNN architectures [16] [110].

5.3.3 Fully Connected Layer

Fully Connected (FC) Layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Recent work show that FC layers with Rectified Linear Units and Dropout can greatly improve the learning speed as well as avoid overfitting for deep CNNs
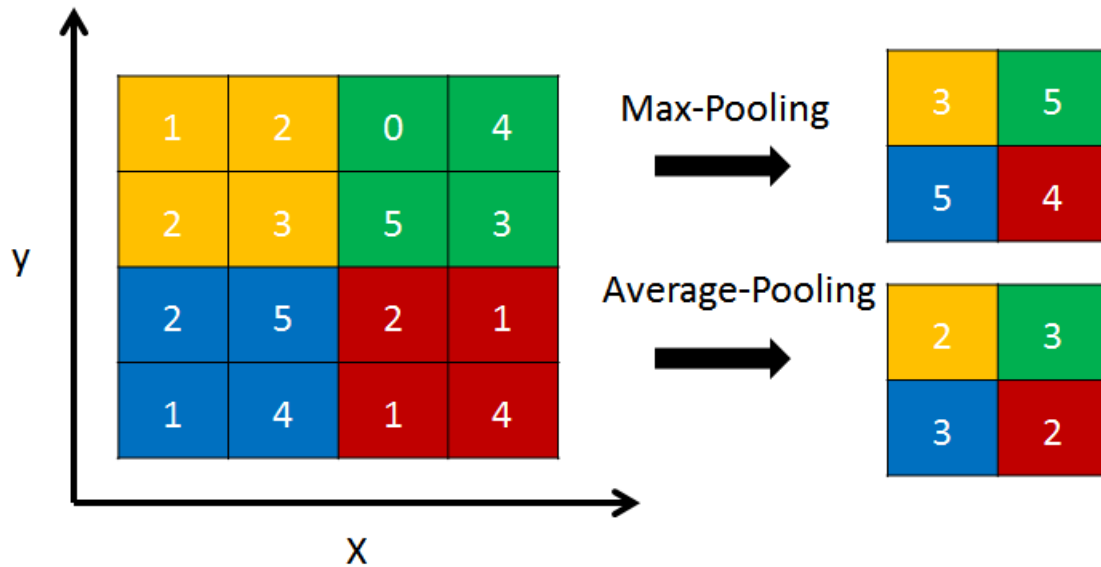
Figure 5.3: 2×2 pooling layer with stride 2 and padding 0.

[44] [77].

5.3.3.1   Rectified Linear Units (ReLUs) for Activation

Rectified Linear Units can be considered as replacing each binary unit with sigmoid activation by an infinite number of copies that all have the same weights but have progressively more negative biases. This replacing procedure can be mathematically presented as:

$$\sum_{i}^{N} \sigma(x - i + 0.5) \approx \log(1 + e^x) \tag{5.1}$$

where $\sigma(x)$ is the sigmoid function. In practice, Rectified Linear Units use the function

$$f(x) = \max(x, 0) \approx \log(1 + e^x) \tag{5.2}$$

 as the activation function for approximation [49]. With *max* function, the derivatives of the active ($x > 0$) and inactive neurons are 1 and 0 respectively. As a result, ReLUs can speed up the learning procedure greatly and improve the performance.

5.3.3.2   DropOut

In FC layer, nodes are connected to each other and this leads to a large number of parameters. Generally, larger number of parameters means more power for Neural Networks and more easily prone to overfitting. Dropout is a technique for addressing this problem. The key idea is

(a) Standard Neural Net   (b) After Dropout
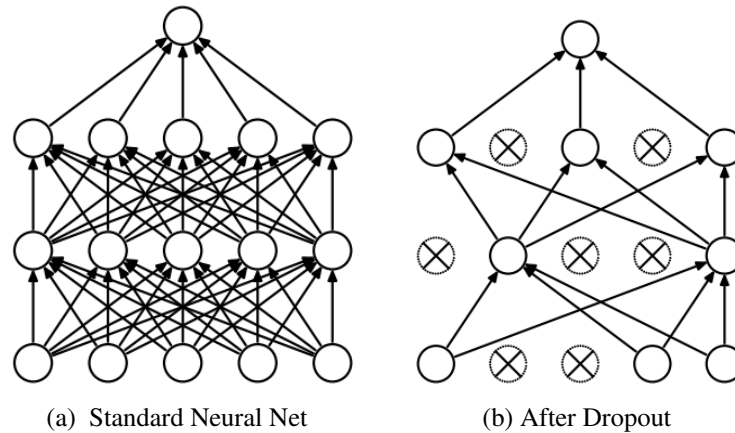
Figure 5.4: Dropout Layers. Adopted from Standford CS231n Convolutional Neural Networks for Visual Recognition

to randomly drop units (along with their connections) from the neural network during training [94]. Technically, dropout can be interpreted as adding extra noise into the training procedure. Without actually adding noise, FC layer with dropout is tolerant of higher level of noise (20 %-50%). Randomly dropping out the nodes, for any node in FC layer, it can't rely on the other nodes to adjust its result. By eliminating the co-adaptation of hidden units, dropout becomes a technique that can be applied to any general domain and improve the performance of neural nets.

5.4 Experiment Settings

In this section, we introduce some details about the two architectures and the food datasets used in our experiments.

5.4.1 Models

In this paper, AlexNet and GoogLeNet are their Caffe [50] implementations and all the results for a specific CNN architecture are obtained from a single model.

AlexNet contains 5 layers followed by the auxiliary classifier which contains 2 fully connected layers (FC) and 1 softmax layer. Each of the first two layers can be subdivided into 3 components: convolutional layer with rectified linear units (ReLUs), local response normalization layer (LRN) and max pooling layer. Layer 3 and layer 4 contain just convolutional layer with ReLUs while layer 5 is similar to the first two layers except for the LRN. For each of the fully connected layer, 1 ReLUs and 1 dropout [94] layer are followed.
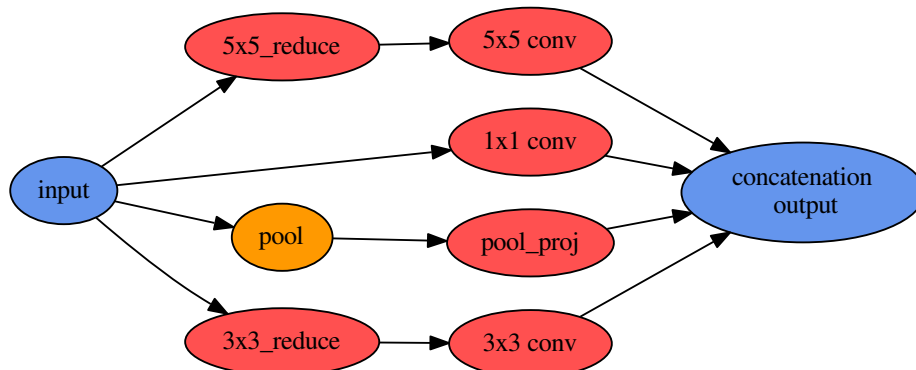
Figure 5.5: Inception Module. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 convolutional layer before the $n \times n$ convolution layer and $pool\_proj$ is another 1×1 convolutional layer after the MAX pooling layer. The output layer concatenates all its input layers.

GoogLeNet shows another trend of deep CNN architecture with lots of small receptive fields. There are 9 Inception modules in GoogLeNet and Figure 5.5 shows the architecture of a single inception module. Inspired by [68], lots of 1×1 convolutional layers are used for computational efficiency. Another interesting feature of GoogLeNet is that there are two extra auxiliary classifiers in intermediate layers. During the training procedure, the loss of these two classifiers is counted into the total loss with a discount weight 0.3, in addition to the loss of the classifier on top. More architecture details can be found from [96].

5.4.2   Food Datasets

Besides ImageNet dataset, there are many popular benchmark datasets for image recognition such as Caltech dataset and CIFAR dataset, both of which contain hundreds of classes. However, in this paper, we try to focus on a more specific area, food classification. Compared to other recognition tasks, there are some properties of the food (dishes) which make the tasks become a real challenge:

- Food doesn't have any distinctive spatial layout: for other tasks like scene recognition, we can always find some discriminative features such as buildings or trees, etc;

- Food class is a small sub-category among all the categories in daily life, so the inter-class variation is relatively small; on the other hand, the contour of food varies depending on

      many aspects such as the point of view or even its components.

These properties make food recognition catastrophic for some recognition algorithms. Therefore, the training these two architectures on the food recognition task can reveal some important aspects of themselves and help people better understand them. In this paper, we use two image datasets Food-256 [54][1] and Food-101 [14][2]. It is worthy to mention that PFID dataset is also a big public image database for classification, but their images are collected in a laboratory condition which is considerably not applicable for real recognition task.

Food-256 Dataset. This is a relatively small dataset containing 256 kinds of foods and 31644 images from various countries such as French, Italian, US, Chinese, Thai, Vietnamese, Japanese and Indonesia. The distribution among classes is not even and the biggest class (vegetable tempura) contains 731 images while the smallest one contains just 100 images. For this small dataset, we randomly split the data into training and testing set, using around 80% (25361 images) and 20% (6303 images) of the original data respectively and keep the class distribution in these two sets uniform. The collector of this dataset also provides boundary box for each image to separate different foods and our dataset is cropped according to these boundary boxes.

Food-101 Dataset. This dataset contains 101-class real-world food (dish) images which were taken and labeled manually. The total number of images is 101,000 and there are exactly 1000 images for each class. Also, each class has been divided into training and testing set containing 750 images and 250 images respectively by its collector. The testing set is well cleaned manually while the training set is not well cleaned on purpose. This noisy training set is more similar to our real recognition situation and it is also a good way to see the effect of the noise on these two architectures.

5.4.3 Data Augmentation

In this section, we introduce some data augmentation methods in our work to enrich our training data. Previous work shows that with intensive augmentation for the training data, the performance of CNN model can be improved [109]. Data Augmentation is an efficient way to enrich the data. There are also some techniques that can apply to enlarge the dataset such as subsampling and mirroring. The original images are firstly resized to 256×256 pixels. We crop the 4 corners and center for each image according to the input size of each model and flap the 5 cropped images to obtain 10 crops. For the testing set, the prediction of an image is the average prediction of the 10 crops (see Figure 5.6).

---

[1]Dataset can be found http://foodcam.mobi/dataset.html

[2]Dataset can be found http://www.vision.ee.ethz.ch/datasets_extra/food-101

(a) Original image

(b) Center

(c) Center mirror

(d) Up-left

(e) Up-left mirror

(f) Up-right

(g) Up-right mirror

(h) Bottom-left

(i) Bottom-left mirror

(j) Bottom-right

(k) Bottom-right mirror
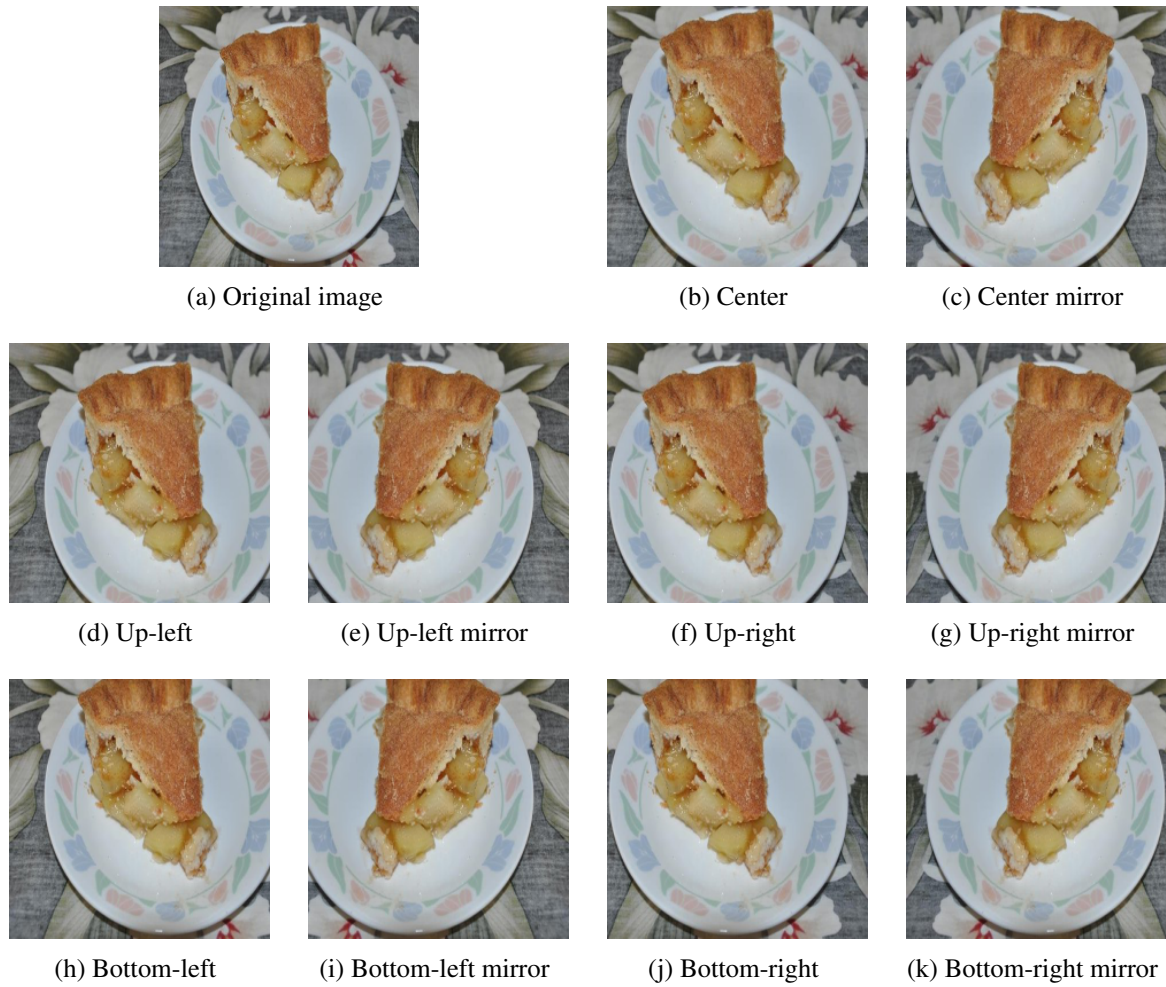
Figure 5.6: Crop area from original image

Batch Size	16
Learning Rate	0.01
Learning Rate Decay Policy	Drop 0.1 every 10000 iterations
Drop Rate	0.1
Training Iteration	100000
Momentum	0.9
Weight Decay Rate	0.0002

Table 5.1: Experimental configuration for GoogLeNet

Before cropping subsamples from the original image, we also use other augmentation methods such as color casting and vignette etc., to enrich our data and make our model less sensitive to lighting changes and other invariance (see Figure 5.7).

Compared to color shifting in [58], we use color casting to alter the intensities of the RGB channels in training images. For each image, we firstly use a random boolean parameter to determine whether its R, G, and B channel should be changed. For any channel that should be changed, we add a random integer ranging from $[-20, 20]$ to this specific channel. We also apply vignetting effect to the original image. In our implementation, we apply a 2D Gaussian kernel on the original image for vignetting. The two parameter σ_x and σ_y are randomly chosen from $[160, 200)$. We also apply some geometric transformation such as stretching and rotation, on the original image for data augmentation. In summary, we enriched the data by 11 times, 3 times color shifting, 2 times vignetting, 4 times stretching and 1 time rotation and plus the original image.

During the fine-tuning process, we initial the learning rate 0.01 and it decreases 90% (times 0.1) every 10000 iterations. The detail training configuration is shown in Table 5.1.

5.5 Discussion

Training a CNN with millions of parameters on a small dataset could easily lead to horrible overfitting. But the idea of supervised pre-training on some huge image datasets could prevent this problem in a certain degree. Compared to other randomly initialized strategies with a certain distribution, supervised pre-training is to initialize the weights according to the model trained for a specific task. Indeed, initialization using the pre-trained model has certain bias as there is no single dataset including all the invariance for natural images [1], but this bias can be reduced as the pre-trained image dataset increases and the fine-tuning should benefit from it.

(a) Original image

(b) Red casting

(c) Green casting

(d) Blue casting

(e) RGB casting

(f) Vignette

(g) More vignette

(h) Horizontal stretch

(i) More horizontal stretch

(j) Vertical stretch

(k) More vertical stretch

(l) Rotation
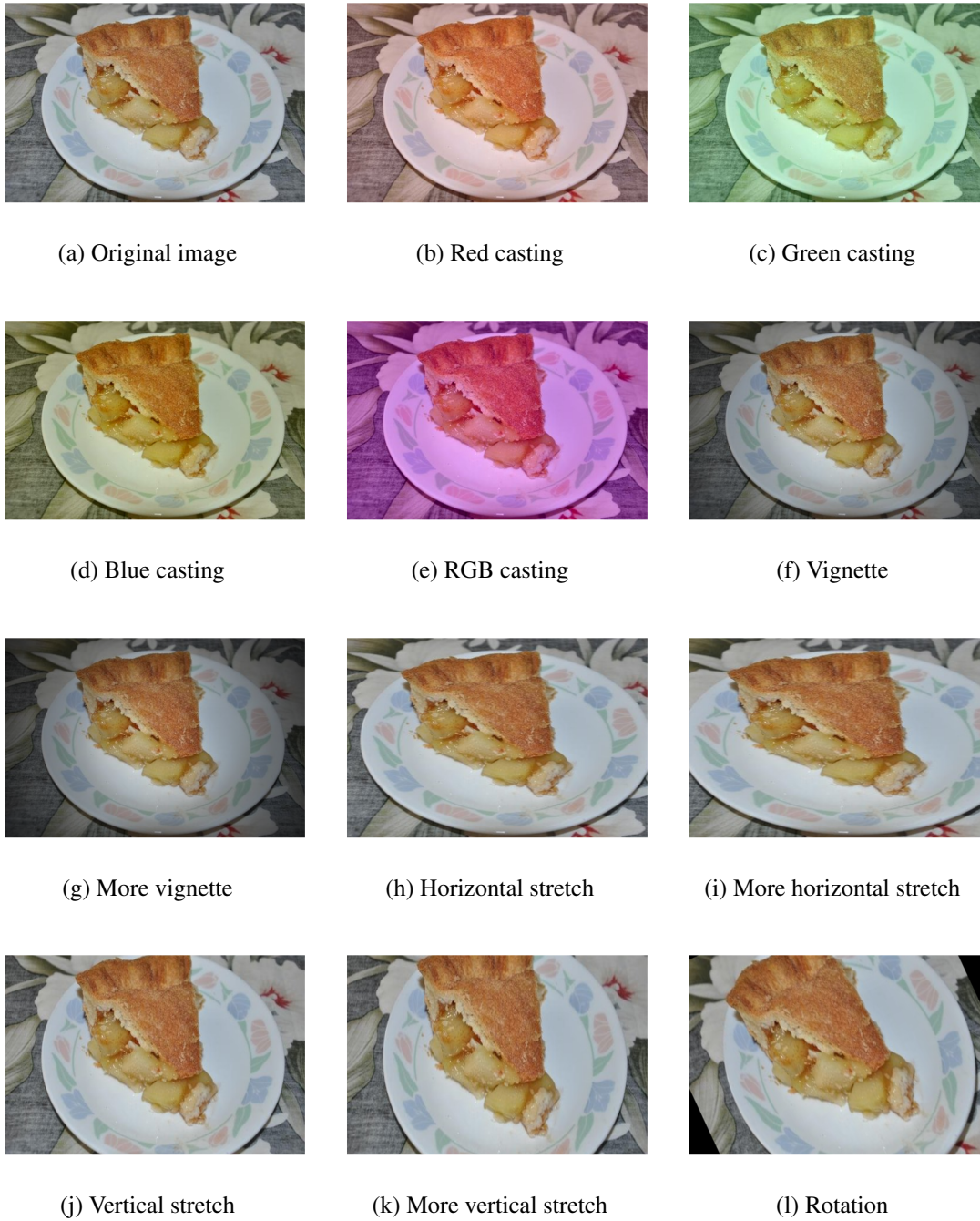
Figure 5.7: Different data augmentation methods

5.5.1 Pre-training and Fine-tuning

We conduct several experiments on both architectures and use different training initialization strategies for both Food-256 and Food-101 datasets. The scratch models are initialized with Gaussian distribution for AlexNet and Xavier algorithm for GoogLeNet[41]. These two initializations are used for training the original models for the ImageNet task. The ft-last and fine-tuned models are initialized with the weights pre-trained from the ImageNet dataset. For the ft-last model, we just re-train the fully connected layers while the whole network is fine-tuned for the fine-tune model.

Table 5.2: Top-5 Accuracy in percent on fine-tuned, ft-last and scratch model for two architectures

	AlexNet		GoogLeNet	
	Food-101	Food-256	Food-101	Food-256
Fine-tune	**88.12**	**85.59**	**93.51**	**90.66**
Ft-last	76.49	79.26	82.84	83.77
Scratch	78.18	75.35	90.45	81.20

Table 5.3: Accuracy compared to other methods on Food-256 dataset in percent

	fv+linear [55]	GoogLeNet	AlexNet
Top1	50.1	**70.13**	63.82
Top5	74.4	**90.66**	85.59

Table 5.4: Top-1 accuracy compared to other methods on Food-101 dataset in percent

	RFDC[14]	MLDS($\approx$[93])	GoogLeNet	AlexNet
Top1 accuracy	50.76	42.63	**78.11**	66.40

From Table 5.2 we can see that fine-tuning the whole network can improve the performance of the CNN for our task. Compared to other traditional computer vision methods (see Table 5.3 and 5.4), GoogLeNet outperforms the other methods with large margins and we provide the state-of-the-art performance of these two food image datasets.

In Figure 5.8 we visualize the feature maps of the pre-trained GoogLeNet model and fined-tuned GoogLeNet model with the same input image for some layers. We can see that the feature maps of the lower layer are similar as the lower level features are similar for most recognition
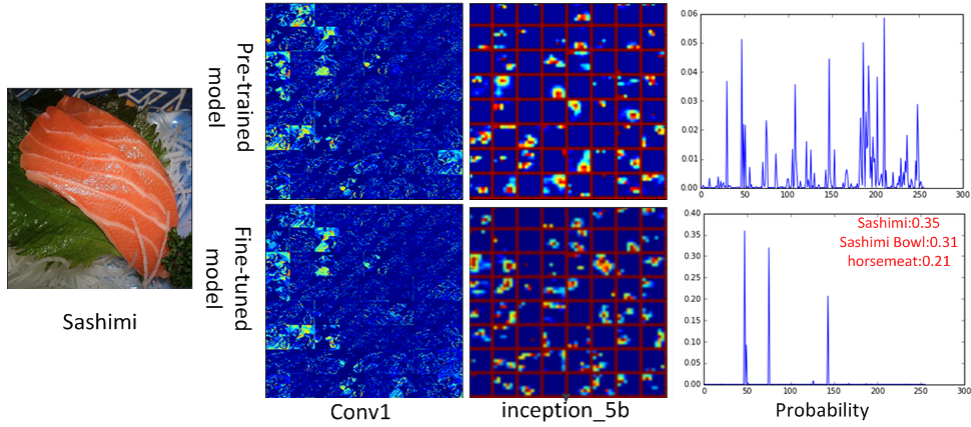
Figure 5.8: Visualization of some feature maps of different GoogLeNet models in different layers for the same input image. 64 feature maps of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

tasks. Then we can see that the feature maps in the high-level are different which leads to totally different recognition results. Since only the last layer (auxiliary classifier) of the ft-last model is optimized, we can infer that the higher level features are more important which is consistent with our intuition. Also from Table 5.2, it is interesting to see that for the Food-101 task, the accuracy of the scratch models outperforms the pre-trained models. Since Food-101 is a relatively large dataset with 750 images per class while Food-256 dataset is an imbalanced small one, this indicates that it is difficult to obtain a good deep CNN model while the data is insufficient.

From Table 5.2 we can see that GoogLeNet always performances better than AlexNet on both datasets. This implies that the higher level features of GoogLeNet are more discriminative compared to AlexNet and this is due to the special architecture of its basic unit, Inception module. Table 5.5 and 5.6 show the weights' cosine similarity of each layer between the fine-tuned models and their pre-trained models. From the results we can see that the weights in the low layer are more similar which implies that these two architectures can learn the hierarchical features. As the low-level features are similar for most of the tasks, the difference of the objects is determined by high-level ones which are the combination of these low-level features. Also from Table 5.6, we can observe that the weights of the pre-trained and fine-tuned models are extremely similar in AlexNet. This can be caused by the size of receptive filed. Since ReLUs are used in both architectures, vanishing gradients do not exist. Rectified activation function is mathematically given by:

$$h = \max(w^T x, 0) = \begin{cases} w^T x & w^T x > 0 \\ 0 & else \end{cases} \tag{5.3}$$

The ReLU is inactivated when its input is below 0 and its partial derivative is 0 as well. Sparsity can improve the performance of the linear classifier on top, but on the other hand, sparse representations make the network more difficult to train as well as fine-tune. The derivative of the filter is $\frac{\partial J}{\partial w} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w} = \frac{\partial J}{\partial y} * x$ where $\frac{\partial J}{\partial y}$ denotes the partial derivative of the activation function, $y = w^T x$ and x denotes the inputs of the layer. The sparse input could lead to sparse filter derivative for back propagation which would eventually prevent the errors passing down effectively. Therefore, the filters of the fine-tuned AlexNet is extremely similar. Compared to large receptive field used in AlexNet, the inception module in GoogLeNet employs 2 additional $n \times n\_reduced$ convolutional layers before the 3×3 and 5×5 convolutional layers (see Figure 5.5). Even though the original purpose of these two 1×1 convolutional layer is for computational efficiency, these 2 convolutional layers tend to squeeze their sparse inputs and generate the dense outputs for the following layer. We can see from Table 5.7 that the sparsity of the $n \times n\_reduce$ layers are denser than other layers within the inception module. This makes the filters in the following layer more easily to be trained for transfer learning and generate efficient sparse representations.

The unique structure of the Inception module guarantees that the sparse outputs from the previous layer can be squeezed with the 1×1 convolutional layers and feed to convolutional layers with a bigger receptive field to generate sparser representation. The squeeze action promises the back propagation error can be transferred more efficiently and makes the whole network more flexible to fit different recognition tasks.

5.5.2 Learning across the datasets

From the previous experiments we can see that pre-training on the ImageNet dataset can improve the performance of the deep convolutional neural network in our specific area. In this part, we will discuss the generalization ability within the food recognition problem. Zhou et al. trained AlexNet for Scene Recognition across two datasets with identical categories [116]. But for more complex situation, such as two similar datasets with a little overlapped categories, we are very interested in exploring whether deep CNN can still successfully handle. Therefore, we conduct the following experiment to stimulate a more challenging real world problem: transferring the knowledge from the fine-tuned Food-101 model to a target set, Food-256 dataset. To make the experiment more practical, we limit the number of samples per category from Food-256 for training, because if we want to build our model using deep CNN for a specific task, the resource is always limited and it is exhausted to collect hundreds of labeled images for each category.

The Food-101 and Food-256 datasets share about 46 categories of food even though the

Table 5.5: Cosine similarity of the layers in Inception modules between fine-tuned models and pre-trained model for GoogLeNet

food256						
	1x1	3x3_reduce	3x3	5x5_reduce	5x5	pool_proj
inception_3a	0.72	0.72	0.64	0.67	0.73	0.69
inception_3b	0.59	0.64	0.53	0.70	0.60	0.56
inception_4a	0.46	0.53	0.54	0.50	0.67	0.38
inception_4b	0.55	0.58	0.63	0.52	0.69	0.41
inception_4c	0.63	0.64	0.63	0.57	0.68	0.52
inception_4d	0.60	0.62	0.60	0.58	0.68	0.50
inception_4e	0.60	0.61	0.67	0.61	0.68	0.50
inception_5a	0.51	0.53	0.58	0.48	0.60	0.39
inception_5b	0.40	0.44	0.50	0.41	0.59	0.40
food101						
	1x1	3x3_reduce	3x3	5x5_reduce	5x5	pool_proj
inception_3a	0.71	0.72	0.63	0.67	0.73	0.68
inception_3b	0.56	0.63	0.50	0.71	0.60	0.53
inception_4a	0.43	0.50	0.50	0.47	0.62	0.36
inception_4b	0.48	0.52	0.57	0.50	0.67	0.35
inception_4c	0.57	0.61	0.59	0.53	0.63	0.47
inception_4d	0.54	0.58	0.53	0.54	0.64	0.44
inception_4e	0.53	0.54	0.61	0.55	0.62	0.42
inception_5a	0.43	0.47	0.53	0.45	0.57	0.34
inception_5b	0.36	0.39	0.46	0.38	0.52	0.37

Table 5.6: Cosine similarity of the layers between fine-tuned models and pre-trained model for AlexNet

	conv1	conv2	conv3	conv4	conv5	fc6	fc7
food256	0.997	0.987	0.976	0.976	0.978	0.936	0.923
food101	0.996	0.984	0.963	0.960	0.963	0.925	0.933

Table 5.7: Sparsity of the output for each unit in GoogLeNet inception module for training data from Food101 in percent

	1x1	3x3_reduce	3x3	5x5_reduce	5x5	pool_proj
inception_3a	69.3 ± 1.3	69.6 ± 1.1	80.0 ± 1.0	64.1 ± 2.2	75.8 ± 1.6	76.2 ± 5.4
inception_3b	92.8 ± 0.9	76.5 ± 0.9	94.7 ± 0.9	71.6 ± 2.3	94.4 ± 0.5	94.7 ± 1.6
inception_4a	90.9 ± 0.9	70.0 ± 1.2	93.8 ± 1.1	63.3 ± 4.0	91.9 ± 1.8	95.1 ± 2.0
inception_4b	71.9 ± 1.6	67.5 ± 1.2	75.4 ± 1.0	58.5 ± 2.6	78.9 ± 1.6	85.6 ± 3.6
inception_4c	75.1 ± 2.4	72.6 ± 1.3	81.0 ± 2.0	66.3 ± 6.1	79.7 ± 3.6	88.1 ± 3.3
inception_4d	87.3 ± 2.7	78.0 ± 2.2	88.0 ± 1.6	67.9 ± 3.1	88.9 ± 2.8	93.0 ± 2.2
inception_4e	91.8 ± 1.1	62.3 ± 2.2	91.0 ± 2.5	49.5 ± 3.7	94.0 ± 1.0	92.3 ± 1.5
inception_5a	78.7 ± 1.6	66.5 ± 1.7	82.3 ± 2.6	59.9 ± 3.2	86.4 ± 2.3	87.1 ± 2.6
inception_5b	88.2 ± 2.3	86.8 ± 1.6	83.3 ± 4.4	84.0 ± 3.1	81.4 ± 5.3	94.7 ± 1.5

	AlexNet		GoogLeNet	
instances per class	ImageNet	Food101_ft	ImageNet	Food101_ft
20	68.80	75.12	74.54	77.77
30	73.15	77.02	79.21	81.06
40	76.04	80.23	81.76	83.52
50	78.90	81.66	84.22	85.84
all	85.59	87.21	90.66	90.65

Table 5.8: Top5 Accuracy for transferring from Food101 to subset of Food256 in percent

images in the same category may vary across these two datasets. The types of food in Food-101 are mainly western style while most types of food in Food-256 are typical Asian foods. We compare the top-5 accuracy trained from different size of the subset for Food-256 on a different pre-trained model and the results are shown in Table 5.8. The ImageNet columns denote using the model pre-trained from ImageNet dataset as the pre-trained model and Food101_ft columns denote using the fine-tuned Food-101 model (the same one in Table 5.2) as the pre-trained model.

From the result of Table 5.8 we can see that, with this further transfer learning, both CNNs can achieve around 95% of the accuracy trained on full dataset while just utilizing about half of them (50 per class, 12800 of 25361 images). This indicates that when there is not enough labeled data, with its strong generalization ability, deep CNN trained from a general task can still achieve a satisfying result and perform even better when an additional relevant dataset is involved. This encouraging result may attract more people to use deep CNN for their specific task and continue to explore the potential of the existing architecture as well as designing new ones.

5.6   Summary

In this chapter, we show that fine-tuning GoogLeNet on two food databases can effectively transfer the knowledge from the general image recognition task to some specific recognition tasks. Without using any information of the source data, fine-tuning the deep CNNs on the target data shows impressive results. By comparing the changes of the parameters in each layer, we found that the high-level features can be easily changed while most low-level features are kept in fine-tuning.

Chapter 6

Conclusion

Large object recognition task can be effectively solved with deep CNNs, but learning from a small size of data is still challenging. Transfer learning becomes a popular way to solve the small data regime by leveraging knowledge from learned tasks. In this thesis, we investigate the visual transfer learning problem in two scenarios under the setting where the source data is absent. Transfer learning under this setting is common and investigating the transfer learning problem in the absence of the source data is meaningful for the practical problems. The main contributions of this thesis are as follows:

- In chapter 3, we investigated the supervised domain adaption problem under the HTL setting. We proposed EMTLe that can leverage the knowledge from the source model. Compared to previous methods, EMTLe can better leverage the source knowledge and achieve improved performance.

- In chapter 4, we proposed a framework called GDSDA for semi-supervised domain adaptation, which can leverage the knowledge from the source model in the semi-supervised learning scenario. To make GDSDA more practical, we then proposed GDSDA-SVM as an example that uses SVM as the classifier in GDSDA. Experimental results show that GDSDA can effectively leverage the source knowledge for semi-supervised domain adaptation problem.

- Finally, we investigated the problem of fine-tuning the deep CNNs for food recognition tasks. We compared the performances of two CNNs architectures and found that GoogLeNet is more suitable as the pre-trained model for transfer learning.

Visual transfer learning in the absence of the source data is challenging and important in many real transfer learning scenarios. How to better leverage the knowledge from the source data and void negative transfer at the same time is still an open question. In this thesis, we

provided a few methods for this problem. In our future work, we plan to use the deep neural network for visual transfer task. There are still many challenges in apply deep neural network for transfer learning. An important issue of applying deep transfer learning is to solve the problem of overfitting. Possible solutions could be eliminating the redundant "nodes" in deep neural networks while keeping those informative "nodes" to reduce the size of the net and therefore, can better avoid overfitting problem.

Bibliography

[1] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *Computer Vision–ECCV 2014*, pages 329–344. Springer, 2014.

[2] MA Aizerman, E Braverman, and LI Rozonoer. Probability problem of pattern recognition learning and potential functions method. *Avtomat. i Telemekh*, 25(9):1307–1323, 1964.

[3] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[4] Yusuf Aytar and Andrew Zisserman. Tabula rasa: Model transfer for object category detection. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2252–2259. IEEE, 2011.

[5] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[6] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *The Journal of Machine Learning Research*, 4:83–99, 2003.

[7] Pierre Baldi and Yves Chauvin. Neural networks for fingerprint recognition. *Neural Computation*, 5(3):402–418, 1993.

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[9] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[10] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[11] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards ai. *Large-Scale Kernel Machines*, 34, 2007.

[12] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[13] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 232–237. IEEE, 1998.

[14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014*, pages 446–461. Springer, 2014.

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

[16] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010.

[17] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[18] Gavin C Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1661–1668. IEEE, 2006.

[19] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[20] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High performance convolutional neural networks for document processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006.

[21] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[22] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223, 2011.

[23] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[24] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[25] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

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

[27] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[28] Hal Daumé III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics, 2010.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[30] Jeff Donahue, Judy Hoffman, Erik Rodner, Kate Saenko, and Trevor Darrell. Semi-supervised domain adaptation with instance constraints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[31] Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 289–296. ACM, 2009.

[32] Lixin Duan, Dong Xu, and Ivor W. Tsang. Learning with augmented features for heterogeneous domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 711–718, Edinburgh, Scotland, 2012. Omnipress.

[33] Lixin Duan, Dong Xu, Ivor Wai-Hung Tsang, and Jiebo Luo. Visual event recognition in videos by learning from web data. volume 34, pages 1667–1680. IEEE, 2012.

[34] A Evgeniou and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007.

[35] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

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

[37] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

[38] Stanley J Farlow. *Self-organizing methods in modeling: GMDH type algorithms*, volume 54. CrC Press, 1984.

[39] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):594–611, 2006.

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

[41] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[42] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for super-vision transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[43] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2014.

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

[45] Judy Hoffman, Eric Tzeng, Jeff Donahue, Yangqing Jia, Kate Saenko, and Trevor Darrell. One-shot adaptation of supervised deep convolutional models. *arXiv preprint arXiv:1312.6204*, 2013.

[46] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[47] Saburo Ikeda, Mikiko Ochiai, and Yoshikazu Sawaragi. Sequential gmdh algorithm and its application to river flow prediction. *Systems, Man and Cybernetics, IEEE Transactions on*, (7):473–479, 1976.

[48] Alexey Grigorevich Ivakhnenko and Valentin Grigorévich Lapa. *Cybernetic predicting devices*. CCM Information Corporation, 1965.

[49] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

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

[51] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pages 264–271, 2007.

[52] Luo Jie, Tatiana Tommasi, and Barbara Caputo. Multiclass transfer learning from un-constrained priors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1863–1870. IEEE, 2011.

[53] Søren Johansen and Katarina Juselius. Maximum likelihood estimation and inference on cointegrationwith applications to the demand for money. *Oxford Bulletin of Economics and statistics*, 52(2):169–210, 1990.

[54] Y. Kawano and K. Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

[55] Yoshiyuki Kawano and Keiji Yanai. Foodcam-256: A large-scale real-time mobile food recognitionsystem employing high-dimensional features and compression of classifier weights. In *Proceedings of the ACM International Conference on Multimedia*, MM '14, pages 761–762, New York, NY, USA, 2014. ACM.

[56] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[57] Tadashi Kondo and Junji Ueno. Multi-layered gmdh-type neural network self-selecting optimum neural network architecture and its application to 3-dimensional medical image recognition of blood vessels. *International Journal of innovative computing, information and control*, 4(1):175–187, 2008.

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

[59] Gregory Kuhlmann and Peter Stone. Graph-based domain mapping for transfer learning in general games. In *Machine Learning: ECML 2007*, pages 188–200. Springer, 2007.

[60] Ilja Kuzborskij and Francesco Orabona. Stability and hypothesis transfer learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 942–950, 2013.

[61] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. From n to n+ 1: Multiclass transfer incremental learning. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3358–3365. IEEE, 2013.

[62] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[63] B Boser Le Cun, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.

[64] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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

[66] Xuejun Liao, Ya Xue, and Lawrence Carin. Logistic regression with an auxiliary data source. In *Proceedings of the 22nd international conference on Machine learning*, pages 505–512. ACM, 2005.

[67] Joseph Jaewhan Lim. *Transfer learning by borrowing examples for multiclass object detection*. PhD thesis, Massachusetts Institute of Technology, 2012.

[68] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[69] D. Lopez-Paz, B. Schölkopf, L. Bottou, and V. Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations*, 2016.

[70] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[71] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: a survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[72] Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, and Xiaoou Tang. Face model compression by distilling knowledge from neurons. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[73] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[74] Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. *arXiv preprint arXiv:1503.01558*, 2015.

[75] Christopher Poultney Marc'Aurelio Ranzato, Sumit Chopra, and Yann Lecun. Efficient learning of sparse representations with an energy-based model. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS 2006*. Citeseer, 2006.

[76] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Information bottleneck learning using privileged information for visual recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[77] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[78] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[79] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[80] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 737–746, 2016.

[81] John Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[82] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[83] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

[84] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *In Proceedings of International Conference on Learning Representations*, 2015.

[85] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[86] Frank Rosenblatt. Principles of neurodynamics. 1962.

[87] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Transfer Learning*, volume 898, 2005.

[88] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[89] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE, 2004.

[90] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[91] Viktoriia Sharmanska, Novi Quadrianto, and Christoph H. Lampert. Learning to rank using privileged information. In *The IEEE International Conference on Computer Vision (ICCV)*, 2013.

[92] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[93] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–ECCV 2012*, pages 73–86. Springer, 2012.

[94] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[95] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[96] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[97]   Erik Talvitie and Satinder P Singh. An experts algorithm for transfer learning. In *IJCAI*, pages 1065–1070, 2007.

[98]   Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3081–3088. IEEE, 2010.

[99]   Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. Learning categories from few examples with multi model knowledge transfer. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(5):928–941, 2014.

[100] Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[101] Lisa Torrey, Trevor Walker, Jude Shavlik, and Richard Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Machine Learning: ECML 2005*, pages 412–424. Springer, 2005.

[102] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[103] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[104] Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdel rahman Mohamed, Matthai Philipose, and Matthew Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? In *International Conference on Learning Representations (workshop track)*, 2016.

[105] Joost Van De Weijer and Cordelia Schmid. Coloring local feature extraction. In *Computer Vision–ECCV 2006*, pages 334–348. Springer, 2006.

[106] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*, 16:2023–2049, 2015.

[107] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[108] Marcin Witczak, Józef Korbicz, Marcin Mrugalski, and Ron J Patton. A gmdh neural network-based approach to robust fault diagnosis: Application to the damadics benchmark problem. *Control Engineering Practice*, 14(6):671–683, 2006.

[109] Ren Wu, Shengen Yan, Yi Shan, Qingqing Dang, and Gang Sun. Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*, 2015.

[110] Jianchao Yang, Kai Yu, Yihong Gong, and Tingwen Huang. Linear spatial pyramid matching using sparse coding for image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1794–1801. IEEE, 2009.

[111] Jun Yang, Rong Yan, and Alexander G Hauptmann. Adapting svm classifiers to data with shifted distributions. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 69–76. IEEE, 2007.

[112] Jun Yang, Rong Yan, and Alexander G Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th international conference on Multimedia*, pages 188–197. ACM, 2007.

[113] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2142–2150, 2015.

[114] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[115] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[116] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014.

Appendix A

Proofs of Theorems

A.1 Closed-form Leave-out Error for LS-SVM

Theorem A.1.1 (Extension of [18]) *Given a dataset $D = \{(x_i, y_i)|i = 1, ..., l\}$, the solution of a LS-SVM on D can be written as:*

$$
\left[\begin{array}{cc} K + \frac{1}{C}I & 1 \\ 1^T & 0 \end{array} \right] \left[\begin{array}{c} \alpha \\ b \end{array} \right] = \left[\begin{array}{c} y \\ 0 \end{array} \right]
\tag{A.1}
$$

Assume that $D^{(n)} = \{(x_i, y_i)|i = 1, ..., n\}$ is a subset of D and $D \backslash D^{(n)}$ is the complement of $D^{(n)}$ in D , The unbiased leave out error of a LS-SVM trained from $D \backslash D^{(n)}$ on $D^{(n)}$ can be estimated as:

$$
ERR_{leave-out} = \left(S_n - sS^{-1}_{(l-n+1)}s^T \right) [\alpha_1, ..., \alpha_n]^T
$$

Where $[\alpha_1, ..., \alpha_n]$ is the first n rows of α in (A.1). S_n, s and $S_{(l-n+1)}$ are the square blocks of matrix:

$$
\left[\begin{array}{c|c} S_n & s \\ \hline s^T & S_{(l-n+1)} \end{array} \right] = \left[\begin{array}{cc} K + \frac{1}{C}I & 1 \\ 1^T & 0 \end{array} \right]
$$

Proof Following the result of Eq. (A.1) and noticing that the matrix of the left hand in Eq. (A.1) is symmetrical, it can be written as follow:

$$
\left[\begin{array}{cc} K + \frac{1}{C}I & 1 \\ 1^T & 0 \end{array} \right] = \left[\begin{array}{c|c} S_n & s \\ \hline s^T & S_{(l-n+1)} \end{array} \right]
\tag{A.2}
$$

Where $S_n \in R^{n \times n}$, $s \in R^{n \times (l-n+1)}$ and $S_{(l-n+1)} \in R^{(l-n+1) \times (l-n+1)}$.

For the first round of the cross validation situation, assume the first n examples are used as the validation set. In this case, let $\left[\alpha^{-1}, b^{-1}\right] \in R^{l-n+1}$ denote the optimal parameters for a LS-SVM f^{-1} trained on the rest of the samples and they can be found by:

$$\begin{bmatrix} \alpha^{-1} \\ b^{-1} \end{bmatrix} = S_{(l-n+1)}^{-1}[y_{n+1}, ..., y_l, 0]^T \tag{A.3}$$

The prediction of f^{-1} on the validation set $\hat{Y} = [\hat{y}_1, ..., \hat{y}_n]$ is given by:

$$[\hat{y}_1, ..., \hat{y}_n]^T = s \begin{bmatrix} \alpha^{-1} \\ b^{-1} \end{bmatrix} = sS_{(l-n+1)}^{-1}[y_{n+1}, ..., y_l, 0]^T \tag{A.4}$$

Moreover, the last $l - n + 1$ rows in Eq. (A.1) can be represented as $\begin{bmatrix} s^T & S_{l-n+1} \end{bmatrix} [\alpha, b]^T = [y_{n+1}, ..., y_l, 0]^T$. So

$$\begin{aligned} \hat{Y} = [\hat{y}_1, ..., \hat{y}_n]^T &= sS_{(l-n+1)}^{-1} \begin{bmatrix} s^T & S_{l-n+1} \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} \\ &= sS_{(l-n+1)}^{-1} s^T [\alpha_1, ..., \alpha_n]^T + s[\alpha_{n+1}, ..., \alpha_l, b]^T \end{aligned} \tag{A.5}$$

Then first n rows in Eq. (A.1) can be represented as:

$$Y = [y_1, ...y_n]^T = \begin{bmatrix} S_n & s \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = S_n[\alpha_1, ..., \alpha_n]^T + s[\alpha_{n+1}, ..., \alpha_l, b]^T \tag{A.6}$$

Thus, combining (A.5) and (A.6), we have the following equation:

$$\begin{aligned} \hat{Y} &= Y - S_n[\alpha_1, ..., \alpha_n]^T + sS_{(l-n+1)}^{-1} s^T [\alpha_1, ..., \alpha_n]^T \\ &= Y - \left(S_n - sS_{(l-n+1)}^{-1} s^T \right) [\alpha_1, ..., \alpha_n]^T \end{aligned} \tag{A.7}$$

According to block matrix inversion lemma

$$\begin{bmatrix} S_n & s \\ s^T & S_{(l-n+1)} \end{bmatrix}^{-1} = \begin{bmatrix} \kappa^{-1} & -\kappa^{-1} sS_{(l-n+1)}^{-1} \\ -S_{(l-n+1)}^{-1} s^T \kappa^{-1} & S_{(l-n+1)}^{-1} + S_{(l-n+1)}^{-1} s^T \kappa^{-1} sS_{(l-n+1)}^{-1} \end{bmatrix} \tag{A.8}$$

Where $\kappa = \left(S_n - sS_{(l-n+1)}^{-1} s^T \right)$. We have:

$$Y - \hat{Y} = \kappa[\alpha_1, ..., \alpha_n]^T \tag{A.9}$$

A.2   Converage of EMTLe

Theorem A.2.1 *Let $L(\beta)$ be a λ-strongly convex function and β^* be its optimal solution. Let $\beta_1, ..., \beta_{T+1}$ be a sequence such that $\beta_1 \in B$ and for $t > 1$, we have $\beta_{t+1} = \beta_t - \eta_t \Delta_t$, where Δ_t is*

the sub-gradient of $L(\beta_t)$ and $\eta_t = 1/(\lambda t)$. Assume we he $\|\Delta_t\| \leq G$ for all t. Then we have:

$$L(\beta_{T+1}) \leq L(\beta^*) + \frac{G^2(1 + \ln(T))}{2\lambda T} \tag{A.10}$$

Proof: As $L(\beta)$ is strongly convex and Δ_t is in its sub-gradient set at β_t, according to the definition of λ-strong convexity [83], the following inequality holds:

$$\langle \beta_t - \beta^*, \Delta_t \rangle \geq L(\beta_t) - L(\beta^*) + \frac{\lambda}{2} \|\beta_t - \beta^*\|^2 \tag{A.11}$$

For the term $\langle \beta_t - \beta^*, \Delta_y \rangle$, it can be written as:

$$
\begin{aligned}
\langle \beta_t - \beta^*, \Delta_t \rangle &= \left\langle \beta_t - \frac{1}{2}\eta_t\Delta_t + \frac{1}{2}\eta_t\Delta_t - \beta^*, \Delta_t \right\rangle \\
&= \frac{1}{2} \left\langle [(\beta_t - \eta_t\Delta_t) - \beta^*] + (\beta_t - \beta^*) + \eta_t\Delta_t, \Delta_t \right\rangle \\
&= \frac{1}{2} \left\langle (\beta_{t+1} - \beta^*) + (\beta_t - \beta^*), \Delta_t \right\rangle + \frac{1}{2}\eta_t\Delta_t^2 \\
&= \frac{1}{2} \langle \beta_{t+1} + \beta_t - 2\beta^*, \Delta_t \rangle + \frac{1}{2}\eta_t\Delta_t^2
\end{aligned}
\tag{A.12}
$$

Then we have:

$$\|\beta_t - \beta^*\|^2 - \|\beta_{t+1} - \beta^*\|^2 = \langle \beta_{t+1} + \beta_t - 2\beta^*, \eta_t\Delta_t \rangle \tag{A.13}$$

Using the assumption $\|\Delta_t\| \leq G$, we can rearrange (A.11) and plug (A.12) and (A.13) into it, we have:

$$
\begin{aligned}
Diff_t &= L(\beta_t) - L(\beta^*) \\
&\leq \frac{\|\beta_t - \beta^*\|^2 - \|\beta_{t+1} - \beta^*\|^2}{2\eta_t} - \frac{\lambda}{2}\|\beta_t - \beta^*\|^2 + \frac{1}{2}\eta_t\Delta_t^2 \\
&\leq \frac{\|\beta_t - \beta^*\|^2 - \|\beta_{t+1} - \beta^*\|^2}{2\eta_t} - \frac{\lambda}{2}\|\beta_t - \beta^*\|^2 + \frac{1}{2}\eta_t G^2 \\
&\leq \frac{\lambda(t-1)}{2}\|\beta_t - \beta^*\|^2 - \frac{\lambda t}{2}\|\beta_{t+1} - \beta^*\|^2 + \frac{1}{2}\eta_t G^2
\end{aligned}
\tag{A.14}
$$

Due to the convexity, for each pair of $L(\beta_t)$ and $L(\beta_{t+1})$ for $t = 1, ..., T$, we have the following sequence $L(\beta^*) \leq L(\beta_T) \leq L(\beta_{T-1}) \leq ... \leq L(\beta_1)$. For the sequence $Diff_t$ for $t = 1, ..., T$, we have:

$$\sum_{t=1}^{T} Diff_t = \sum_{t=1}^{T} L(\beta_t) - TL(\beta^*) \geq T\left[L(\beta_T) - L(\beta^*)\right] \tag{A.15}$$

Next, we show that

$$
\begin{aligned}
\sum_{t=1}^{T} Diff_t &= \sum_{t=1}^{T} \left\{ \frac{\lambda(t-1)}{2}\|\beta_t - \beta^*\|^2 - \frac{\lambda t}{2}\|\beta_{t+1} - \beta^*\|^2 + \frac{1}{2}\eta_t G^2 \right\} \\
&= -\frac{\lambda T}{2}\|\beta_{T+1} - \beta^*\|^2 + \frac{G^2}{2\lambda}\sum_{t=1}^{T}\frac{1}{t} \leq \frac{G^2}{2\lambda}\sum_{t=1}^{T}\frac{1}{t} \leq \frac{G^2}{2\lambda}(1 + \ln(T))
\end{aligned}
\tag{A.16}
$$

Combining (A.15) and rearranging the result, we have:

$$L(\beta_{T+1}) \le L(\beta^*) + \frac{G^2(1 + \ln(T))}{2\lambda T}$$

Appendix B

Configuration of GoogLeNet

Table B.1: Configuration of GoogLeNet

type	patch size / stride	output size	depth	#1x1	#3x3 reduce	#3x3	#5x5 reduce	#5x5 pool	proj
convolution	7x7/2	112x112x64	1						
max pool	3x3/2	56x56x64	0						
convolution	3x3/1	56x56x192	2		64	192			
max pool	3x3/2	28x28x192	0						
inception(3a)		28x28x256	2	64	96	128	16	32	32
inception(3b)		28x28x480	2	128	128	192	32	96	64
maxpool	3x3/2	14x14x480	0						
inception(4a)		14x14x512	2	192	96	208	16	48	64
inception(4b)		14x14x512	2	160	112	224	24	64	64
inception(4c)		14x14x512	2	128	128	256	24	64	64
inception(4d)		14x14x528	2	112	144	288	32	64	64
inception(4e)		14x14x832	2	256	160	320	32	128	128
maxpool	3x3/2	7x7x832	0						
inception(5a)		7x7x832	2	256	160	320	32	128	128
inception(5b)		7x7x1024	2	384	192	384	48	128	128
avg pool	7x7/1	1x1x1024	0						
dropout (40%)		1x1x1024	0						
linear		1x1x1000	1						
softmax		1x1x1000	0						

Curriculum Vitae

Name: Shuang Ao

Post-Secondary   University of Western Ontario
Education and   London, ON
Degrees: 2012 - 2017 Ph.D.

                     China University of Geosciences (Wuhan)
                     Wuhan, Hubei, China
                     2005 - 2009 B.Sc

Related Work   Teaching Assistant
Experience: The University of Western Ontario
                     2012 - 2017

                     Research Assistant
                     The University of Western Ontario
                     2012 - 2017

Publications:

Ao, Shuang, and Li, Xiang and Ling, Charles. "Fast Generalized Distillation for Semi-supervised Domain Adaptation." Thirtieth-First AAAI Conference on Artificial Intelligence, 2017.

Ao, Shuang, and Li, Xiang and Ling, Charles. "Effective Multiclass Transfer For Hypothesis Transfer Learning." Pacific-Asia Conference on Knowledge Discovery and Data Mining,

2017. (Accepted)

Ao, Shuang, and Ling, Charles. "Adapting new categories for food recognition with deep representation." 2015 IEEE International Conference on Data Mining Workshop (ICDMW).

Luo, Yan and Ling, Charles and **Ao, Shuang**. Mobile-based Food Classification For Type-2 Diabetes Using Nutrient and Textual Features. 2014 IEEE International Conference on Data Science and Advanced Analytics.