VISUAL TRANSFER LEARNING IN THE ABSENCE OF THE SOURCE DATA

(Spine title: Plib)

(Thesis format: Monograph)

by

Shuang Ao

Graduate Program in Computer Science

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

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO

School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....

Dr. Charles X. Ling

Supervisory Committee:

.....

Dr. W. J. Braun

.....

Dr. A. Bing

Examiners:

.....

Dr. Q. Ring

.....

Dr. W. Fing

.....

Dr. G. Hing

The thesis by

Shuang Ao

entitled:

Visual Transfer Learning in the Absence of the Source Data

is accepted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

.....

Date

.....

Chair of the Thesis Examination Board

Acknowlegements

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 the scenario where the source data is absent. 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 original 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 both supervised (in chapter 3 & 4) and semi-supervised learning scenario (in chapter 5). In chapter 3, we investigate the problem of fine-tuning the deep CNN to learn new food categories using the large ImageNet database as our source. 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. In chapter 4, we investigate the visual domain adaptation problem under the setting of Hypothesis Transfer Learning. We propose 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 5, we investigate the semi-supervised domain adaptation scenario under the setting of HTL and propose our framework GDSDA. Specifically, we show that GDSDA can effectively transfer the knowledge using the unlabeled data. 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 brutal force search or background knowledge, GDSDA-SVM is more effective in real applications.

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 our methods can outperform previous work.

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

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

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

In order to successfully capture the diversity of different objects around us, many recognition models contain thousands or sometimes even millions of parameters and require large amount of training images to tune these parameters as well. Unfortunately, for some real applications, it is often difficult and cost to collect 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 the 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. As more companies pay attention to the individual customer experience, personalized service system become more important in recent years. For image recognition, one application of personalization can be learning the categories defined by individuals. The challenge of the self-defined categories learning comes from two aspects:

1. Flexibility: people could define any categories they want. The recognition algorithm should be more dynamic to adapt the new categories.

2. Rare learning examples: it is almost impossible to get abundant images from the self-defined categories and the algorithm should be able to learn the new categories from just a few examples.

On the other hand, recognition is one of the most important part of our human visual system.

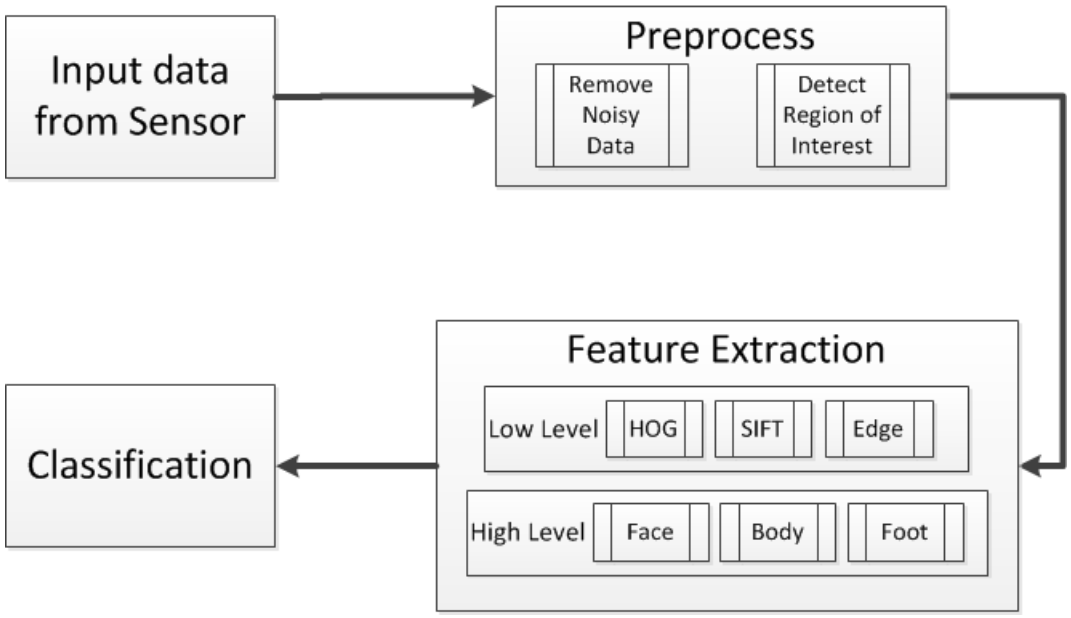


Figure 1.1: Major procedure for image recognition.

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[9]. 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 a objects with just a handful examples. It could be ideal if there is a visual recognition model that can effectively adapt new learning scenarios, which is referred to as **visual transfer learning**.

In previous studies of visual transfer learning, people assume that the source data are available and can be freely accessed which is rare in real applications due to many reasons such as privacy and disclosure

1.1 Overview for Image Recogntion

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

1.1.1 Preprocess

After receiving the raw data of a image from the sensor, preprocess generates 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 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 a type of dimensionality reduction that efficiently represents interesting parts of an image 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: one is the hand engineered method and the other one is representation learning method.

Hand Engineered Feature

Hand engineered features are 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 [97]. Common low level hand engineered features include Histogram of Oriented Gradients (HOG) [23], Scale Invariant Feature Transform (SIFT) [64], Speeded Up Robust Features (SURF) [7], Local Binary Patterns (LBP) [71], and color histograms [10]. Feature descriptors obtain from these low level features refer to a pattern or distinct structure found in an image, such as a point, edge, or small image patch. 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, just that it is distinct from its surroundings. These low level 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.

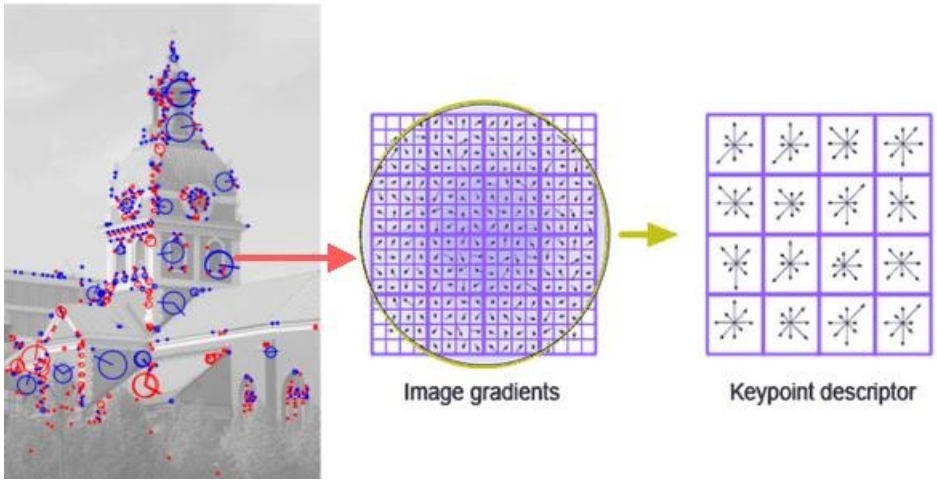


Figure 1.2: Feature extraction using SIFT.

Representation Learning

Representation learning is mainly described by Deep Learning algorithms or Auto Encoders. The ideas is to learn a group of filters that are able to discern one category of images from the another category with some supervised or unsupervised algorithm. Learning representation from an image can start from either low level features (Auto Encoders) or raw pixels of an image (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 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.

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. The first deep CNN that had great success on image recognition is the LeNet proposed by Y.LeCan in 1989 [59]. Back-propagation 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.

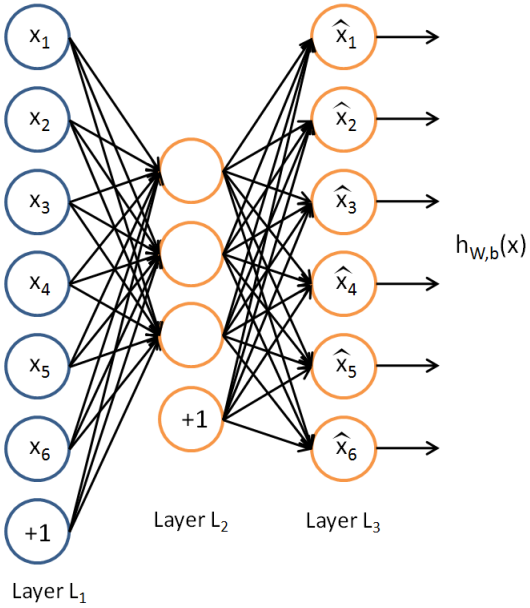


Figure 1.3: General Scheme of Auto Encoders. L1 is the input layer, possibly raw-pixel intensities. L2 is the compressed learned latent representation and L3 is the reconstruction of the given L1 layer from L2 layer. AutoEncoders tries to minimize the difference between L1 and L3 layers with some sparsity constraint.

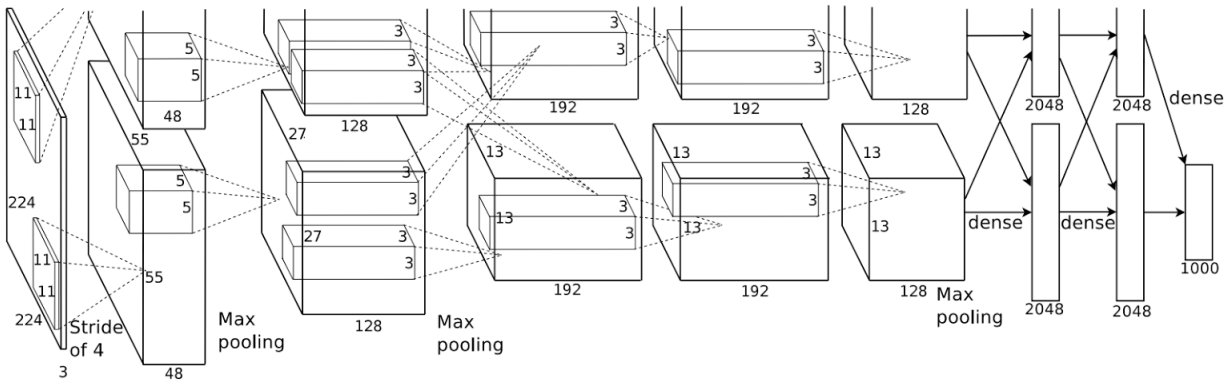


Figure 1.4: The architecture of ALEXNET (adapted from [53]).

Classification

After extracting feature representation from the images, a classifier should be 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 [21]. As we mentioned before, in order to capture different variances of the images for one category, the size of the feature representation for a 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 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 model are more convenient for training and can be effectively optimized with stochastic gradient descent which is suitable for very large training set [12].

1.2 Our Scenario (to be refined)

1.2.1 Importance of learning self-defined categories

Nowadays, more and more companies provide individual service for their clients. Personalized recommendation system has been widely used in many electronic commerce, such as Amazon and eBay. The requirement of personalized service is growing every year. Personalized system means it is possible to handle the variance of the request from individuals. For image recognition, many interesting applications have been proposed. However, unlike other personalized system, it is difficult to train personalized recognition system for individuals. There could be enormous variances of one object and it is always difficult for a model to capture these variances.

Even though, the state-of-the-art recognition system can do as well as a human, its great success is achieved based on the fact that millions of labeled training images are used for training. Selecting large amount of images for the new categories is always a tedious job. Moreover, an important property of self-defined category images is that the source of the data is inherently scarce. Therefore, it is not possible to obtain abundant training data. For example, users of Google's im2calories can track their nutrition of every meal by taking a pictures of their foods via image recognition techniques. The system firstly check the category of each food item and find their nutritions in the database. However, the existing model can only recognize the general food categories which means it is not possible for a user to track the nutrition of his/her daily home-made meals. These home-made foods can be similar to some

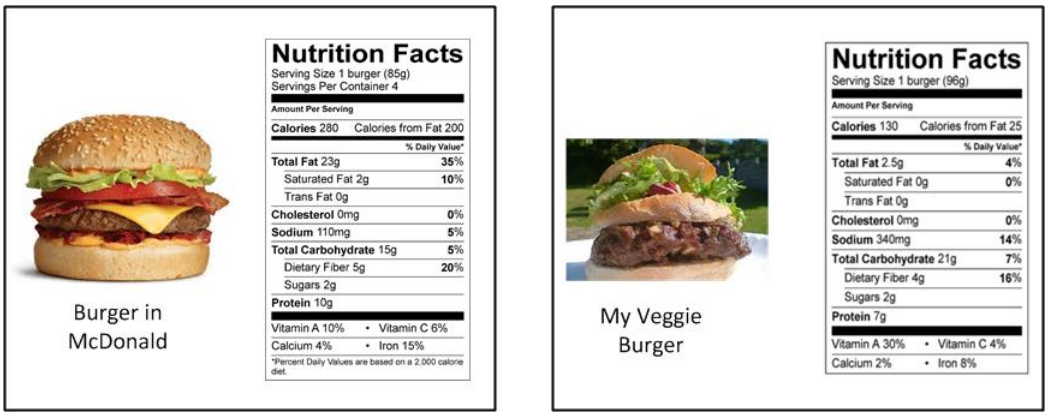


Figure 1.5: Different nutrition facts between the burgers in McDonald and home-made.

existing food categories (home-made veggie burger can be similar to burger in McDonald), but they may have different nutritions. This scenario can be applied to learning any exclusive category in our life. Therefore, a model that can learn these self-defined categories can be important.

Our human is good at learning the new objects. For our human, all the information acquired is stored in our memory. These information are 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 "zebra", we would probably say that: zebra is a horse with while and black striped coats. People who never see a zebra could instantly have a rough idea of a zebra.

This means to learn a new category effectively, we should be able to make use of the gained knowledge instead of learn it from scratch. This process is commonly refer to as transfer learning. Traditional machine learning methods work under the common assumption: training data and testing data are drawn from the same feature space and same distribution. When considering adding classifying the data from the new categories, the distribution of the data changes and a new model should be rebuilt from newly collected data.Those 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 training the new model to classify the target data.

In this thesis, we try to use transfer learning approaches to solve the problem of learning self-defined categories. We can find many examples in knowledge engineering where transfer learning does benefit the learning process [72]. This implies that, by leveraging the learned knowledge properly, we can learn the self-defined categories with a few examples. In this

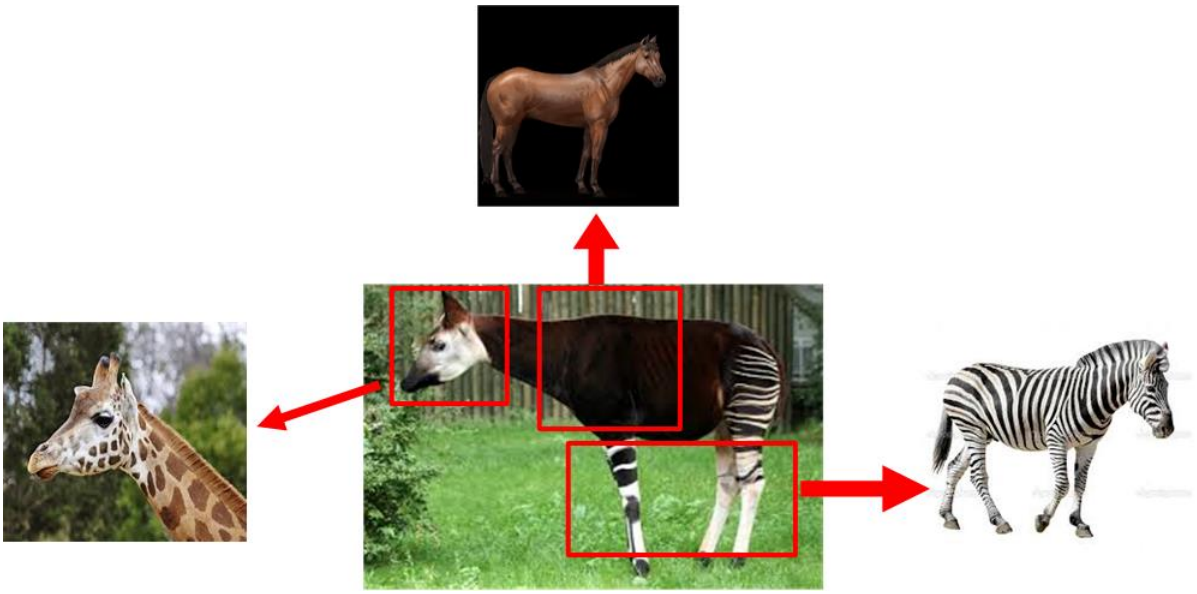


Figure 1.6: Multi-source category case: an okapi can be roughly described as the combination of a body of a horse, legs of the zebra and a head of giraffe.

thesis, we also assume that the self-defined categories are not isolated and similar to certain existed categories. Therefore, we can use leverage the knowledge from those existed categories to help us learn the self-defined categories. Then, we split the self-defined categories into two groups according to their relationship to the exited categories:

- One source category, a category that is very similar to one existed category. For example, my veggie burger can be very similar to general burger and less similar to other categories.
- Multi-source category, a category that doesn't have an explicit similarity to one category but share some properties with several existed categories (see figure 1.6 for instance).

Due to different properties of these two groups, we should design different strategies to adapt them.

1.2.2 Assumption of the source knowledge

In transfer learning, the source knowledge can be presented in two different ways [72]:

- Instance transfer learning. Even though the source data can not be re-used directly, certain part of the data can be used incorporating with a few labeled target data to train the model.

- Parameter transfer learning. Instead of utilizing the source data, parameter transfer learning approaches re-use the parameters of the model trained from source data (called the source model).

In this thesis, we try to explore a method to learn self-defined categories via the parameter transfer learning approach and try to utilizing the source data in a hard way, by assuming that the source data is access prohibited and we can only access the trained model from the source data. This assumption is made based on the following two facts: (1) In some situations, we may not be able to access the source data and only the model trained from the source data is available. There are many credential datasets. Therefore, it is not always possible to fully access the source data. (2) The source data could be very large and it could be tedious to determine which part of the data could benefit the transfer learning. On the other hand, the trained model from the source data can be as informative as the source data itself. For example, the information extracted from the support vectors (SVs) of a SVM model trained from a dataset could be as much as the information of the whole dataset. Some results from recent work also show that it is possible to obtain a good model by even just utilize the source models and learn new categories from a few examples [35] [90] [91].

In this section, we discuss the scenario of the learning self-defined category problem. We conclude that learning self-defined categories can be achieved by transfer learning. We also make an assumption that we can only access the model trained from the source data and the source data itself is prohibited. In the next section, we discuss the challenges of our problem.

1.3 Challeges

In this section, we discuss several major challenges that we may encounter when solving the learning self-defined category problem. To solve the learning self-defined category problem, there are the following major problems:

1. What's the classifier and parameter we should use to train the model that can transfer the knowledge from the source data effectively?

2. How are the knowledges transferred from the source task to target one and control the amount of the knowledge that should be transferred?

3. How to guarantee the result of transfer model?

The first challenge is to determine the classifier we use for the source data as well as the target data. There are two criterion for us to choose the classifier: (1) This classifier should be

popular and has been used in many tasks. We would expect our method to be general enough and can be applied to many different scenarios. (2) Because we can only access the trained source model, this source model should be able to restore as much information as its training data to make sure that there is enough knowledge to be leveraged for the transferred model. Then we have to choose the parameters of the model to be used for transfer learning. As we discussed above, the parameter should be informative enough to represent the source data.

The second challenge of out problem is to determine the way of the knowledge transferred from source model to target one. To learn a new category by transfer learning, according to our human experience, we would firstly select several known categories that are similar to the new one and then describe the new category as a combination of these known ones (see figure 1.6). It is worthy to note that despite of the knowledge from the source models, we will also extract some new knowledge from the examples of the new category. This indicates that how to combine those knowledge from learned categories (e.g. the parameters of the source models) and the knowledge from the new category is the key for our task.

The last challenge is to guarantee the performance of our transfer method. A baseline criterion is that after leveraging the knowledge from the source model, the performance of the transfer model to distinguish the self-defined categories will be improved or at least won't get worse when there is little useful knowledge in the source model. This is an important criterion for transfer learning. We should always expect to leverage knowledge of the source model to help training the target model. However, inappropriately leveraging the knowledge of the source models would decrease the performance of the target model. For example, we could never expect to obtain a good model to distinguish apple and other fruits by relying too much on the knowledge of distinguishing human and animal.

Chapter 2

Related Work

In this chapter, we review some previous work related to ours. In Section we review.....

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, there should always be a recognition model to distinguish the objects from different categories automatically trained by supervised learning.

In supervised 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 recognition [3] [53].

Here, we give a formal definition of the scenario of binary classification and for multi-class scenario, 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)[74] [94].

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

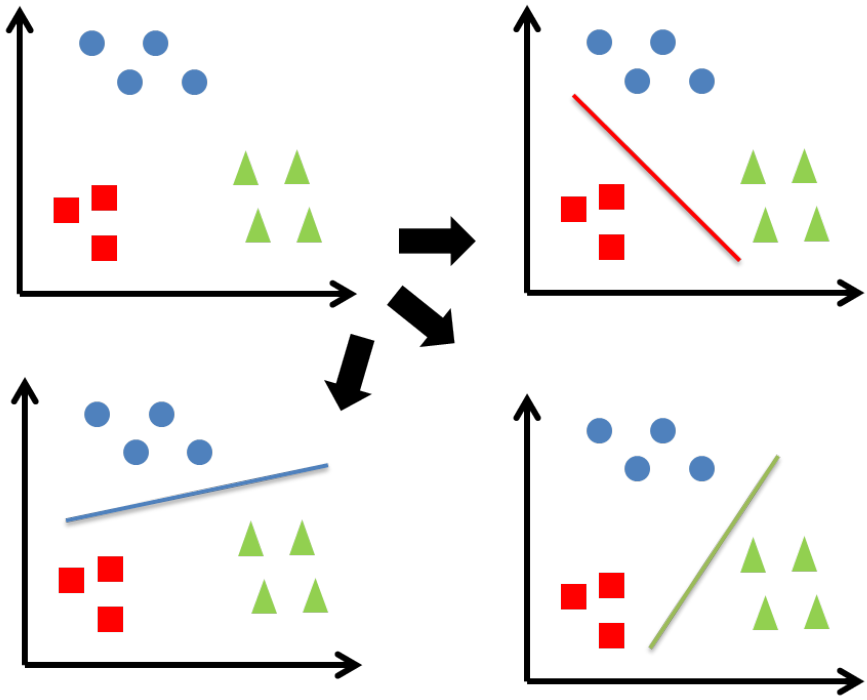


Figure 2.1: One-vs-Rest strategy for multi-class scenario. A three classes problem can be decomposed into 3 binary classification sub-problems.

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 $f : \mathcal{X} \rightarrow \mathcal{Y}$ from the training set D_{train} such that f can achieve good performance on both D_{train} and D_{test}.

In the next subsection, we first review.....

2.1.1 Linear Method: Softmax Classifier

In this subsection, we will introduce a widely used linear classifier softmax classifier. Linear classifier is commonly used as the classification model for image recognition. A 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. 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 work very well when the number of dimensions of the input is large. Therefore, it is widely used as the classifier for image recognition.

Softmax classifier is widely used for image classification especially on Neural Networks and Convolutional Neural Networks [59]. 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:

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

Here $w^{(1)T}, w^{(2)T}, ..., w^{(N)T}$ are the parameters of the softmax classifier model. The term $\frac{1}{\sum_{j=1}^{N} \exp\left(w^{(j)T} x\right)}$ 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:

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

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) [48].

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

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

2.1.2 Kernel Method: Support Vector Machines

In this subsection, we will review another widely used discriminant classifier, Support Vector Machine (SVM) [21]. SVM is another classifier that has been adoptive in many image recognition tasks [19] [81] [102]. 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 [21] [82].

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:

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

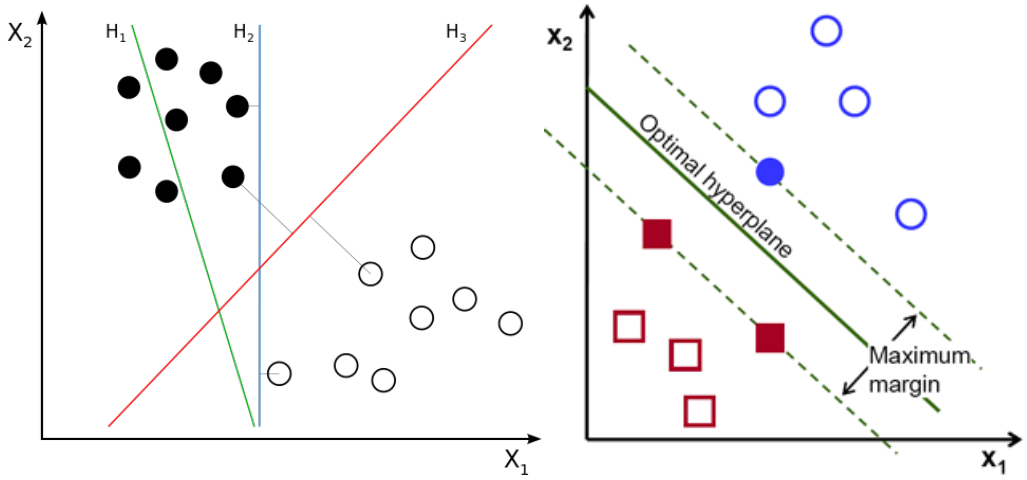
where w is called the weight vector and b is called the bias. In a binary scenario, for the input example x and the class labels $y \in \{c1, c2\}$, x is assigned to class $c1$ if $f(x) \geq 0$ and $c2$ otherwise. Therefore, the corresponding decision surface is defined by $f(x) = 0$. For two points 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:

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

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

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



(a) Different sparating hyperplanes (b) Max-Margin hyperplanes

Figure 2.2: Support Vector Machine

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:

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

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:

$$\xi_i = hinge(x_i) = \max(0, 1 - y_i(w^T x_i + b)) \tag{2.10}$$

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

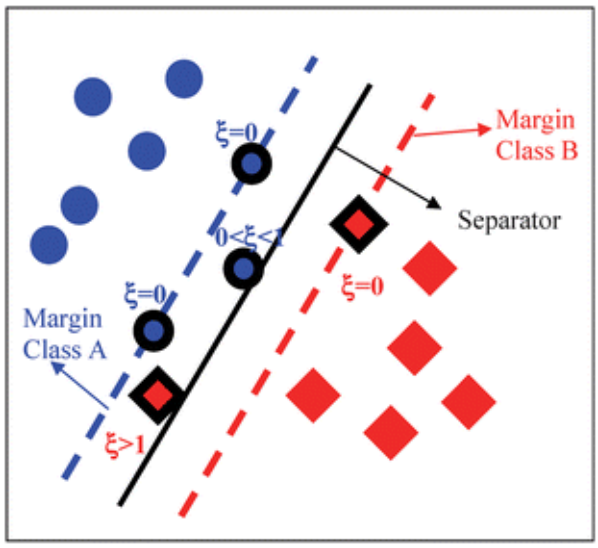


Figure 2.3: Slack variables for soft-margin SVM

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 [82].

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 [51].

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 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:

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

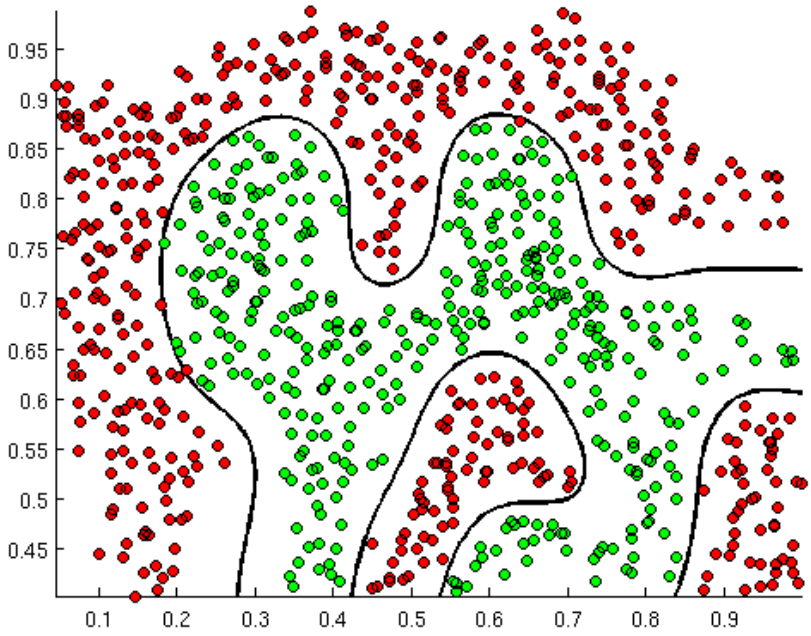


Figure 2.4: The hyperplane of SVM with RBF kernel for non-linear separable data.

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:

$$\begin{aligned} \max \quad & \sum_i^n \alpha_i - \sum_i^n \sum_j^n y_i y_j \alpha_i \alpha_j \phi(x_i)\phi(x_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\lambda} \\ & \sum_i^n y_i \alpha_i = 0 \quad \text{for all } 1 \leq i \leq n \end{aligned} \tag{2.14}$$

We can obtain the solution of (2.14) by Sequential Minimal Optimization (SMO) [73] or Dual Coordinate Descent [41].

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 [82] or SMO [73].

To summarize, in this section, we have briefly reviewed 2 types of classifier for image recognition, namely softmax classifier and SVM. Softmax classifier is typically used as the last layer of deep neural network. SVM is a more general method method for classification task. Moreover, The generalization ability of SVM classifier can reduce the bias from the training data. Therefore, in our work, we choose SVM as our classifier for image recognition.

2.1.3 Convlutional Neural Networks

Early work with Convlutional Neural Networks

The first simple version of Neural Networks (NNs) trained with supervised learning was proposed in 1960s [77][78]. Networks trained by the Group Method of Data Handling (GMDH) could be the first DL systems of the Feedforward Multilayer Perceptron type [43][80]. Later, there have been many applications of GMDH-style nets [34] [42] [52] [100].

Apart from deep GMDH networks, the Neocognitron, a hierarchical, multilayered artificial neural network, was perhaps the first artificial NN to incorporate the neurophysiological insights [36]. 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 was applied (LeCun et al., 1989, 1990a, 1998) to Convolutional Neural networks with adaptive connections [59]. 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 [6] [58].

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).

Recent achievements with Convlutional Neural Networks

In 2006, CNN trained with backpropagation set a new MNIST record of 0.39% without using unsupervised pre-training [68]. Also in 2006, an early GPU-based CNN implementation was introduced which was up to 4 times faster than CPU-CNNs [17]. 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 [18].

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 [53]. 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 system, 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 Fergus improve AlexNet by 1.7% on top 5 accuracy [107]. 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 [84]. After the AlexNet-like deep CNNs won ILSVRC2012 and ILSVRC2013, Szegedy et al. built a 22 layers deep network, called GoogLeNet and won the 1st prize on ILSVRC2014 for 93.33% top-5 accuracy, almost as good as human annotation[88]. Different from AlexNet-like architecture, GoogLeNet shows another trend of design, utilizing many 1×1 receptive field. Recently, Wu et. al present a image recognition system by aggressive data augmentation on the training data,

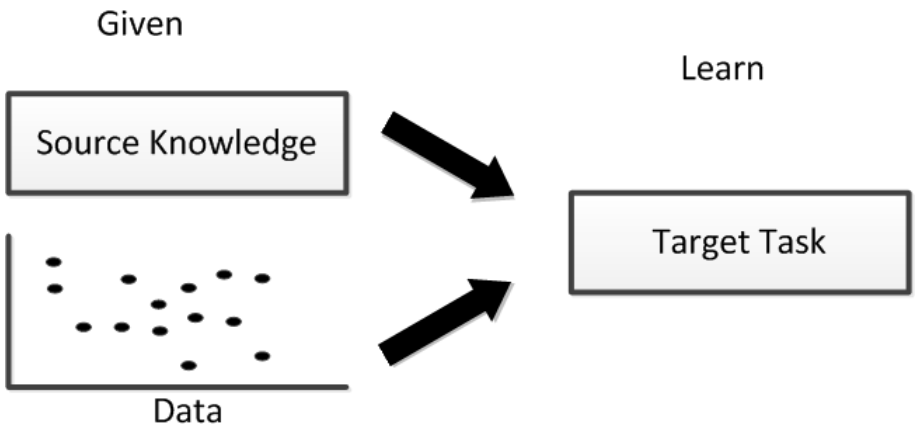


Figure 2.5: Apart from the standard machine learning, transfer learning can leverage the information from an additional source: knoweldge from one or more related tasks.

achieving a top-5 error rate of 5.33% on ImageNet dataset[101]. Searchers from Google successfully trained an ingredient detector system based on GoogLeNet with 220 million images harvested from Google Images and Flickr [67].

Besides its impressive performance on those huge dataset, 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% [107]. Chatfield et al used their pre-trained model on VOC2007 dataset and outperformed the previous state-of-the-art by 0.9% [16]. 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 [108]. Hoffman et al. fine-tuned a 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 [40].

2.2 Visual Transfer Learning without the source data

As we mentioned before, we use the transfer learning method to learn the self-defined categories. In this section, we review some problems transfer learning faces and the popular methods to learn new categories in 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).

Transfer learning can be categorized into 3 sub-settings: *inductive transfer learning, transductive transfer leanring* and *unsupervised transfer learning* based on the different situations of the source and target domains and tasks. [72]. We compared the differences of these three sub-categories and show them in Table 2.2. Recalling our learning self-defined category problem, we already obtained the source models from labelled source data and our goal is to utilize the knowledge from the source models and help us to train the models from a small set of labelled examples, i.e. examples from our self-defined categories. From Table 2.1 and 2.2 we can see that, our problem should be classified as inductive transfer learning. More specifically, it belongs to the multi-task transfer learning.

Table 2.1: 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 2.2: 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	unlabeled
Unsupervised transfer learning		unlabeled	unlabeled

2.2.1 Inductive Transfer Learning

In inductive transfer learning, we try to utilize the knowledge from the source task to help learning the target task. Here we give a formal definition of inductive transfer learning [72]:

Definition (Inductive Transfer Learning) Given a source domain D_s and the source learning task T_s, a target domian D_t and the target learning task T_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 and T_s where $T_s \neq T_t$.

According to the type of the source knowledge comes from, inductive 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. [22] 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. [46] propose 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. [61] propose 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. [8] provide 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. [31] propose a method that can learn sparse low-dimension feature representations that can share between different tasks. Jie et al. [47] reconstruct 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 [24] proposes a simple feature reconstruction method for both source and target data so that source and target data are triple argumented 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 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

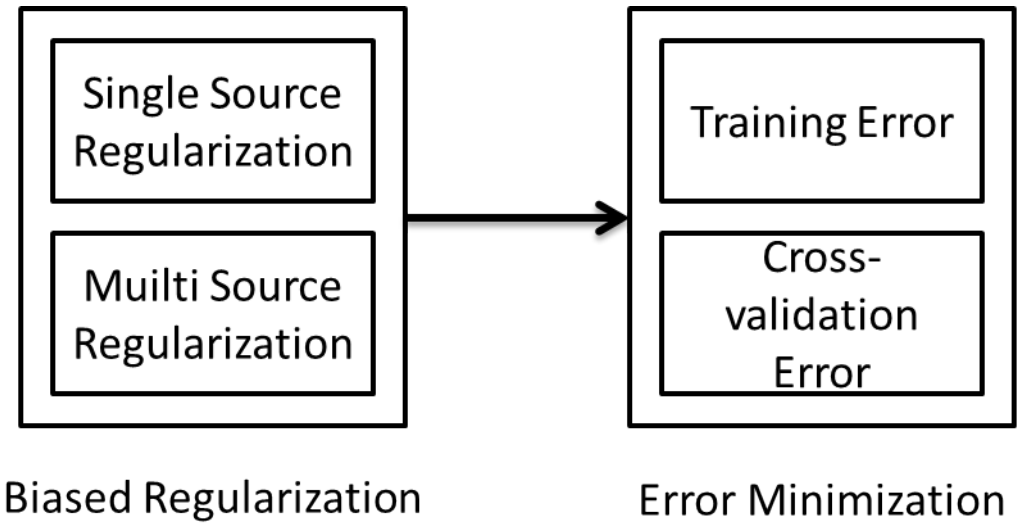


Figure 2.6: Two steps for parameter transfer learning. In the first step multi-source and single source combination are usually used to generate the regularazation term. The hyperplane for the transfer model can be obtained by either minimizing training error or cross-validation error on the target training data.

target task should be related to the hyperplane of the source models. Evgeniou et al. [32] propose 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 [3] [90] [104]. 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 build. Then another objective function is build 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 bias when the target data is small. Yosinski et al. [106] 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 [16] [40] [107] [108].

- 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. [35] use 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. [79] use 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.

2.2.2 Fine-tuning the Deep Net

2.2.3 Hypothesis Transfer Learning

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 visiting the source data. We first introduce the basic principle of LS-SVM. Based on 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.

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 [87]. By replacing the hinge loss in classical SVM with L2 loss, LS-SVM classifier is obtained by reformulating the minimization problem as:

$$\begin{aligned} \min \quad & L_{LSSVM} = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}\varepsilon_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{2.15}$$

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

$$L(w, b, \alpha, \varepsilon) = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}{\varepsilon_i}^2 - \sum_{i=1}^{l}\alpha_i\{wx_i + b + \varepsilon_i - y_i\} \tag{2.16}$$

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

$$\begin{bmatrix} K + \frac{1}{C}\mathrm{I} & 1 \\ 1^T & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{2.17}$$

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

$$\psi^{-1} = \begin{bmatrix} K + \frac{1}{C}\mathrm{I} & 1 \\ 1^T & 0 \end{bmatrix}^{-1} \tag{2.18}$$

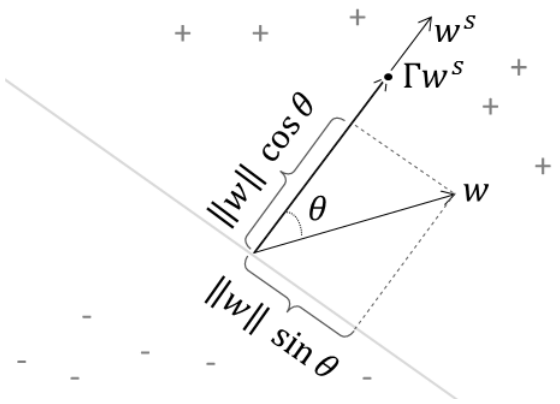


Figure 2.7: Projecting w to w' in PMT-SVM (adapted from [3]).

Problem (2.16) can be solved by:

$$\begin{bmatrix} \alpha \\ b \end{bmatrix} = \psi^{-1} \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{2.19}$$

ASVM & PMT-SVM

Adaptive SVM (ASVM) is the first work using LS-SVM for transfer learning for vision related tasks [103]. 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:

$$\begin{aligned} \min \quad & L_{ASVM} = \frac{1}{2}\|w - \gamma w'\|^2 + \frac{C}{2}\sum_{i=1}^{l} \varepsilon_i^2 \\ \text{s.t.} \quad & y_i = w x_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{2.20}$$

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 angel 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 tradeoff 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 [3]:

$$\begin{aligned} \min \quad & L_{PMT} = \frac{1}{2}\|w\|^2 + \gamma\|Pw'\|^2 + \frac{C}{2}\sum_{i=1}^{l}\varepsilon_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \\ & w^T w' \geq 0 \end{aligned} \tag{2.21}$$

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 2.7). 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 [91]. Similar to the objective function (2.20), its objective function is defined as follow:

$$\begin{aligned} \min \quad & L_{Multi-KT} = \frac{1}{2}\left\| w - \sum_k \beta_k w'_k \right\|^2 + \frac{C}{2}\sum_{i=1}^{l}\zeta_i\varepsilon_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{2.22}$$

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 [15], the closed-form LOO error is defined as:

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

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:

$$\begin{aligned} \min \qquad & \mathcal{L} = \sum_{i}^{l} \zeta_i |1 - y_i \hat{y}_i|_+ \\ \text{s.t.} \qquad & \|\beta\| \leq 1 \end{aligned} \tag{2.25}$$

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.

MULTIpLE

MULticlass Transfer Incremental LEarning (MULTIpLE) focuses on adding a new class to a existing N class source problem while preserving the performance on the old classes [57]. 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:

$$\begin{aligned} \min \qquad & L_{MULTIpLE} = \frac{1}{2}\sum_{n=1}^{N} \|w_n - w'_n\|^2 + \frac{1}{2}\left\| w_{N+1} - \sum_{k=1}^{N} w'_k \beta_k \right\|^2 + \frac{C}{2}\sum_{n=1}^{N+1}\sum_{i=1}^{l} \varepsilon_{i,n}^2 \\ \text{s.t.} \qquad & \varepsilon_{i,n} = Y_{in} - x_i w_n - b_n \end{aligned} \tag{2.26}$$

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 [20]:

$$\begin{aligned} \min \qquad & \mathcal{L}(\beta, i) = \begin{cases} \max\limits_{n \neq y_i} \left|1 - \hat{Y}_{in}(\beta) - \hat{Y}_{iy_i}(\beta)\right|_+ & : y_i \neq N+1 \\ \left|1 - \hat{Y}_{in}(\beta) - \hat{Y}_{iy_i}(\beta)\right| & : y_i = N+1 \end{cases} \\ \text{s.t.} \qquad & \|\beta\| \leq 1 \end{aligned} \tag{2.27}$$

We can find the optimal β with projected subgradient descent [14].

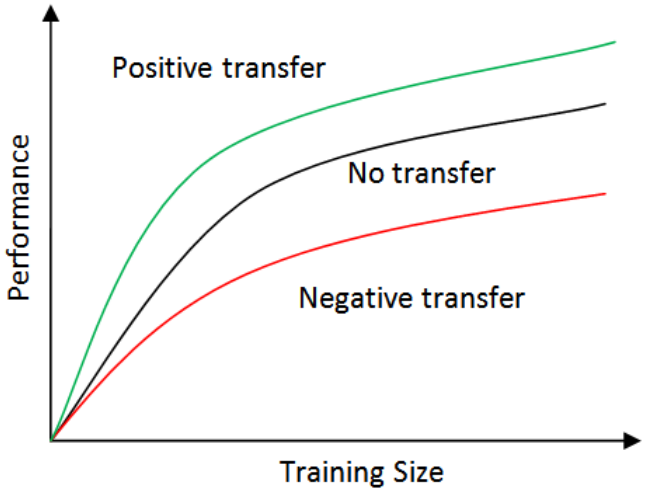


Figure 2.8: Positive transfer VS Negative transfer.

2.2.4 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 [72]. 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 negative transfer [92]. Even though voiding negative transfer is an important issue in transfer

learning, how to avoid negative transfer has not been widely addressed [65] [72]. Previous work show suggest that negative transfer can be alleviated through 3 approaches [92]:

- 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. [93] propose a method using advice-taking algorithm to reject the unrelated source knowledge. Rosenstein et al. [79] present 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. [89] propose 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. [55] construct 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. [5] propose 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. [91] construct 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 [47] [57]. Kuzborskij et al. [56] provide 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 [90] [56] 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 reviewed 3 types of inductive transfer learning approaches and discussed how to avoid the negative transfer in transfer learning. In our problem, as we have the assumption that the source data is access prohibited, we are not able to use instance transfer and feature transfer approaches. Therefore, we propose a transfer method based on parameter transfer approach. To reduce the affect negative transfer, in our transfer method, we adopt several ideas from the work mentioned above.

2.3 Summary

Chapter 3

Learning Food Recognition Model with Deep Representation

3.1 Introduction

3.2 CNN layers:conv/pool/norm etc

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is a $m \times m \times r$ image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has $r = 3$.

3.2.1 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.

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 3.1). 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,

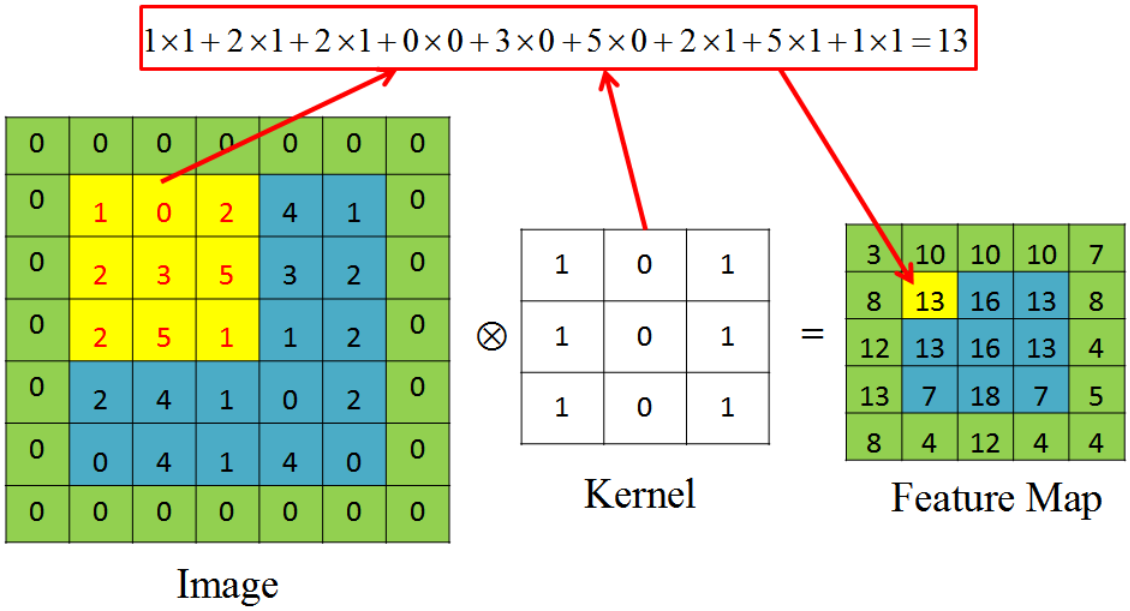


Figure 3.1: Convolution operation with 3×3 kernel, stride 1 and padding 1. $\otimes$ denotes the convolutional operator.

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 [53] [60] [84] [107]. Recent development of tiny 1×1 kernel shows an improvement on both accuracy and computational efficiency [88].

3.2.2 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 convolutional layer for classification would be a computationally 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, this results in a vector of $892 \times 400 = 3,168,400$ features per example. Learning a classifier from over 3 millions 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,

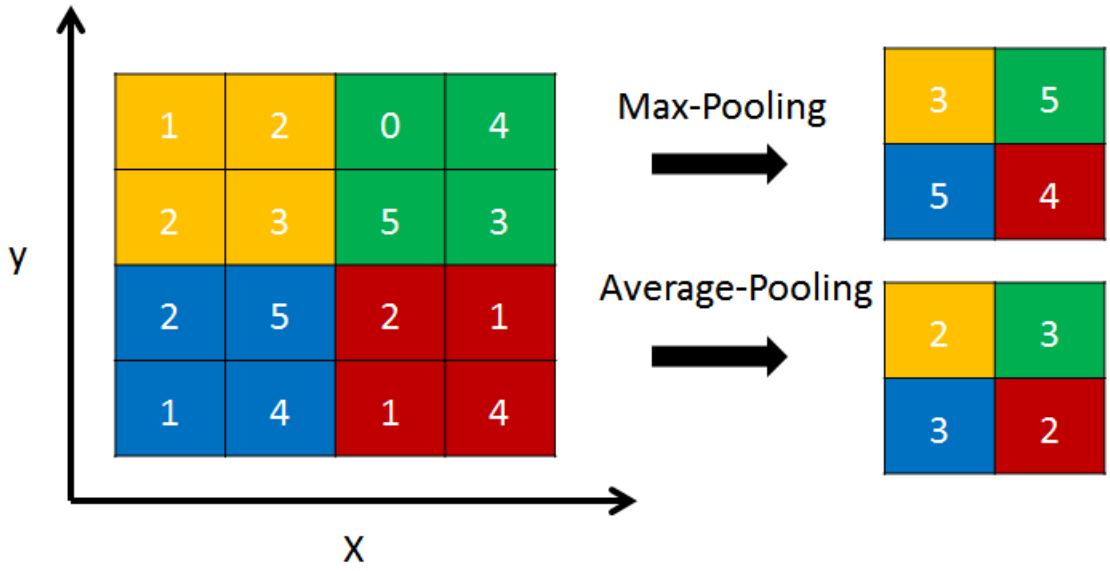


Figure 3.2: 2×2 pooling layer with stride 2 and padding 0.

and hence to also control overfitting. The pooling layer works independently on the channel dimension and resize 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.

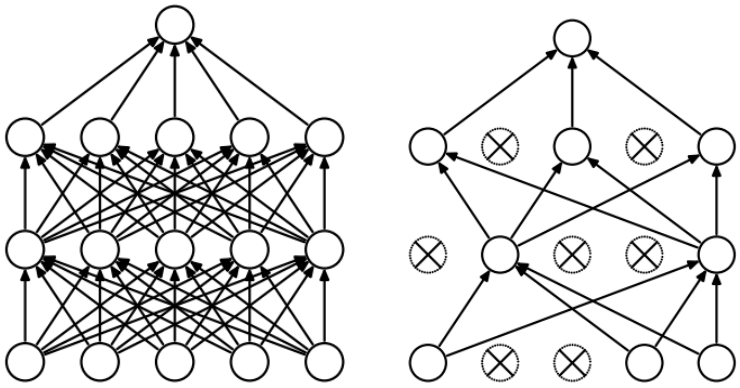
In general, two kinds of pooling strategy, Max Pooling and Average Pooling, are commonly used in CNN architecture (see Figure 3.2). Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice [67] [88]. Max Pooling is been widely used in all kinds CNN architectures [13] [102].

3.2.3 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 [39] [70].

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



(a) Standard Neural Net (b) After Dropout

negative biases. This replacing procedure can be mathematically presented as:

$$\sum_{i}^{N} \sigma(x - i + 0.5) \approx \log(1 + e^{x}) \tag{3.1}$$

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

$$f(x) = \log(1 + e^{x}) \approx \max(x, 0) \tag{3.2}$$

as the activation function for approximation [44]. 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.

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 to randomly drop units (along with their connections) from the neural network during training [86]. 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.

3.3 Datasets

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

3.3.1 Models

In this paper, AlexNet and GoogLeNet are their Caffe [45] implementations and all the results for a specific CNN architecture are obtained from 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 [86] layer are followed.

GoogLeNet shows another trend of deep CNN architecture with lots of small receptive fields. There are 9 Inception modules in GoogLeNet and Figure 3.3 shows the architecture of a single inception module. Inspired by [62], 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 are counted into the total loss with a discount weight 0.3, in addition with the loss of the classifier on top. More architecture details can be found from [88].

3.3.2 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 a food varies depending on many aspects such as the point of the view or even its components.

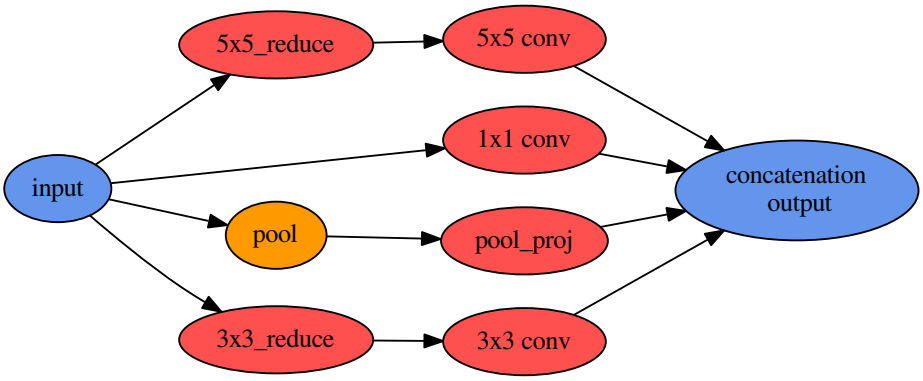


Figure 3.3: Inception Module. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 convolutional layer before the $n \times n$ convolution layer and $pool\_proj$ is another 1×1 convolutional layer after the MAX pooling layer. The output layer concatenates all its input layers.

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 [49][1] and Food-101 [11][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

[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

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.

3.3.3 Data Argumentation

In this section, we introduce some data argumentation methods in our work to enrich our training data. Previous work shows that with intensive argumentation for the training data, the performance of CNN model can be improved [101]. Data argumentation is an efficient way to enrich the data. There are also some techniques that can applied 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 3.4).

Before cropping subsamples from the original image, we also use other argumentation 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 3.5).

Compared to color shifting in [53], 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 argumentation. 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.

3.4 Experimental Discuss

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 preventing this problem in certain degree. Compared to other randomly initialized strategies with certain distribution, supervised pre-training is to initialize the weights according to the model

(a) Original image

(b) Center

(c) Center mirror

(d) Up-left

(e) Up-left mirror

(f) Up-right

(g) Up-right mirror

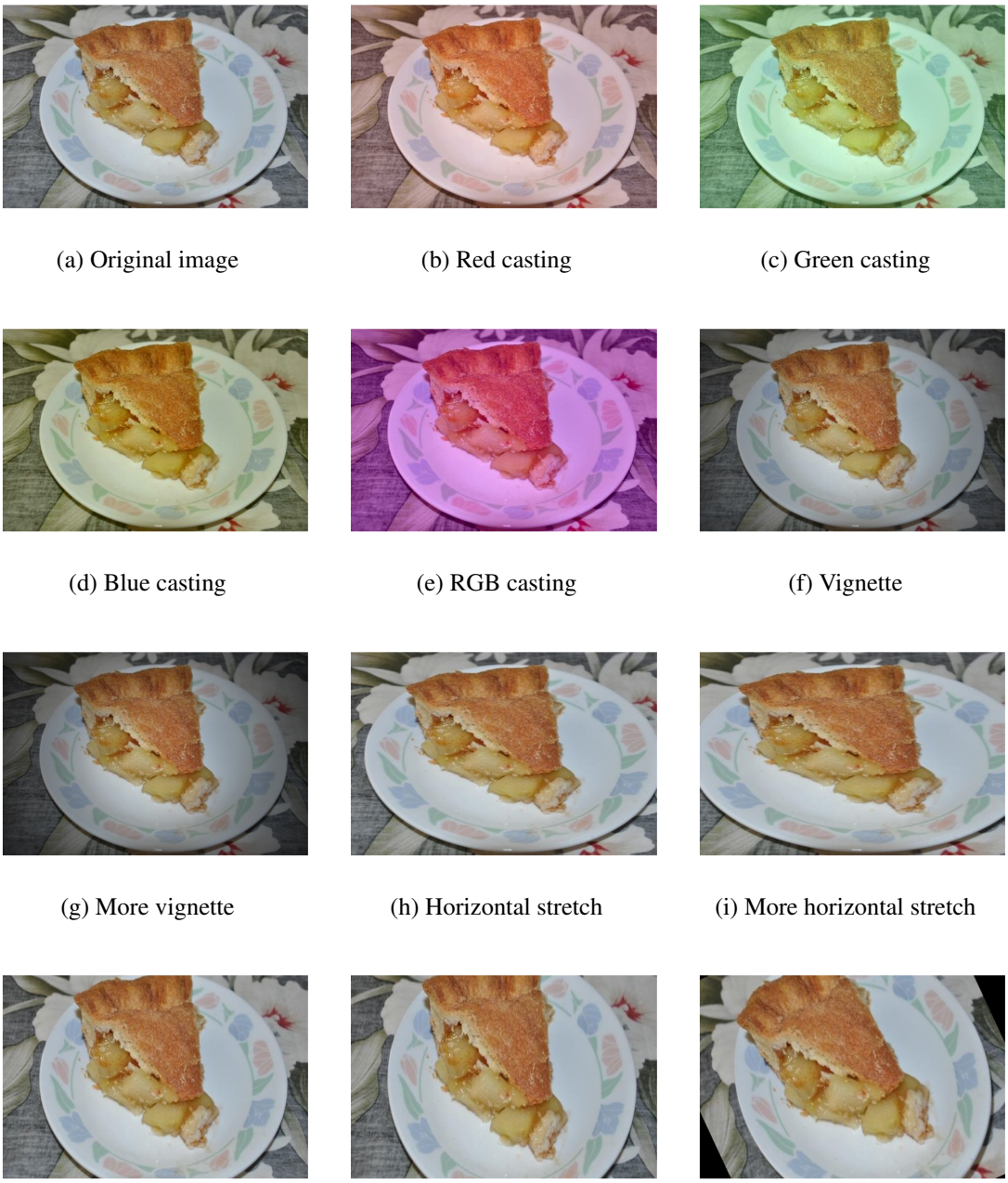
(h) Bottom-left

(i) Bottom-left mirror

(j) Bottom-right

(k) Bottom-right mirror

Figure 3.4: Crop area from original image



(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 3.5: Different data argumentation methods

trained from a specific task. Indeed, initialization with 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 be benefit from it.

3.4.1 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[37]. 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 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 3.1: 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 3.2: Accuracy compared to other method on Food-256 dataset in percent

	fv+linear [50]	GoogLeNet	AlexNet
Top1	50.1	**70.13**	63.82
Top5	74.4	**90.66**	85.59

Table 3.3: Top-1 accuracy compared to other methods on Food-101 dataset in percent

	RFDC[11]	MLDS(≈[85])	GoogLeNet	AlexNet
Top1 accuracy	50.76	42.63	**78.11**	66.40

From Table 3.1 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

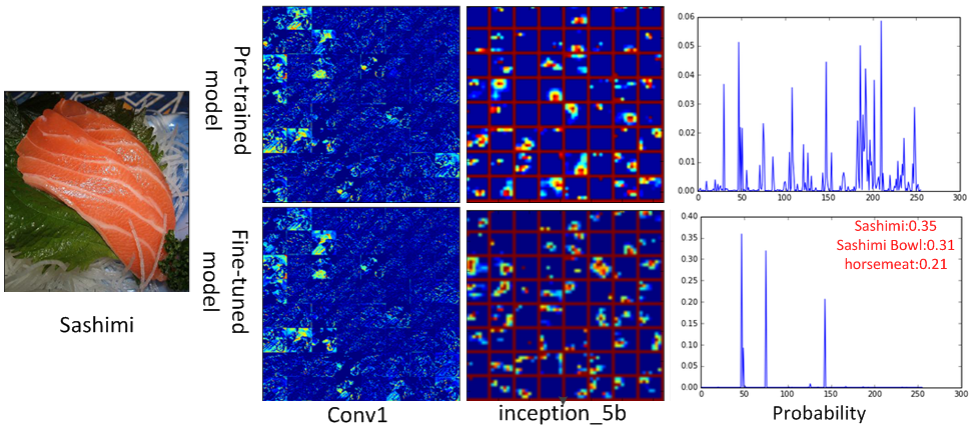


Figure 3.6: Visualization of some feature maps of different GoogLeNet models in different layers for the same input image. 64 feature maps of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

3.2 and 3.3), 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 3.6 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 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 3.1, 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 3.1 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 3.4 and 3.5 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 3.5, 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:

$$h = \max(w^T x, 0) = \begin{cases} w^T x & w^T x > 0 \\ 0 & else \end{cases} \tag{3.3}$$

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 3.3). 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 a dense outputs for the following layer. We can see from Table 3.6 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 previous layer can be squeezed with the 1×1 convolutional layers and feed to convolutional layers with 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 recogntion tasks.

3.4.2 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 on 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 [108]. 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.

Table 3.4: 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 3.5: 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 3.6: 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

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 a our own 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.

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

The Food-101 and Food-256 datasets share about 46 categories of food even though the 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 subset for Food-256 on different pre-trained model and the results are shown in Table 3.7. 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 3.1) as the pre-trained model.

From the result of Table 3.7 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 general task can still achieve 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.

3.5 Summary

Chapter 4

Learning Food Recognition Model with Deep Representation

4.1 Introduction

4.2 Using the Source Knowlege as the Auxiliary Bias

Some previous work such as MKTL[47] 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 a 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 own decisions as well (probabilities from the source models). Their decisions can provide extra information regarding this example as the auxiliary bias and adjust

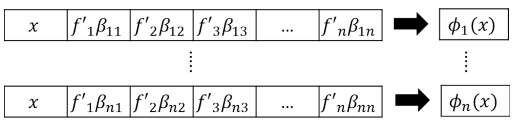


Figure 4.1: Illustration of feature augmentation in MKTL. f'_i is the output of the i-th source model and β_{in} is the hyperparameter (need to be estimated) to weigh the augmented feature. $\phi_n(x)$ is augmented feature for the n-th binary model.

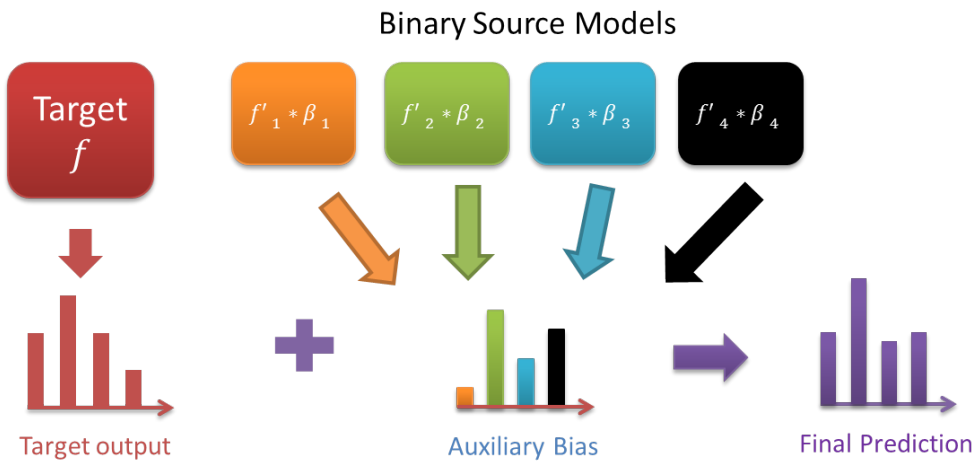


Figure 4.2: Demonstration of using the source class probability as the auxiliary bias to adjust the output of the target model.

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 (see Figure 4.2).

Unlike previous work[3, 91, 103] 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[47], we only have to estimate N hyperparameters for the N-class problem while there are $N \times N$ hyperparameters in MKTL (see Figure 4.1). 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 an effective 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.

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:

$$f = \arg\min_{f \in \mathcal{F}} \ell\left(f + \beta f' | D, \beta\right) \tag{4.1}$$

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. (4.1) 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.(4.1) 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.

4.3 Bi-level Optimization for Transfer Parameter Estimation

As we discussed before, the transfer parameter in Eq. (4.1) is a hyperparameter that cannot be solved directly. Here we use bi-level optimization (**BO**)[**?**], 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. 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:

$$\begin{aligned} &\text{High level} \qquad \beta = \arg\min_{\beta} \sum_{i}^{K} \mathcal{L}(f^i(\beta)|D_i^{val}) \\ &\text{Low level} \qquad f^i(\beta) = \arg\min_{f \in \mathcal{F}} \ell\left(f + \beta f' | D_i^{tr}, \beta\right) \end{aligned} \tag{4.2}$$

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.(4.2) 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 [56] suggests that LOOCV can increase the robustness of the estimated hyperparameter especially on the small dataset. In previous studies[**?**, **?**], BO is a non-convex problem and can only obtain the approximate solution. However, we will show that problem (4.2) is strongly convex and we are able to obtain its optimal solution.

4.3.1 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. (4.2), 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 $D^{\setminus i} = D \setminus \{x_i, y_i\}$. 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:

$$\begin{aligned} \text{Low-level:} \quad f^{\setminus i}(\beta) : \min_{w,b} & \sum_{n}^{N} \frac{1}{2}\|w_n\|^2 + \frac{C}{2}\sum_{j} e_{jn}^2 \\ \text{s.t.} \quad & f_n(x) = w_n x + b_n; \quad x_j \in D^{\setminus i} \\ & e_{jn} = Y_{jn} - f_n(x_j) - \beta_n f_n'(x_j) \end{aligned} \tag{4.3}$$

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 (4.3) is that it can provide an unbiased closed form Leave-one-out error estimation for each binary model f_n[15]. 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.(4.3). We have:

$$\psi = \left[K(X, X) + \frac{1}{C}\mathrm{I}\right] \tag{4.4}$$

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 $f_n^{\setminus i}$ for sample x_i, can be written as[15]:

$$\hat{Y}_{in} = Y_{in} - \frac{\alpha_{in}}{\psi_{ii}^{-1}} \quad \text{for} \quad n = 1, ..., N \tag{4.5}$$

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

$$\boldsymbol{\alpha} = \psi^{-1} Y - \psi^{-1} f'(X) diag(\boldsymbol{\beta}) \tag{4.6}$$

4.3.2 High-level optimization problem

For the high level optimization problem, we use multi-class hinge loss [20] with ℓ_2 penalty in our objective function.

$$\begin{aligned} \text{High-level:} \quad \beta : \min & \frac{\lambda}{2}\sum_{n}^{N} \|\beta_n\|^2 + \sum_{i} \xi_i \\ \text{s.t.} \quad & 1 - \varepsilon_{ny_i} + \hat{Y}_{in} - \hat{Y}_{iy_i} \leq \xi_i \end{aligned} \tag{4.7}$$

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 [57, 91] 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 (4.7) 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: $\beta = \left\{\beta^1, ..., \beta^n\right\}$

$\beta^0 = 1$,$\alpha' = \psi^{-1}Y, \alpha'' = \psi^{-1}f'$

for $t = 1$ to T **do**

  $\hat{Y} \leftarrow Y - \left(\psi^{-1} \circ I\right)^{-1} (\alpha' - \alpha'' diag(\beta))$

  for $i = 1$ to l **do**

    $\Delta_\beta = \lambda\beta$

    $l_{ir} = \max(1 - \varepsilon_{y_i r} + \hat{Y}_{ir} - \hat{Y}_{iy_i})$

    if $l_{ir} > 0$ **then**

      $\Delta_\beta^{y_i} \leftarrow \Delta_\beta^{y_i} - \frac{\alpha''_{iy_i}}{\psi_{ii}^{-1}}, \Delta_\beta^r \leftarrow \Delta_\beta^r + \frac{\alpha''_{ir}}{\psi_{ii}^{-1}}$

    end if

  end for

  $\beta^t \leftarrow \beta^{(t-1)} - \frac{\Delta_\beta}{\lambda \times t}$

end for

4.4 Experiments

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

4.4.1 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 examples are

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

extracted using AlexNet[53]. We compare our algorithm EMTLe with two kinds of baselines.

Table 4.1: Statistics of the datasets and subsets

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

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[47]** and **Multi-KT[91]**. 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 own convex objective function without the ℓ_2 penalty terms. We use linear kernel for all methods in all our experiments.

4.4.2 Transfer from Single Source Domain

In this subsection, following the experiment protocol in [47, 91] for 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. (4.7):

$$\lambda = 2e^{err_n - err_s} \tag{4.8}$$

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 4.3.

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.

4.4.3 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 4.2. Similar to our previous experiment configurations, we only use Caltech and Amazon as the target domains. We show the experiment results in Figure 4.4.

Table 4.2: The selected classes of the two source domains and the classifier type of the source model.

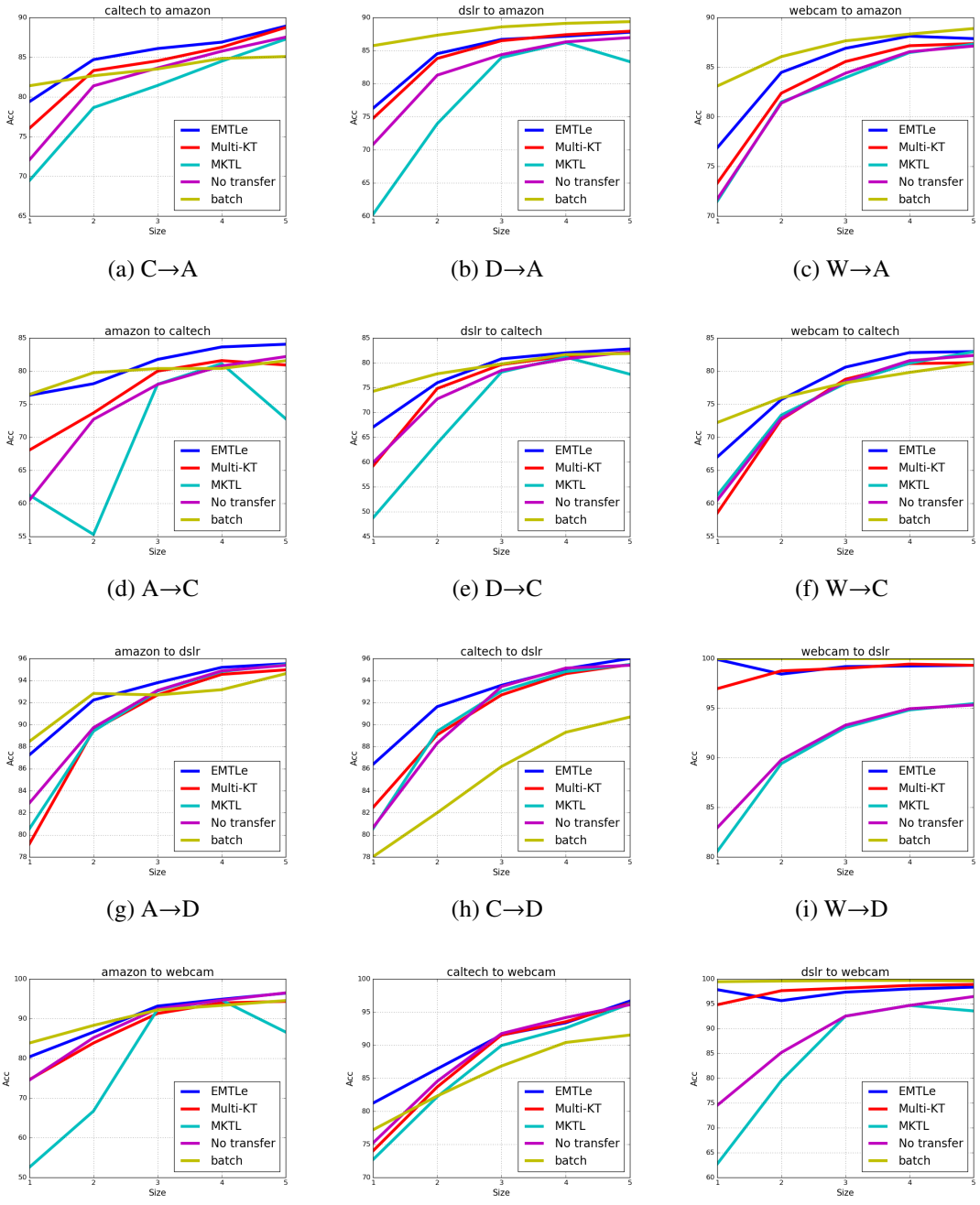
	class	classifier
DSRL	monitor,bike, helmet,calcu,headphone,projector	Logistic
Webcam	keyboard,mouse,phone,backpack,mug,bottle,laptop	SVMs

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.

4.5 Conclusion

In this paper, 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.



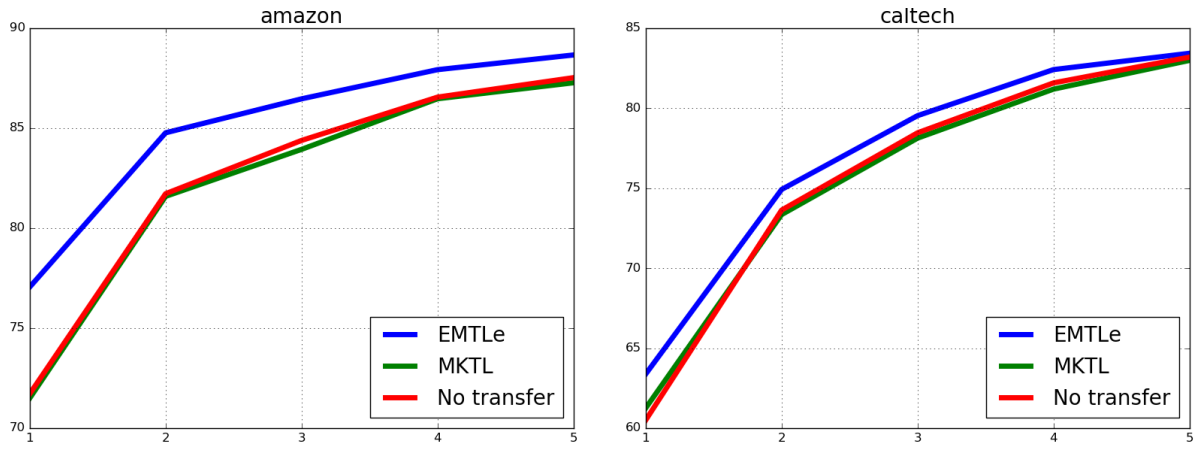
(a) C→A (b) D→A (c) W→A

(d) A→C (e) D→C (f) W→C

(g) A→D (h) C→D (i) W→D

(j) A→W (k) C→W (l) D→W

Figure 4.3: Recognition accuracy for HTL domain adaptation from a single source. 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 4.1 respectively.



(a) D+W → A

(b) D+W → C

Figure 4.4: Recognition Accuracy for Multi-Model & Multi-Source experiment on two target datasets.

Chapter 5

Fast Generalized Distillation for Semi-supervised Domain Adaptation

5.1 Introduction

5.2 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. [105] introduced a framework named Semi-supervised Domain Adaptation with Subspace Learning (SDASL) to correct data distribution mismatch and leverage unlabeled data. [27] proposed a framework for adapting classifiers by "borrowing" the source data to the target domain using a combination of available labeled and unlabeled examples. [25] show that augmenting the feature space of the data can compensate the domain shift. [28] 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. [83] proposed a Rank Transfer method that uses attributes, annotator rationales, object bounding boxes, and textual descriptions as the privileged information for object recognition. [69] 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. [95] proposed a CNN architecture for domain adaptation to leverage the knowledge from limited or no labeled data using the soft label. [96] 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, [66] used ℓ_2

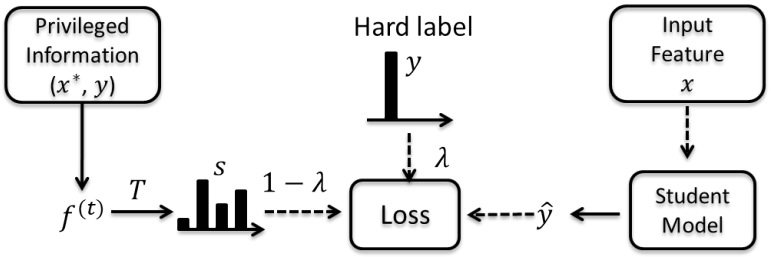


Figure 5.1: Illustration of Generalized Distillation training process.

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 brutal force search or domain knowledge.

5.3 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.

5.3.1 An overview of Generalized Distillation and GDSDA

Distillation [38] and *Learning Using Privileged Information* (LUPI) [98] 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, [63] 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:

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

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).

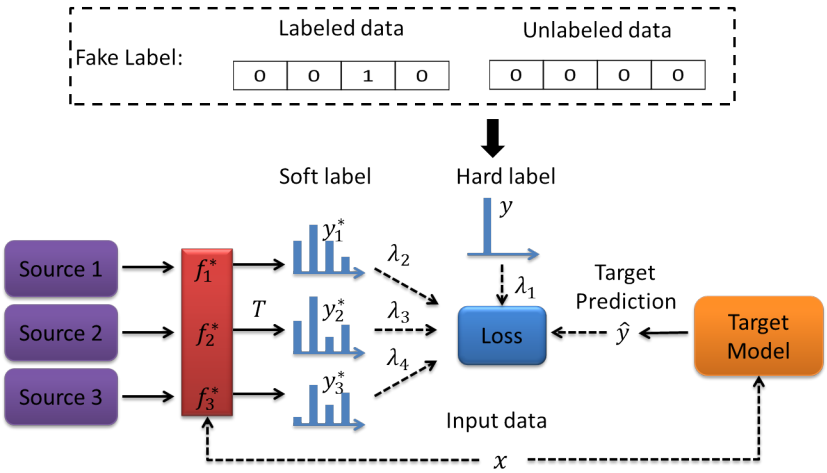


Figure 5.2: Illustration of GDSDA training process and our "fake label" strategy.

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:

$$s_i = \sigma(f^{(t)}(x_i)/T) \tag{5.1}$$

where T is a parameter called temperature to control the smoothness of the soft label. In step 3, learn the student $f^{(s)}$ from the pairs $\{(x_i, y_i), (x_i, s_i)\}_{i=1}^n$ using:

$$f^{(s)} = \arg\min_{f^{(s)} \in \mathcal{F}^{(s)}} \frac{1}{n} \sum_{i=1}^{n} \left[\lambda \ell \left(y_i, \sigma(f^{(s)}(x_i)) \right) + (1-\lambda) \ell \left(s_i, \sigma(f^{(s)}(x_i)) \right) \right] \tag{5.2}$$

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. (5.2), 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[63] 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 5.2). 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 (5.3):

$$\begin{aligned} f_t(\lambda) =& \arg\min_{f_t \in \mathcal{F}} \frac{1}{L} \sum_{i=1}^{L} \left[\lambda_1 \ell\left(y_i, f_t(x_i)\right) + \sum_{j=1}^{M-1} \lambda_{j+1} \ell\left(y_{ij}^*, f_t(x_i)\right) \right] \\ \text{s.t.} \quad & \sum_i \lambda_i = 1 \end{aligned} \tag{5.3}$$

Compared to other studies on SDA where each example of the source domain was used, by either re-weighting [27, 30] or augmentation [25], 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 [26] 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[29] 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).

5.3.2 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 [63]. **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.

According to the ERM principle[99], 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 mimick the output of the source model on the training data. It is worthy to notice that in this process, the target model only has to mimick the output of the source model (soft label) without considering the hard labels of the examples. In another word, GDSDA provide an effective way to utilize the unlabeled data.

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 [38], we can use labeled data from the target domain to compensate for the domain shift and achieve a better performance on the target task with Eq. (5.3). 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 5.2, when we set $\lambda_2 = 0$, we actually ignore the knowledge from source domain 1. As a result, GDSDA can compensate for the domain shift under the setting of SDA (for more details, please see the experiment section).

5.3.3 Key parameter: the imitation parameter

From the above analysis, we can see that GDSDA can effectively transfer the knowledge from the source to the target domain. 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. Previ-

ous work has addressed the importance of knowledge control in domain adaptation [29, 30]. Without carefully controlling the amount of knowledge transferred from the source domain, the target model may suffer from degraded performance or even negative transfer [72]. 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 [63] or background knowledge [95]. Meanwhile, in real applications, it is common that multiple source domains can be exploited. As suggested by [91], 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.

5.4 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.

5.4.1 Distillation with multiple sources

As suggested in [98], 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 [4, 66, 76, 96] 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:

$$\begin{aligned} \min_{w_n} \quad & \frac{1}{2}\|\mathbf{w}_n\|^2 + C\sum_j e_{jn}^2 \\ s.t. \quad & e_{jn} = \sum_i \lambda_i S_{ijn} - \mathbf{w}_n \mathbf{x}_j \end{aligned} \tag{5.4}$$

We use the KKT theorem [21] and add dual sets of variables to the Lagrangian of the optimization problem:

$$\begin{aligned} \mathcal{L} =& \frac{1}{2}\|\mathbf{w}_n\|^2 + C\sum_j e_{jn}^2 \\ & + \sum_j \eta_{jn}\left(\sum_i \lambda_i S_{ijn} - \mathbf{w}_n \mathbf{x}_j - e_{jn}\right) \end{aligned} \tag{5.5}$$

To find the saddle point,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_n} = 0 \rightarrow \mathbf{w}_n = \sum_j \eta_{jn}\mathbf{x}_j; \quad \frac{\partial \mathcal{L}}{\partial e_{jn}} = 0 \rightarrow \eta_{jn} = 2Ce_{jn} \tag{5.6}$$

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:

$$\sum_j \eta_{jn}\mathbf{x}_j\mathbf{x}_i + \frac{\eta_{in}}{2C} = \sum_i \lambda_i S_{ijn} \tag{5.7}$$

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 [15], the Leave-one-out (LOO) estimation of the example $\mathbf{x}_j$ for the nth binary SVM can be written as:

$$\hat{y}_{jn} = \sum_i \lambda_i \left(S_{ijn} - \frac{\eta'_{ijn}}{\Omega_{jj}^{-1}}\right) \tag{5.8}$$

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.

5.4.2 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:

$$\begin{aligned} \min \quad & L_c(\lambda) = \frac{1}{2}\sum_i^M \|\lambda_i\|^2 + \frac{1}{L}\sum_{j,n} \ell\left(y_{in}, \hat{y}_{jn}(\lambda)\right) \\ s.t. \quad & \sum \lambda_i = 1 \end{aligned} \tag{5.9}$$

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 $f_t = Wx$

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

Find the imitation parameter λ with Algorithm 3

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

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

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

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.

$$\ell\left(y_{in}, \hat{y}_{jn}(\lambda)\right) = y_{in} \log(P_{jn}) \qquad P_{jn} = \frac{e^{\hat{y}_{jn}}}{\sum_h e^{\hat{y}_{jh}}} \tag{5.10}$$

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 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 (5.10) 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:

$$\mu_{ijn} := S_{ijn} - \frac{\eta'_{ijn}}{\Omega^{-1}_{jj}} \tag{5.11}$$

The derivative can be written as:

$$\frac{\partial \ell(\lambda)}{\partial \lambda_i} = \sum_n \mu_{ijn}\left(P_{jn} - y_{jn}\right) \tag{5.12}$$

We describe GDSDA-SVM in Algorithm 2. As the optimization problem (5.9) 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. Due to the space limit, we are not able to provide the proof here.

5.5 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.

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 (5.11)
for $it \in \{1, ..., iter\}$ **do**
  Compute $\hat{y}_{jn}$ and P_{jn} with (5.8) and (5.10)
  $\Delta_\lambda \leftarrow 0$
  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

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 [54] 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.

5.5.1 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[33]) 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 5.4.

From the results of the brutal 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.

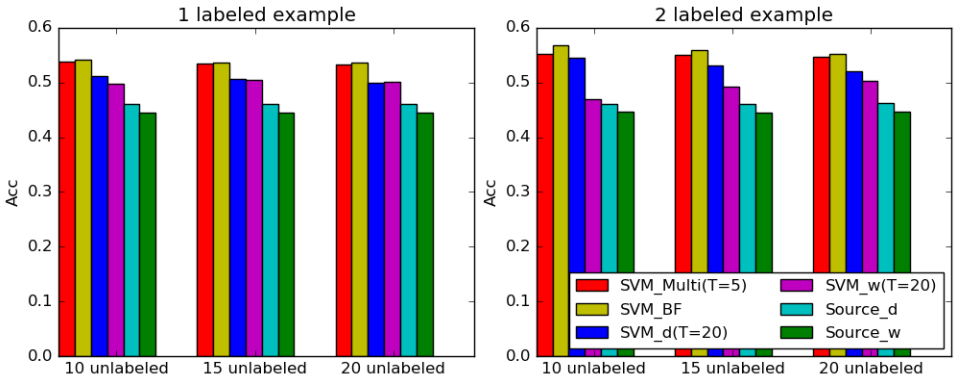


Figure 5.3: D+W→A, Multi-source results comparison.

Moreover, we can see that GDSDA-SVM can achieve competitive results compared to baselines using brutal 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 brutal force search) obtain a good target model with GDSDA-SVM.

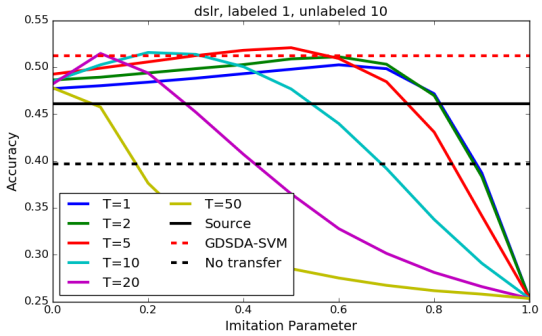
5.5.2 Multi-Source for Office datasets

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 brutal 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 5.3.

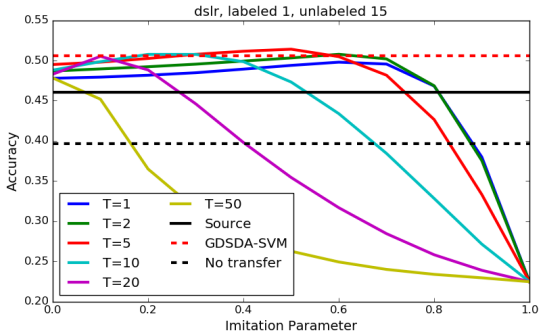
From the results, we can see that, given 2 source models, SVM_Multi can outperform any single source model trained with GDSDA. This indicates GDSDA-SVM can still exploit the knowledge even in the complex multi-source scenario. Even though SVM_Multi performs slightly worse than the best result found by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

5.6 Summary

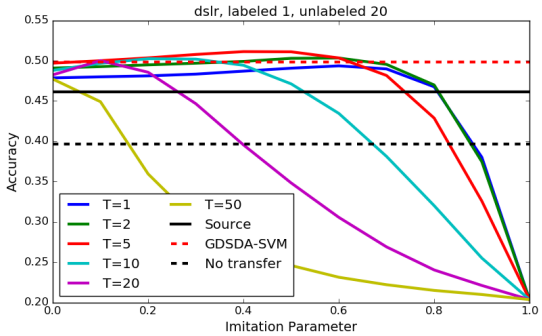
In this paper, we propose a novel framework called *Generalized Distillation Semi-supervised Domain Adaptation* (GDSDA) that can effectively leverage the knowledge from the source domain for SDA problem without accessing to the source data. To make GDSDA more effective in real applications, we proposed 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



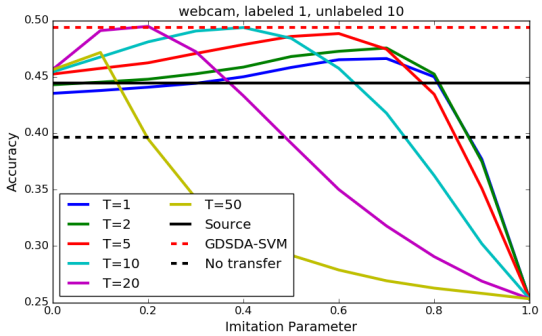
(a) D → A, 10 unlabeled



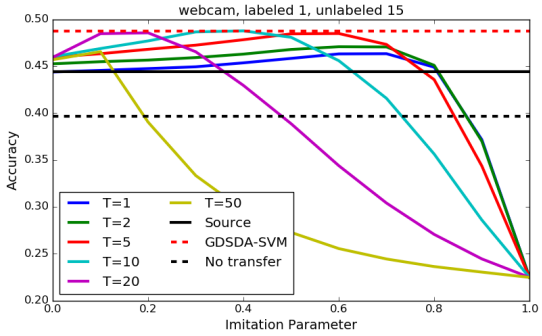
(b) D → A, 15 unlabeled



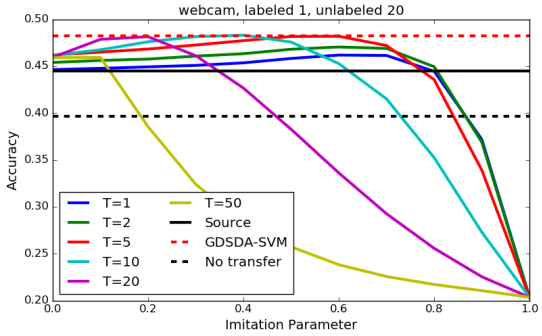
(c) D → A, 20 unlabeled



(d) W → A, 10 unlabeled



(e) W → A, 15 unlabeled



(f) W → A, 20 unlabeled

Figure 5.4: 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 5.2) and the corresponding imitation parameter of the soft label is set to $1 - \lambda_1$.

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] 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.

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

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

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

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

[8] 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.

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

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

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

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

[48] 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.

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1106–1114, 2012.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

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

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

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

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

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

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

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

[76] 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.

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

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

[79] 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.

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

[81] 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.

[82] 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.

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

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

[85] 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.

[86] 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.

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

[88] 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.

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

[90] 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.

[91] 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.

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

[93] 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.

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

[95] 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.

[96] 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.

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

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

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

[100] 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.

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

[102] 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.

[103] 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.

[104] 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.

[105] 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.

[106] 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.

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

[108] 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 Cross Validation Error for LS-SVM

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

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

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:*

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

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:

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

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:

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

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

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

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

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

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:

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

According to block matrix inversion lemma

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

Where $\kappa = \left(S_n - sS^{-1}_{(l-n+1)}s^T\right)$. 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 have $||\Delta_t|| \leq G$ 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 [75], 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 $\left\langle \beta_t - \beta^*, \Delta_y \right\rangle$, it can be written as:

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

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:

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

Curriculum Vitae

Name:	Shuang Ao
Post-Secondary Education and Degrees:	La La School La La Land 1996 - 2000 M.A. University of Western Ontario London, ON 2008 - 2012 Ph.D.
Honours and Awards:	NSERC PGS M 2006-2007
Related Work Experience:	Teaching Assistant The University of Western Ontario 2008 - 2012

Publications:

La La