Knowledge Transfer in Vision Recognition: A Survey

Ying Lu, Di Huang, *Member, IEEE*, Lingkun Luo, Alexandre Saidi, Yunhong Wang, *Senior Member, IEEE*, Liming Chen, *Senior Member, IEEE*

Abstract—In this survey, we propose to discuss knowledge transfer works for vision recognition tasks, and to explore the common rules behind these works. As the primary goal of knowledge transfer methods is to harness learned knowledge for re-use in novel learning tasks, we overview in this survey knowledge transfer methods from the viewpoint of the knowledge being transferred. Specifically, We will firstly discuss the reusable knowledge in a vision recognition task, then we will categorize different kinds of knowledge transfer approaches by where the knowledge comes from and where the knowledge goes to. We aim at finding general rules across different Transfer Learning (TL) settings instead of focusing on their particularities. This viewpoint is in clear contrast to previous surveys on TL. However, any existing method has its own scope of applicability, and we will indicate the applicable scenarios when introducing each method.

Contents

1 Introduction 1

2 Knowledge in Vision Recognition 3

3 Knowledge transfer from source data 5
- 3.1 Knowledge transfer from reweighted source data samples/sets to target classifier parameters . . . 5
- 3.2 Knowledge transfer from source data to target feature extractor . . . 6
  - 3.2.1 Knowledge from labeled source data . . . 6
  - 3.2.2 Knowledge from unlabeled source data . . . 7
- 3.3 Domain Adaptation: knowledge transfer from source data to target feature extractor and classifier parameters . . . 9
  - 3.3.1 Learning a shared feature space with dimensionality reduction methods (Subspace Learning) . . . 10
  - 3.3.2 Learning a shared feature space with metric learning . . . 14
  - 3.3.3 Learning a shared feature space with Deep Neural Networks . . . 14
  - 3.3.4 Subspace alignment . . . 16
  - 3.3.5 Relaxation on shared label space assumption . . . 19

4 Knowledge transfer from pre-learned model 19
- 4.1 From feature extractor parameters (Transferable/shared feature extractor) . 19
  - 4.1.1 Transferable feature learning with shallow Neural Networks 19
  - 4.1.2 Structural learning methods (Shared feature extractor) . . . 19
  - 4.1.3 Transferable feature learning with metric learning . . . 21
  - 4.1.4 Transferable feature learning with Deep Neural Networks . 21
- 4.2 From classifier parameters . . . 22
  - 4.2.1 Knowledge transfer from SVM model parameters . . . 22
  - 4.2.2 Knowledge transfer from generative models . . . 23
- 4.3 Knowledge distillation for DNN (Knowledge from both feature extractor and classifier parameters) . . . 25
- 4.4 From model predictions (Hypothesis transfer learning) . . . 26

References 26

1 Introduction

Vision recognition is a core problem in computer vision. Its goal generally is to determine whether or not the input visual data contains some specific concept, object, or activity and to give corresponding predictive output about the recognized content. For example, in image classification tasks, the goal is to determine which pre-defined image class/concept (*e.g.* 'airport', 'castle', *etc.*) the input image belongs to; in object detection tasks, the goal is to determine whether the input image (or 3D data) contains some specific object (*e.g.* 'bicycle', 'dog', 'mug' *etc.*) and to output the corresponding bounding box (which defines the minimum rectangular/cubic area that contains the object) if an object exists; in semantic segmentation tasks, the goal is to predict semantic label for each pixel or super pixel in an input image. The common schema of this kind of tasks is that they all take some visual data as input and output some predictive semantic label based on the input, therefore the classical way to solve this kind of tasks is supervised learning-based. One first learns a predictive model (*e.g.* a Convolutional

Neural Network) with sufficient training data, and then applies the learned model for prediction on new data whose probabilistic distribution is assumed the same as that of the training data. This learning principle is known as the Empirical Risk Minimization in statistical learning theory [1]. In this way, each task is solved individually by learning a corresponding model from scratch. The disadvantage of this approach is obvious: the relatedness between different tasks is unexplored, thereby making the learning process inefficient. In many real-life applications, some tasks have abundant training data, but most others often have very few training data. When learning each task in an independent manner, it could be hard to solve a task which have limited training data since the available training data may not be enough for learning a reliable model.

It is thus of capital importance to be able to capitalize on previously learned knowledge. Indeed, taking into account the learned knowledge from previous tasks when learning a new task can be beneficial both in gaining extra training information and in saving training time (by avoiding training from scratch). For example, when training an image classification model for some rare categories, one may face the problem of having few training data, in this case, fine-tuning a CNN model pre-learned on some related image data as feature extractor could significantly improve the classification performance on target categories [2]. Knowledge transfer could also be applied for different kind of tasks. For example, since the ground-truth annotations for Object Detection tasks are usually harder to get than those for Image Classification tasks (the former includes not only class labels but also bounding box information), one could therefore borrow knowledge from a learned image classification task for training a new object detection model [3]. This knowledge transfer (also known as 'transfer learning') has been studied in previous works and has been attracting more and more attention from several research communities, *e.g.*, computer vision, machine learning, within the current big data era.

Task 1
(Source)
Knowledge
Task 2
(Target)

Fig. 1. Illustration of Knowledge Transfer: instead of learning a new task independently, knowledge transfer reuse existing knowledge in previous tasks for learning a new task.

Within the research community of knowledge transfer, one usually defines a *target* task, to which the knowledge will be transferred, and one or several *source* tasks, from which the knowledge will be captured or learned. Depending on assumptions on target and source tasks, knowledge transfer setting can be categorized into different scenarios, *e.g.* domain adaptation, self-taught learning, and few-shot learning.

Because of its importance, there exist an increasingly large amount of research work focused on TL and there have already been several surveys discussing state of the art research work on knowledge transfer at different time period as illustrated in Fig.2 and Fig.3. A first comprehensive survey on transfer learning was made in 2010 by Pan and Yang in [4], which discusses transfer learning methods for a broad range of applications, including vision recognition and other types of applications. They have categorized different transfer learning works according to their assumptions (settings) and the nature of content to transfer. Specifically, as shown in the upper diagram of Fig.2, they distinguish three settings of transfer learning: (1) *Inductive* transfer learning, where the target task is different from the source task, some labeled data in the target domain are required to induce an objective predictive model for use in the target domain; (2) *Transductive* transfer learning, where the source and target tasks are the same, but the source and target data distributions are different; and (3) *Unsupervised* transfer learning, where the target task and source task are all unsupervised tasks, *e.g.* clustering, dimensionality reduction, density estimation, *etc.*. Each setting can further depict different transfer learning scenarios with more detailed assumptions. Finally, as shown in the upper part of Figure 3, they come up with a synthesis of four different kinds of transfer learning approaches: (1) *Instance Transfer*, which re-weights some labeled data in the source domain for use in the target domain; (2) *Feature representation transfer*, which tries to find a "good" feature representation that reduces difference between the source and the target domains and the error of classification and regression models; (3) *Parameter transfer*, which discovers shared parameters or priors between the source domain and target domain models; and (4) *Relational knowledge transfer*, which builds mappings of relational knowledge between the source and the target domains.

Another survey was made in 2014 by Shao *et al.* [5] who focus on transfer learning works for vision categorization problems. As shown by the middle diagram of Figure 3, this survey categorizes transfer learning techniques into feature representation level knowledge transfer and classifier level knowledge transfer, respectively.

As [4] and [5] don't cover important development in TL since 2015, a more recent survey was made in 2017 by Zhang *et al.* [6] who overview transfer learning techniques for cross dataset recognition problems. Like in [4], they also distinguish different works according to the settings. Specifically, they show what kinds of methods can be used when the available source and target data are presented in different forms. Compared to [4], they give a more detailed categorization of different cross-dataset settings, as shown in the lower diagram of Fig.2. They also summarize different kinds of criteria which could be used in solving knowledge transfer problems, as shown in the lower diagram of Fig.3. Like in [4], the transfer learning methods discussed in [6] also covers a broad range of applications. Vision recognition is only one of them.

As shown in the beginning of this section 1, the common way to solve vision recognition tasks follows the principle of Empirical Risk Minimization. Due to the specialty of visual data, this common solution could be further defined as a two-step framework: the feature extraction step and the prediction step. Following this two-step framework, [5] simply categorizes transfer learning works for vision categorization problems into two categories (as shown in figure 3). Although this categorization shows its correspondence to the common way of solving visual problems, it does not fully reveal the characteristics of knowledge transfer within this scope. Furthermore, [5] only covers research works before 2014. While with the rapid growth of vision recognition techniques, a lot of new works, especially those based on deep neural networks, are published since 2015. These recent works are not discussed in [5]. On the contrary, [6] includes more recent transfer learning works, while the authors do not focus on vision recognition.

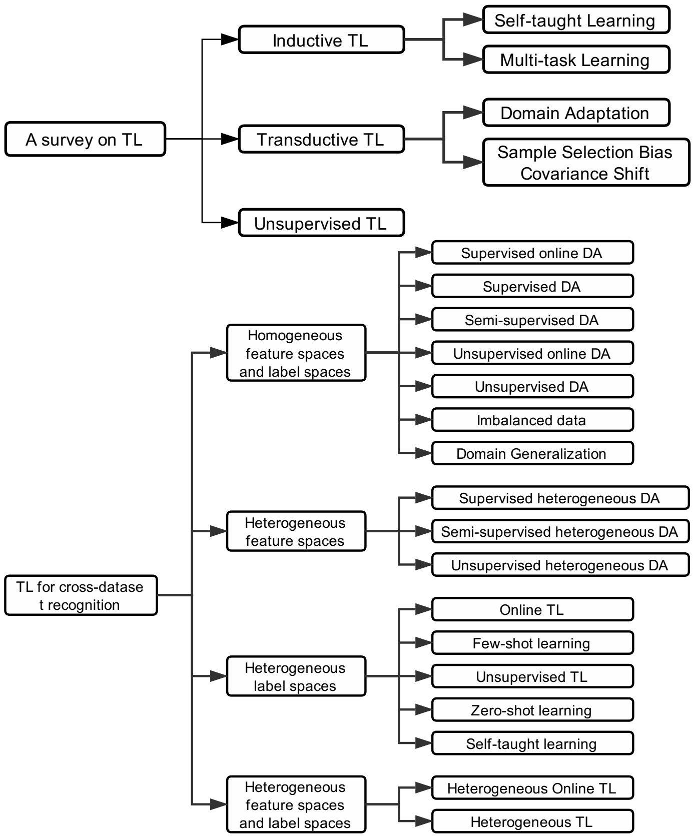

Fig. 2. Existing Categorizations of Transfer Learning Settings: The top part is the categorization of TL settings from [4] and the bottom part is the categorization of TL settings from [6] (TL stands for Transfer Learning and DA stands for Domain Adaptation)

In this survey, we propose to discuss knowledge transfer works for vision recognition tasks, and to explore the common rules behind these works. As the primary goal of TL methods is to harness learned knowledge for re-use in novel learning tasks, we overview in this survey knowledge transfer methods from the viewpoint of the knowledge being transferred. Specifically, We will firstly discuss the reusable knowledge in a vision recognition task, then we will categorize different kinds of knowledge transfer approaches by where the knowledge comes from and where the knowledge goes to. We aim at finding general rules across different TL settings instead of focusing on their particularities. This viewpoint is in clear contrast to previous surveys on TL, *i.e.*, [4] [5][6]. However, any existing method has its own scope of applicability, and we will indicate the applicable scenarios when introducing each method. In Table 1 we have listed some common knowledge transfer settings (scenarios) discussed in this paper.

TABLE 1
Some common Transfer Learning settings concerned in this paper

Name	Same label space?	Inputs: Source Target	Objective
Inductive TL	Different	labeled labeled	single
Domain Adaptation	Same	labeled unlabeled	single
Self-taught Learning	Different	unlabeled labeled	single
Multi-task Learning	Different	labeled labeled	multiple
Zero shot Learning	Different	labeled no	single
One shot Learning	Different	labeled labeled	single

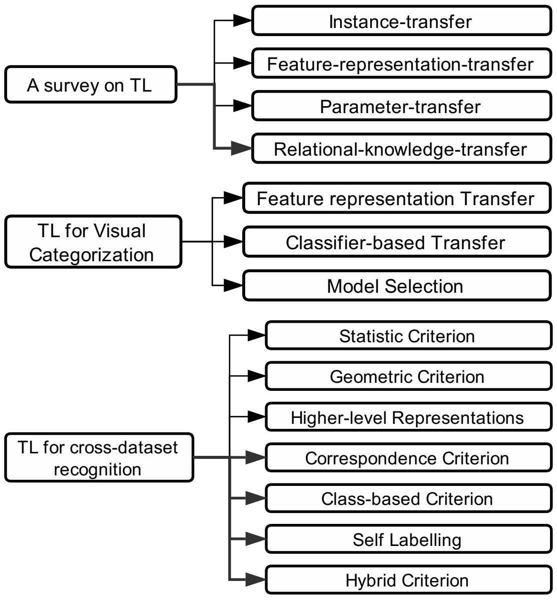
A survey on TL
Instance-transfer
Feature-representation-transfer
Parameter-transfer
Relational-knowledge-transfer
TL for Visual Categorization
Feature representation Transfer
Classifier-based Transfer
Model Selection
TL for cross-dataset recognition
Statistic Criterion
Geometric Criterion
Higher-level Representations
Correspondence Criterion
Class-based Criterion
Self Labelling
Hybrid Criterion

Fig. 3. Existing Categorization of Transfer Learning approaches/techniques: The top part is categorization of TL approaches from [4], the middle part is categorization of TL approaches from [5], and the bottom part is categorization of TL techniques from [6]

2 Knowledge in Vision Recognition

In this section we firstly introduce the notations we adopt in this paper to describe a vision recognition task and its solution. Then we discuss different types of knowledge that can be reused (transferred) from a previous vision recognition task to a new one.

The following notations are adopted in the subsequent: calligraphic letters in upper cases, *e.g.*, $\mathcal{X}$, denote sets or data spaces; bold letters in upper cases, *e.g.*, $\mathbf{M}$, denote matrices; bold letters in lower cases, *e.g.*, $\mathbf{x}$, denote column vectors.

Let's firstly define a vision recognition task $\mathcal{T}$ by two data spaces $\mathcal{X}, \mathcal{Y}$ and the corresponding data distributions $P(X), P(Y)$, where $\mathcal{X}$ is the input data space, $P(X)$ is the marginal probability distribution of the input data X, $\mathcal{Y}$ is

TABLE 2
Characteristics which define a vision recognition task and its solution

In/Out Data	$\mathbf{X}^{tr}$, $\mathbf{Y}^{tr}$, $\mathbf{X}^{te}$, $\mathbf{Y}^{te}$
In/Out data spaces	$\mathcal{X}, \mathcal{Y}$
In/Out data distributions	$P(X)$, $P(Y)$
Learned model	$f = f_K \circ \ldots \circ f_2 \circ f_1$
Feature data	$\mathbf{X}^{set,f_1}, \mathbf{X}^{set,f_2}, \ldots, \mathbf{X}^{set,f_{K-1}}$ $(set = \{tr, te\})$
Feature data spaces	$\mathcal{X}^{f_1}, \ldots, \mathcal{X}^{f_{K-1}}$
Feature data distributions	$P(X^{f_1}), P(X^{f_2}), \ldots, P(X^{f_{K-1}})$

the desired output label space, and $P(Y)$ is the probability distribution of the output data Y. Here $P(Y)$ can also be noted as $P(Y|X)$ and therefore be interpreted as the conditional distribution of Y knowing X. The goal of the vision recognition task is to find an optimal mapping $f(\cdot)$ which projects the input data X from the data space $\mathcal{X}$ to the label space $\mathcal{Y}$ so that the mapped labels Y best correspond to the ground-truth labels Y^{gt}. This mapping $f(\cdot)$ could be further decomposed into a series of projections $f = f_K \circ \ldots \circ f_2 \circ f_1$, where each f_k maps data from the space $\mathcal{X}^{f_{k-1}}$ to the space $\mathcal{X}^{f_k}$. Specifically, $\mathcal{X}^{f_0} = \mathcal{X}$ is the input data space, $\mathcal{X}^{f_K} = \mathcal{Y}$ is the output data space, and $\{\mathcal{X}^{f_1}, \ldots, \mathcal{X}^{f_{K-1}}\}$ are intermediate feature spaces. In this way, a solution to task $\mathcal{T}$ could be defined as $\mathcal{S} = \{f_1, f_2, \ldots, f_K\}$, and these projections define new feature spaces $\mathcal{F} = \{\mathcal{X}^{f_1}, \ldots, \mathcal{X}^{f_{K-1}}\}$ and the output data space given the input data space.

For example, if we use a K-layer Neural Network model to solve the task $\mathcal{T}$, the solution could be denoted as $\mathcal{S}_{KlayersNN} = \{f_1, f_2, \ldots, f_K\}$, and the corresponding feature spaces are $\mathcal{F}_{KlayersNN} = \{\mathcal{X}^{f_1}, \ldots, \mathcal{X}^{f_{K-1}}\}$, where $\mathcal{X}^{f_k}$ $(k = \{1, 2, \ldots, K-1\})$ is the feature space which contains the output of the k-th layer of the Neural Network, and $\mathcal{X}^{f_K} = \mathcal{Y}$ contains the output of the last layer, which is the desired output label space. If we use a traditional way to solve $\mathcal{T}$, for example, a SIFT feature extraction step with an SVM classifier, the solution could then be denoted as $\mathcal{S}_{SIFT-SVM} = \{f_{SIFT}, f_{SVM}\}$ along with $\mathcal{F}_{SIFT-SVM} = \{\mathcal{X}^{f_{SIFT}}\}$, where $\mathcal{X}^{f_{SIFT}}$ is the feature space defined by the output of the SIFT feature extraction.

In reality, the probability distributions $P(X)$ and $P(Y)$ for $\mathcal{T}$ are usually not given in an analytical form. Normally they could be estimated through the given training data set $\{\mathbf{X}^{tr}, \mathbf{Y}^{tr}\}$. The test data $\{\mathbf{X}^{te}, \mathbf{Y}^{te}\}$ are supposed to follow the same distribution as the training data do, therefore allowing the model learned on training data to be applicable on the test data. In the training phase, a model $f(\cdot)$ is learned with the training set $\{\mathbf{X}^{tr}, \mathbf{Y}^{tr}\}$, and then in the testing phase, predictions could be made by applying the learned model $f(\cdot)$ on the test data $\mathbf{X}^{te}$, and the performance of the model could be evaluated by comparing the predictions with the groundtruth labels $\mathbf{Y}^{te}$. Following the decomposition of f described above, we could find out that there exist a series of feature data $\{\mathbf{X}^{set,f_1}, \mathbf{X}^{set,f_2}, \ldots, \mathbf{X}^{set,f_{K-1}}\}$ $(set = \{tr, te\})$, each belongs to their corresponding feature space, *i.e.* $\mathbf{X}^{tr,f_k}$ and $\mathbf{X}^{te,f_k}$ belong to $\mathcal{X}^{f_k}$. And in each feature space the data should follow a specific probability distribution, we denote these data distributions as $\{P(X^{f_1}), P(X^{f_2}), \ldots, P(X^{f_{K-1}})\}$.

Table 2 lists the main characteristics introduced above that describe a vision recognition task and its solution. At this level, We can observe that there exist two types of *knowledge* for a given vision recognition task: the first one is directly represented by raw data which includes the input and output data, the corresponding data spaces they belong to, and their data distributions; the other is learned knowledge which includes the learned model, feature data generated by this model, feature data spaces and feature data distributions. Both these two kinds of knowledge have the possibility to be reused (transferred) in a new vision recognition task. The knowledge directly represented by raw data is more flexible to be reused since they could be adapted to the target data for learning a new model; In contrast, the learned knowledge is more restricted to the source data. Nevertheless, when the source task is well chosen (*i.e.*, well related to target task), the reuse of learned knowledge could be both efficient and effective.

For example, it has been shown that deep convolutional neural networks (CNN) have the ability to produce transferable features [2]. Therefore, one can adopt a pre-learned CNN model, fix the feature extraction layers' parameters and only retrain the classification layer's parameters on new data, and the resulting new model is expected to have discriminative performance on the new data. In this way, we are actually reusing the *knowledge* from the parameters of the learned feature projections (*i.e.* $\{f_1, f_2, \ldots, f_{K-1}\}$) of a given pre-learned vision recognition task.

In some cases, the target training data is far from enough for learning a reliable classification model, some instance based transfer learning approaches then choose to select source samples to enrich the target training data directly. For example, Dai *et al.* in [7] make use of AdaBoost to choose from the source labeled samples the ones which are close to the target probability distribution to help the learning of the target classification model. In this way, the *knowledge* transferred from the source task are source raw data (*i.e.* $\mathbf{X}$ and $\mathbf{Y}$), and they are reused for learning a prediction model for the target task.

Another example is unsupervised Domain Adaptation (DA). In unsupervised DA, the target training data is unlabeled, therefore it is impossible to use the target training data solely for learning a reliable classification model. We thus need to seek for help in labeled source data, which is assumed to share the same label space with the target domain and to own similar but different marginal distribution *w.r.t* the target domain. In this case, one can attempt to align the source and target data distributions (either by projecting the two distribution into a shared feature space in which the two feature distributions are as close to each other as possible; or by projecting one distribution to fit the other one). Then as the two distributions are well aligned in a new feature space, a classification model learned on the source distributions is then considered as applicable for target data. In this way, the *knowledge* reused from source is actually the same as in the previous example, *i.e.*, source labeled data is reused and adapted to target data for learning a new classification model for target task.

In the following sections, we overview different knowledge transfer techniques or approaches for vision recognition tasks. They are categorized according to the origin of *knowledge* being transferred, *i.e. raw data* or *pre-learned model*), as well as the destination of the transferred knowledge, *e.g.* feature extractor parameters or predictor parameters, *etc.*). Table 3 synthesizes the different knowledge transfer techniques that are presented in the subsequent.

TABLE 3
Categorization of Knowledge Transfer Approaches by where the knowledge come from and where the knowledge go to

From \ **To**	Feature Extractor parameters	Predictor parameters	Both
Raw data	section 3.2	section 3.1	section 3.3
Learned Feature extractor	section 4.1	-	-
Learned Predictor	-	section 4.2	-
Learned model	-	-	section 4.3

3 Knowledge transfer from source data

As we have discussed in section 2, *raw data* is low level information compared to *pre-learned model*. Therefore there exist more possibilities to reuse *raw data* in target task. The main advantage of using *raw data* instead of *pre-learned model* is that *raw data* could be more easily adapted to target task. There are various ways to adapt source data to target data. For example, when the source and target data distributions are not very far from each other, a straight forward way is to re-weight (or select) source samples (or sets) so that the resulting data set could fit the target data distribution. An alternative way is to learn a shared feature extractor which projects both source and target data into a common feature space, in which the source and target feature distributions are well aligned to each other. When the source and target data distributions are not very close to each other, one could learn a projection which projects source data to the target data space (or the inverse) so that the resulting two data distributions would be close to each other. Once the two data/feature distributions are well aligned, the target task can then benefit from this shared data/feature space in different ways. For example, the target task could benefit from the discriminability of a learned shared feature space, or it could benefit from the source conditional distribution for learning a classifier in the shared data space if the target samples are not enough to support a reliable classification border.

Therefore we further categorize knowledge transfer approaches by two different kinds of knowledge destinations, *i.e.* the *feature extractor parameters* and the *predictor parameters* for the target task. These two kinds of knowledge destinations are not mutually exclusive, some works may focus on transferring source data knowledge to one particular kind of target model parameters, while some works reuse source data for learning all kinds of target model parameters.

In the following we introduce three groups of previous works that transfer source data knowledge for learning target model parameters. Detailed knowledge transfer category for each section can be found in table 3.

3.1 Knowledge transfer from reweighted source data samples/sets to target classifier parameters

A natural way to adapt source data to target is by selecting most related data samples directly from source training data set, or selecting most related source sets when multiple source sets are presented. The selection is usually done by giving weights on source samples/sets. In this subsection we show a group of methods which transfers knowledge from the reweighted source data samples/sets to the learning of target classifier parameters.

In early days before deep convolutional neural networks take over traditional feature extraction skills, research works on knowledge transfer mainly focus on borrowing knowledge from source to build the target predictor (*i.e.* knowledge transfer from source data to target predictor parameters). For feature extraction they use traditional methods (such as SIFT, HOG, *etc.*) on source and target training samples. And they use the extracted features of both source and target training samples as input data to learn a predictive model for target task. In the training process, source samples/sets that are more related to target samples will be given more important weights to enhance the knowledge transferred from them. Different works may use different ways to describe the relatedness between source and target samples, and use different strategies to select related source samples/sets.

In [7] the authors make use of AdaBoost for transfer learning by choosing from the source labeled samples the useful ones for building a classifier for target data. Assume having a few target training samples and a large amount of source training samples, their aim is to select the source samples that follow the same probability distribution as the target samples. To achieve this goal, they build a Transfer AdaBoost framework, namely TrAdaBoost, for learning on target and source training samples at the same time. In each iteration, AdaBoost works normally on target samples, *i.e.* it increases the weights on misclassified target samples; on the other hand, for source training samples, the misclassified ones are considered as the outliers to the target distribution, therefore the weights on misclassified source samples are decreased. In this way, after several iterations, the source samples that fit the target distribution better will have larger weights, while the source samples that do not fit the target distribution will have lower weights. The instances with large weights will intent to help the learning algorithm to train better classifiers.

Since this TrAdaBoost only borrows knowledge from one source task, in [8] the authors extend this method to MultiSource-TrAdaBoost which borrows knowledge from multiple source tasks. Assume having several different source training sample sets, each with abundant labeled samples, and one target training sample set with few labeled samples. In each iteration of AdaBoost, one weak learner is build on each source training set, and the one with the best performance on target set, *i.e.* the one appears to be the most closely related to the target, is chosen as the weak learner for current iteration. In this way, the authors claim that the MultiSource-TrAdaBoost can better avoid negative transfer effect caused by the imposition to transfer knowledge from a single source, which is potentially loosely related to the target.

Beware that, both TrAdaBoost and MultiSource-TrAdaBoost work for binary classification only, *i.e.* the source and target label spaces are the same, which could be defined as $\{+1, -1\}$ where $+1$ indicates positive sample and -1 indicates negative sample. Therefore these two works could make use of selected source samples simply as a part of target training data set for learning classification model. This strategy works when source set and target set are positively correlated, *i.e.* there exist source positive samples which are related to target positive samples and source negative samples which are related to target negative samples. Otherwise, when the two data distributions are not correlated, making use of source samples may harm the performance of the learned model for target task.

An alternative approach would solve this more complicated situation. In [9] the authors propose to use label propagation for knowledge transfer, *i.e.* they propagates labels from samples of selected source sets to each target sample. The resulting method is named Cross-Category Transfer Learning (CCTL). The coefficient for label propagation from a source sample to a target sample is defined by a transfer function, which combines both sample relatedness and domain relatedness between source and target. And the source set selection is also achieved by AdaBoost. Specifically, they define a real-valued transfer function $T_S(\mathbf{x}, \mathbf{x}_{l,i}) = \phi_S(\mathbf{x}, \mathbf{x}_{l,i}) k(\mathbf{x}, \mathbf{x}_{l,i})$ (where $\mathbf{x}_{l,i} \in \mathcal{D}_{S,l}$, $\mathbf{x} \in \mathcal{D}_T$) to connect the l-th source set $\mathcal{D}_{S,l}$ and the target training set $\mathcal{D}_T$. In which, the $\phi_S(\mathbf{x}, \mathbf{x}_{l,i}) = \mathbf{x}^\top S \mathbf{x}_{l,i}$ measures the correlation between two different categories, and the kernel function $k(\mathbf{x}, \mathbf{x}_{l,i})$ measures the sample similarity. A cross-category classifier is learned to propagate the labels from the instances in l-th source set $\mathcal{D}_{S,l}$ to the target training set to form a discriminant function $h_l(\mathbf{x})$ as follows:

$$h_l(\mathbf{x}) = \frac{1}{|\mathcal{D}_{S,l}|} \sum_{\mathbf{x}_{l,i} \in \mathcal{D}_{S,l}} y_{l,i} T_S(\mathbf{x}, \mathbf{x}_{l,i}) \tag{1}$$

where $|\mathcal{D}_{S,l}|$ is the cardinality of $\mathcal{D}_{S,l}$, and $y_{l,i}$ is the ground-truth label for $\mathbf{x}_{l,i}$. The parameter matrix S for $h_l(\mathbf{x})$ is learned by minimizing the following objective function:

$$S^* = \arg\min_S \Omega_l(S) \tag{2}$$

where

$$\Omega_l(S) = \sum_{\mathbf{x}_i \in \mathcal{D}_T} \mathbf{w}_i (1 - y_i h_l(\mathbf{x}_i))_+ + \frac{\lambda}{2} \|S\|_F^2 \tag{3}$$

where $(\cdot)_+ = max(0, \cdot)$, $\|S\|_F$ is the Frobenius norm of the matrix S, λ is the balancing parameter, y_i is the ground-truth label for $\mathbf{x}_i$ and $\mathbf{w}_i$ is the sample weight for $\mathbf{x}_i$.

Finally, they define a common AdaBoost process, in each iteration they learn a cross category classifier from each source domain to the target domain, and a same one from target domain to itself, they then pick from these classifiers the one with the minimum training error as the weak classifier for current iteration. the final output is a combination of the weak classifiers learned in all iterations.

Since this CCTL takes into account both category correlations and sample correlations, it shows a better performance than the previously introduced TrAdaBoost and MultiSource-TrAdaBoost. However, CCTL also only works for binary classification problems. Furthermore, when having L different source domains, in each iteration of CCTL one should solve $L + 1$ optimization problems. This makes this method not very efficient, especially when having a lot of source domains.

Although these methods make use of traditional feature extraction, we could easily replace their feature extractors with a state-of-the-art CNN model, which is pre-learned on some related large scale database, to benefit from the better performance of deep features.

In [10] the authors propose a new method, namely Discriminative Transfer Learning (DTL), to reuse source selected data set for learning target classifier. They also show that by combining deep features and knowledge transfer in target classifier one can achieve better results than using traditional features. Unlike previous methods, the authors propose to build sparse reconstruction based discriminative classifiers for target task with selected source sample sets. They use positively correlated source sets as positive dictionaries and negatively correlated source sets as negative dictionaries, the difference between reconstruction errors of target samples on positive dictionary and those on negative dictionary is served as the discriminator. The source data sets are selected through two parallel AdaBoost processes. Therefore the resulting classification model is a combination of multiple selected dictionary pairs. Since this method makes use of both positively correlated source sets and negatively correlated source sets, it shows a better performance than the previously introduced CCTL, and it is also much more efficient than CCTL both on training time and on prediction time.

As can be seen, the methods introduced in this section are all based on boosting framework and all work for binary classification. It is possible to extend these kind of methods to multi-class classification by one-vs-one, one-vs-all or EOOC (Error Correcting Output codes) based approaches, although this extension may increase the time for training and prediction. In table 4 we give a comparison of the detailed setting of these methods along with the original AdaBoost.

3.2 Knowledge transfer from source data to target feature extractor

In previous subsection we have introduced some transfer learning methods which transfer knowledge from source data to target discriminator parameters. In this section, we show another group of methods, which also transfer knowledge from source data, while they mainly focus on using these knowledge to learn a feature extractor adapted for the target task.

3.2.1 Knowledge from labeled source data

As shown in section 3.1, a straight forward way to adapt source data to target is by re-weighting source data sam-

TABLE 4
Comparison of boosting based knowledge transfer methods

Boosting based methods	In each Boosting iteration:	
	Update sample weights ($\uparrow$: augment weight; $\downarrow$: decrease weight)	Choose weak learner
AdaBoost	Wrongly classified samples $\uparrow$ Correctly classified samples $\downarrow$	Learned with weighted samples
TrAdaBoost	Wrongly classified target samples $\uparrow$ Correctly classified target samples $\downarrow$ Wrongly classified source samples $\downarrow$ Correctly classified source samples $\uparrow$	Learned with weighted target and source samples
MultiSourceTrAdaBoost	Wrongly classified target samples $\uparrow$ Correctly classified target samples $\downarrow$ Wrongly classified source samples $\downarrow$ Correctly classified source samples $\uparrow$	The one with best performance on target from candidates learned with multiple sources
CCTL	Wrongly classified samples $\uparrow$ Correctly classified samples $\downarrow$	The one with best performance on target from candidate cross-category classifiers
DTL	Wrongly classified samples $\uparrow$ Correctly classified samples $\downarrow$	Multiple pairs of source sets which show best performance on target

ples/sets. For learning feature extractor parameters adapted for target, we could also use this kind of strategy.

For example, in [11] the authors propose a method which selects related source samples and then learn a deep convolutional neural network for feature extraction through joint fine-tuning with both selected source samples and target training samples. The proposed *Selective Joint Fine-Tuning* is done by two steps:

In step 1, they select nearest neighbors of each target domain training sample in the source domain via a low-level feature space. This is done by applying a filter bank to all images in both source domain and target domain. Histograms of filter bank responses are used as image descriptors during search. Two filter banks are used for experiments, one is the Gabor filter bank, the other consists of kernels in the convolutional layers of AlexNet pre-trained on ImageNet. An adaptive number of source domain images is associated with each target domain image (Hard training samples in the target domain might be associated with a larger number of source domain images).

In step 2, they jointly optimize the source and target cost functions in their own label space on a shared DNN (a 152-layer residual network with identity mappings is used in this work) initialized with weights pre-trained on ImageNet or Places.

In a more recent work [12] the authors address a similar problem: how to select an optimal Subset of Classes (SOC) from the source data, subject to a budget constraint, for training a feature extractor which generates good features for the target task. To achieve this goal, they use a sub-modular set function to model the accuracy achievable on a new task when the features have been learned on a given subset of classes of the source dataset. An optimal subset is identified as the set that maximizes this sub-modular function. The maximization can be accomplished using a greedy algorithm that comes with guarantees on the optimality of the solution.

3.2.2 Knowledge from unlabeled source data

The methods shown in previous section 3.2.1 all make use of labeled source data for training feature extractor. In reality, it is usually more expensive to get labeled data than unlabeled data, therefore transferring knowledge from unlabeled data would be a good choice when it is not easy to get source labels. In this subsection, we show some works that make use of unlabeled source data for learning a feature extractor purposed for target task.

A first group of methods is the so called *self-taught learning* methods. These methods aim to learn a re-constructive dictionary from unlabeled source data, this learned dictionary could then be used for feature extraction by sparse coding for target task.

[13] is the first work that proposed the *self-taught learning* problem. They proposed an approach to self-taught learning that uses sparse coding to construct higher-level features using the unlabeled data. These features form a succinct input representation and can improve classification performance for the target task.

In self-taught learning, one is given a labeled training set of m samples $\{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \ldots, (x_l^{(m)}, y^{(m)})\}$ drawn i.i.d. from some distribution $\mathcal{D}$. Here, each $x_l^{(i)} \in \mathbb{R}^n$ is an input feature vector (the "l" subscript indicates that it is a labeled example), and $y^{(i)} \in \{1, \ldots, C\}$ is the corresponding class label. In addition, a set of k unlabeled examples $x_u^{(1)}, x_u^{(2)}, \ldots, x_u^{(k)} \in \mathbb{R}^n$ is also given. The unlabeled data are not assumed to be drawn from the same distribution as the labeled data, nor that it can be associated with the same class labels as the labeled data. However, the labeled and unlabeled data are assumed not to be completely irrelevant to each other if unlabeled data is to help the target task.

To learn the higher-level representations, the authors proposed a modified version of the sparse coding algorithm [14]. Specifically, given the unlabeled data $\{x_u^{(1)}, \dots, x_u^{(k)}\}$ with each $x_u^{(i)} \in \mathbb{R}^n$, they propose the following optimization problem:

$$\min_{b,a} \sum_i \|x_u^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 + \beta \|a^{(i)}\|_1 \tag{4}$$

$$\text{s.t.} \quad \|b_j\|_2 \leq \mathbf{1}, \forall j \in 1, \dots, s$$

The optimization variables in this problem are the *basis vectors* $b = \{b_1, b_2, \dots, b_s\}$ with each $b_j \in \mathbb{R}^n$, and the *activations* $a = \{a^{(1)}, \dots, a^{(k)}\}$ with each $a^{(i)} \in \mathbb{R}^s$, $a_j^{(i)}$ is the activation of basis b_j for input $x_u^{(i)}$. The number of bases s can be much larger than the input dimension n. This optimization objective balances two terms: (1) The first quadratic term encourages each input $x_u^{(i)}$ to be reconstructed well as a weighted linear combination of the bases b_j (with corresponding weights given by the activations $a_j^{(i)}$); (2) The second term encourages the activations to have low L_1 norm, *i.e.*, it encourages the activations a to be *sparse* – in other words, for most of its elements to be zero. The problem (4) is convex over each subset of variables a and b (though not jointly convex); in particular, the optimization over activations a is an l_1-regularized least square problem, and the optimization over basis vectors b is an $l2$-constrained least square problem. These two convex sub-problems can be solved efficiently, and the objective in (4) can be iteratively optimized over a and b alternatively while holding the other set of variables fixed.

It is often easy to obtain large amounts of unlabeled data that shares several salient features with the labeled data from the target classification task. Building on this observation, the authors propose the following approach to *self-taught learning*: they first apply sparse coding to the unlabeled data $x_u^{(i)} \in \mathbb{R}^n$ to learn a set of bases b. Then for each training input $x_l^{(i)} \in \mathbb{R}^n$ from the target task, they compute features $\hat{a}(x_l^{(i)}) \in \mathbb{R}^s$ by solving the following optimization problem:

$$\hat{a}(x_l^{(i)}) = \arg\min_{a^{(i)}} \|x_l^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 + \beta \|a^{(i)}\|_1 \tag{5}$$

This is a convex $l1$-regularized least square problem and can be solved efficiently. It approximately expresses the input $x_l^{(i)}$ as a sparse linear combination of the bases b_j. The sparse vector $\hat{a}(x_l^{(i)})$ is the new representation for $x_l^{(i)}$. These new features are taken as input to standard supervised classification algorithms (such as SVMs) for target task.

The authors argue that, compared to PCA, this proposed sparse coding process is a better way for unsupervised feature learning in the self-taught learning scenario for two reasons: first, PCA results in *linear* feature extraction while the sparse coding method learns a nonlinear representation $\hat{a}(x)$ due to the presence of the $l1$ term in equation (4); second, since PCA assumes the bases to be orthogonal, the number of PCA features cannot be greater than the dimension n of the input, while sparse coding can use more basis vectors than the input dimension. By learning a large number of basis vectors but using only a small number of them for any particular input, sparse coding gives a higher-level representation in terms of the many possible "basic patterns".

The authors perform experiments of the proposed sparse coding approach with two standard classifiers: a support vector machine(SVM) and a Gaussian discriminant analysis (GDA). In addition, they also proposed a Fisher kernel based classifier specifically designed for sparse coding features. They show results on several different applications including image classification, the results confirmed the effectiveness of the proposed approach.

This sparse coding based approach is widely adopted for self-taught learning scenarios, and is also improved from different aspects by different researchers. For example, in [15] the authors propose to learn the sparse coding basis (*i.e.*, the redundant dictionary) using not only unlabeled samples, but also labeled samples. They also proposed a principled method to seek the optimal dictionary basis vectors for a smaller dictionary which demands less computational cost.

In a recent work [16], the authors propose a new sparse coding based self-taught learning framework for visual learning, which can utilize the rich low-level pattern information abstracted from the auxiliary domain, in order to characterize the high-level structural information in the target domain. Since many types of visual data have been proven to contain subspace structures, a low-rank constraint is introduced into the coding objective to better characterize the structure of the given target set. This proposed representation learning framework is called self-taught low-rank (S-Low) coding, which can be formulated as a non-convex rank-minimization and dictionary learning problem.

Consider having a set of abundant unlabeled samples $\mathbf{X}_S = \{\mathbf{x}_1^{(S)}, \dots, x_m^{(S)}\} \in \mathbb{R}^{d \times m}$ in the source domain, and a limited number of samples $\mathbf{X}_T = \{x_1^{(T)}, \dots, x_n^{(T)}\} \in \mathbb{R}^{d \times n}$ in the target domain. Here d is the feature dimension, m is the number of samples in source training set, and n is the number of samples in target training set. Unlike the previous approach [13], in this work the authors do not require that the samples in the target domain are labeled. Therefore their framework can be adapted to either unsupervised or supervised situations according to the availability of labels on target training samples.

Conventionally, the sparse coding, dictionary learning or low-rank learning methods approximately represent the samples in a single domain (here we take the target domain as an example) as:

$$\mathbf{X}_T \approx \mathbf{D}_T \mathbf{Z}_T \tag{6}$$

where $\mathbf{Z}_T \in \mathbb{R}^{r \times n}$ is the representation coefficient matrix and $\mathbf{D}_T \in \mathbb{R}^{d \times r}$ is a dictionary with r the size of this dictionary. $\mathbf{Z}_T$ is usually expected to be sparse or low-rank, according to the application scenario. To make use of samples in both domains, the authors propose to learn the dictionary from all available samples in two domains (source and target). The whole sample set is denoted as $\mathbf{X} = [\mathbf{X}_S \ \mathbf{X}_T]$. Therefore they propose the following constraint:

$$[\mathbf{X}_S \ \mathbf{X}_T] = \mathbf{D}[\mathbf{Z}_S \ \mathbf{Z}_T] + [\mathbf{E}_S \ \mathbf{E}_T] \tag{7}$$

where $\mathbf{Z}_S \in \mathbb{R}^{r\times m}$ and $\mathbf{Z}_T \in \mathbb{R}^{r\times n}$ are the coefficient matrices corresponding to source domain and target domain, respectively. $\mathbf{E}_S \in \mathbb{R}^{d\times m}$ and $\mathbf{E}_T \in \mathbb{R}^{d\times n}$ are the sparse noise matrices that model the reconstruction errors in auxiliary and target domains. The noise matrices $\mathbf{E}_S$ and $\mathbf{E}_T$ are often constrained using the surrogate of $l0$ norm which enables the model to learn a robust dictionary.

To discover the underlying subspace structure in training data, the authors further propose to impose a low-rank constraint on the coefficient matrix $\mathbf{Z}_T$ in the target domain where the learning tasks are performed. The objective function can therefore be formulated as:

$$\begin{aligned} \min_{\mathbf{D},\mathbf{Z}_S,\mathbf{Z}_T,\mathbf{E}_S,\mathbf{E}_T} \quad & \text{rank}(\mathbf{Z}_T) + \lambda_1\|\mathbf{E}_S\|_0 + \lambda_2\|\mathbf{E}_T\|_0 \\ \text{s.t.} \quad & \mathbf{X}_S = \mathbf{D}\mathbf{Z}_S + \mathbf{E}_S \\ & \mathbf{X}_T = \mathbf{D}\mathbf{Z}_T + \mathbf{E}_T \end{aligned} \tag{8}$$

where $\text{rank}(\cdot)$ denotes the rank function, $\|\cdot\|_0$ is the l_0 norm, and λ_1 and λ_2 are two trade-off parameters. In Eq.(8) the first term characterizes the low rankness of $\mathbf{Z}_T$ in the target domain, and the last two terms model the reconstruction errors. This is a variant of rank minimization problem that is NP-hard in general. Therefore, it cannot be solved directly, normal solution would be to relax the l_0 norm and the rank function with l_1 norm and nuclear norm respectively. However, the authors argue that the l_1 norm and the nuclear norm are biased estimators, as they over penalize large entries and large singular values. Therefore, the authors propose to employ the non-convex surrogates of l_0 norm and rank function, which are MCP norm and matrix γ-norm, respectively.

The definition of matrix MCP norm for a matrix $\mathbf{B} \in \mathbb{R}^{p\times q}$ is:

$$M_{\lambda,\gamma}(\mathbf{B}) = \sum_{i,j} \phi_{\lambda,\gamma}(B_{i,j}) \tag{9}$$

$$\phi_{\lambda,\gamma}(t) = \lambda\int_0^t \left[1 - \frac{x}{\gamma\lambda}\right] dx = \begin{cases} \gamma\frac{\lambda^2}{2}, & \text{if } |t| \geq \gamma\lambda \\ \gamma|t| - \frac{t^2}{2\gamma}, & \text{otherwise.} \end{cases}$$

where $[z]_+ = max(z, 0)$, λ is set to 1, and for simplicity denote $M_\gamma(\mathbf{B}) = M_{1,\gamma}(\mathbf{B})$.

The matrix γ-norm is defined as:

$$\begin{aligned} \|\mathbf{B}\|_\gamma &= \sum_{i=1}^{s} \int_0^{\sigma_i(\mathbf{B})} \left(1 - \frac{u}{\gamma}\right)_+ du \\ &= \sum_{i=1}^{s} \phi_{1,\gamma}(\sigma_i(\mathbf{B})) = M_\gamma(\sigma(\mathbf{B})), \ \gamma > 1 \end{aligned} \tag{10}$$

where $\sigma(\mathbf{B}) = (\sigma_1(\mathbf{B}), \ldots, \sigma_s(\mathbf{B}))^\top$ denotes a function from $\mathbb{R}^{p\times q}$ to $\mathbb{R}^s_{+,s} = \min(p, q)$. The matrix γ-norm is non-convex with respect to $\mathbf{B}$.

Furthermore, the dictionary is jointly learned from both auxiliary and target domains, in order to transfer useful knowledge from the auxiliary domain. As the source data set usually contains much more samples than target data set $\mathbf{X}_T$, the learning of dictionary is easily dominated by the source data. To emphasize the reconstruction power of $\mathbf{D}$ in the target domain, the authors propose to introduce an $l_{2,1}$ norm constraint on the source coefficient matrix $\mathbf{Z}_S$. In this way, some rows in $\mathbf{Z}_S$ are encouraged to be zero, which enables $\mathbf{X}_S$ to adaptively select bases from $\mathbf{D}$.

By replacing the rank function and l_0 norm with matrix γ-norm and MCP norm, and adding the $l_{2,1}$ norm constraint on source coefficient matrix, the objective function (8) can then be rewritten as:

$$\begin{aligned} \min_{\mathbf{D},\mathbf{Z}_S,\mathbf{Z}_T,\mathbf{E}_S,\mathbf{E}_T} \ & \|\mathbf{Z}_T\|_{\gamma 1} + \lambda_1 M_{\gamma 2}(\mathbf{E}_S) + \lambda_2 M_{\gamma 2}(\mathbf{E}_T) + \lambda_3\|\mathbf{Z}_S\|_{2,1} \\ & \text{s.t. } \mathbf{X}_S = \mathbf{D}\mathbf{Z}_S + \mathbf{E}_S, \ \ \mathbf{X}_T = \mathbf{D}\mathbf{Z}_T + \mathbf{E}_T \end{aligned} \tag{11}$$

where λ_3 is a trade-off parameter, and $\|\mathbf{Z}_S\|_{2,1} = \sum_{j=1}^{n}(\sum_{i=1}^{d}([\mathbf{Z}_S]_{ij})^2)^{1/2}$ is the $l_{2,1}$ norm. Each column in the learned coefficient matrix $\mathbf{Z}_T$ corresponds to one sample in the target domain, which is named *low-rank coding* of the corresponding sample.

The authors proposed a majorization-minimization augmented Lagrange multiplier (MM-ALM) algorithm to solve the problem (11). They presented two applications of this S-Low coding, one for clustering and the other for classification. Experimental results on five benchmark data sets demonstrated the effectiveness of the proposed algorithms compared with the state-of-the-art self-taught learning methods.

(Although there has not been much work studying unlabeled source data selection when learning a transferable feature extractor for the target task, we believe this should be a research direction worth to explore. Since data selection has already shown its effectiveness when transferring knowledge from labeled source data to target feature extractor, it should show equal importance when transferring knowledge from unlabeled source data to target feature extractor.)

3.3 Domain Adaptation: knowledge transfer from source data to target feature extractor and classifier parameters

A special research direction of knowledge transfer, which has been widely explored for a relatively long time, is the Domain Adaptation problem. In Domain Adaptation, we assume that the target data and source data share the same input and output spaces ($\mathcal{X}_T = \mathcal{X}_S$ and $\mathcal{Y}_T = \mathcal{Y}_S$), while having different but related data distributions ($P(X_T) \neq P(X_S)$ and $P(Y_T|X_T) \neq P(Y_S|X_S)$). Depending on whether labeled target training samples are available or not, DA problems could be further categorized into unsupervised Domain Adaptation (data available in training phase including: labeled source data and unlabeled target test data) and supervised Domain Adaptation (data available in training phase including: labeled source data, unlabeled target test data and a few labeled target training data). Whether supervised or unsupervised, the main goal of Domain Adaptation is to adapt source data distribution to target data distribution, so that the model parameters learned with both source and target data could show good performance for target task. Therefore we consider Domain

Adaptation as a kind of knowledge transfer from source raw data to target model parameters (instead of knowledge transfer from pre-learned model).

It is noteworthy that, in DA, especially in unsupervised DA, the training is usually done in a transductive manner. This means, unlabeled target test data is also involved in the training stage in order to achieve adaptation between source data distribution and target data distribution. This is not a common way, since in machine learning we usually assume test data not accessible during training phase. In reality, we also prefer a model trained with only training data, which allows us to apply the model directly on any future test data that follows the distribution assumption without re-training. However in transfer learning scenarios the situation may be a little different. As in most scenarios which knowledge transfer is needed, labeled target training data must be expensive to be got, that's the reason why we need to transfer knowledge from source task to help the learning of target task. In this case, transductive learning shows its advantage: by making use of the unlabeled target test data, one can get a more accurate estimation about the target data distribution instead of only rely on the rare target training data. The disadvantage of transductive learning is that, it would be more expensive to apply this kind of methods in reality, since it requires accessing test data in training phase, making it impossible to deliver a nicely pre-learned model for direct application on new data.

As we have talked about, the main goal of Domain Adaptation is adaptation between source and target data distribution. There exists various ways to achieve this distribution adaptation, in the following we show several groups of methods. The ultimate goal of adaptation is of course to adapt both marginal distributions ($P(X_S)$ and $P(X_T)$) and conditional distributions ($P(Y_S|X_S)$ and $P(Y_T|X_T)$), while some works may only focus on adapting one of the two, and others may focus on adapting both of them.

3.3.1 Learning a shared feature space with dimensionality reduction methods (Subspace Learning)

A natural way to achieve distribution adaptation is to learn a shared new feature space for source and target data, in which the distribution discrepancy between source and target feature data is minimized. A lot of DA works follow this way. Here we show a group of methods doing adaptation in this way, which all make use of dimensionality reduction methods for finding the new feature space.

3.3.1.1 *Adaptation with Bregman divergence based distance measure*: The first work is [17] where the authors proposed a Bregman Divergence based regularization schema for transfer subspace (representation) learning, which combines Bregman divergence with conventional dimensionality reduction algorithms. This regularized *subspace learning* learns a feature mapping and a classifier at the same time. The regularization term on the feature transformation parameters is based on a *bregman divergence* between the source marginal distribution and the target marginal distribution. Therefore the difference between the two marginal distributions will be explicitly reduced during optimization.

Specifically, assume some feature transformation $v^{\theta}(\mathbf{x}) = \theta\mathbf{x}$ where θ is a z by d matrix that maps the original d dimensional input feature vector into a new z dimensional feature space. In subspace learning framework we learn this matrix θ by minimizing a specific objective function $F(\theta)$:

$$\theta^* = \underset{\theta \in \mathbb{R}^{z\times d}}{\arg\min} F(\theta) \tag{12}$$

The objective function $F(\theta)$ is designed for specific applications, here it minimizes the data classification loss in the selected subspace according to different assumptions or intuitions. For example, Fisher's linear discriminant analysis (FLDA) selects a subspace, where the trace ratio of the within-class scatter matrix and the between-class scatter matrix is minimized.

To reduce the distribution difference between source and target data, the authors propose a Bregman divergence based regularization term $D_\theta(P_S \parallel P_T)$ which measure the distribution difference of samples drawn from different domains in the projected subspace θ. By integrating this regularization into (12), we obtain a new framework for transfer subspace learning (TSL):

$$\theta^* = \underset{\theta \in \mathbb{R}^{z\times d}}{\arg\min} F(\theta) + \lambda D_\theta(P_S \parallel P_T) \tag{13}$$

with respect to specific constraints, *e.g.*, $\theta^\top\theta = I$. Here λ is the regularization parameter that controls the trade-off between the two terms in (13).

Let $v : I \rightarrow I'$ be a C^1 convex function defined on a closed convex set $I \subset \mathbb{R}^+$. We denote the first order derivative of v by v', denote its inverse function by $\xi = (v')^{-1}$. The probability density for the source and target samples in the projected subspace I' is $P_S(v(\mathbf{x}))$ and $P_T(v(\mathbf{x}))$ respectively. The regularization term is defined as follows:

$$D_\theta(P_S \parallel P_T) = \int d(\xi(P_S(v(\mathbf{x}))), \xi(P_T(v(\mathbf{x}))))d\mu \tag{14}$$

where $d(\xi(P_S(v(\mathbf{x}))), \xi(P_T(v(\mathbf{x}))))$ is the difference at $\xi(P_T(v(\mathbf{x})))$ between the function v and the tangent line to v at point $(\xi(P_S(v(\mathbf{x}))), v(\xi(P_S(v(\mathbf{x})))))$, $d\mu$ (*i.e.*, $du(v(\mathbf{x}))$) is the Lebesgue measure. The right hand side of (14) is also called the U-divergence on the subspace $\mathbb{R}^d$.

The authors show examples of this transfer subspace learning framework using different $F(\theta)$ (*i.e.* combining with different dimensionality reduction methods), such as transfered principal components analysis (TPCA), transfered Fisher's linear discriminant analysis (TFLDA), transfered locality preserving projections (TLPP) with supervised setting, *etc.* They present experimental evidence on both face image data sets and text data sets, suggesting that the proposed framework is effective to deal with cross-domain learning problems.

3.3.1.2 *Adaptation with Maximum Mean Discrepancy (MMD) as distance measure*: Similar to the previous approach, in [18] the authors proposed a transfer learning algorithm which also combines conventional dimensionality reduction method and a distance measure for measuring the distance between marginal distributions of source data and target data. In this work the authors make use of the Maximum Mean Discrepancy as distribution distance measure, and PCA as the dimensionality reduction method.

Maximum Mean Discrepancy (MMD) is a two samples test criterion for comparing distributions based on Reproducing Kernel Hilbert Space (RKHS). Let $X = \{x_1, \dots, x_{n_1}\}$ and $Y = \{y_1, \dots, y_{n_2}\}$ be two random variable sets from distributions $\mathcal{P}$ and $\mathcal{Q}$, respectively, and $\mathcal{H}$ be a universal RKHS with the reproducing kernel mapping ϕ: $f(x) = \langle \phi(x), f \rangle$, $\phi : \mathcal{X} \to \mathcal{H}$. The empirical estimate of distance between $\mathcal{P}$ and $\mathcal{Q}$ defined by MMD is as follows:

$$Dist(X, Y) = \parallel \frac{1}{n_1} \sum_{i=1}^{n_1} \phi(x_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(y_i) \parallel_{\mathcal{H}} \tag{15}$$

As can be seen, the MMD between two sample sets is equivalent to the distance between the means of the two sample sets mapped into a RKHS. Based on this, the authors proposed a new dimensionality reduction method, denoted as MMDE (Maximum Mean Discrepancy Embedding), to learn a low-dimensional latent space F common to source and target domains. A classifier is then learned in this latent space with source labeled data, and this learned classifier is directly used for target classification task (*i.e.*, they assume that in the latent space the conditional distributions of source data and target data are the same).

Denote the source domain data as $\mathcal{D}_{src} = \{(x_1^{src}, y_1^{src}), \dots, (x_{n_1}^{src}, y_{n_1}^{src})\}$, where $x_i^{src} \in \mathbb{R}^m$ is the input sample feature and y_i^{src} the corresponding label. Similarly, denote the target domain data as $\mathcal{D}_{tar} = \{(x_1^{tar}, y_1^{tar}), \dots, (x_{n_2}^{tar}, y_{n_2}^{tar})\}$ with $x_i^{tar} \in \mathbb{R}^m$. Let the feature projection map be ψ. Then learning a common low-dimensional latent space in which the distributions of the source and target data (*i.e.*, X'_{src} and X'_{tar}) can be close to each other is equivalent to minimizing the MMD between X'_{src} and X'_{tar}:

$$\begin{aligned} Dist(X'_{src}, X'_{tar}) &= Dist(\psi(X_{src}), \psi(X_{tar})) \\ &= \parallel \frac{1}{n_1} \sum_{i=1}^{n_1} \phi \circ \psi(x_i^{src}) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi \circ \psi(x_i^{tar}) \parallel \end{aligned} \tag{16}$$

Denote the corresponding kernel of $\phi \circ \psi$ by k, then equation (16) can be written in terms of the kernel matrices defined by k as:

$$Dist(X'_{src}, X'_{tar}) = \text{trace}(KL) \tag{17}$$

where

$$K = \begin{bmatrix} K_{src,src} & K_{src,tar} \\ K_{tar,src} & K_{tar,tar} \end{bmatrix} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$$

is a composite kernel matrix, and $L = [L_{ij}] \succeq 0$ with

$$L_{ij} = \begin{cases} \frac{1}{n_1^2} & x_i, x_j \in X_{src}, \\ \frac{1}{n_2^2} & x_i, x_j \in X_{tar}, \\ -\frac{1}{n_1 n_2} & \text{otherwise}. \end{cases}$$

The authors proved that this kernel matrix K correspond to an universal kernel, so we can learn this kernel matrix instead of learning the universal kernel k. Thus, the embedding problem can be formulated as the following optimization problem:

$$\begin{aligned} \min_{K = \widetilde{K} + \varepsilon I} \quad & \text{trace}(KL) - \lambda \text{trace}(K) \\ \text{s.t.} \quad & K_{ii} + K_{jj} - 2K_{ij} = d_{ij}^2, \ \forall (i,j) \in \mathcal{N}, \\ & K\mathbf{1} = \mathbf{0}, \ \widetilde{K} \succeq 0. \end{aligned} \tag{18}$$

where ε is a small positive constant and $\mathbf{1}$ and $\mathbf{0}$ are the vectors of ones and zeros, respectively. The second term is added to unfold the high dimensional data by maximizing the trace of K. This optimization problem can be further rewritten as a semidefinite program (SDP) which learns $\widetilde{K}$ and can be solved by standard SDP solvers. After obtaining $\widetilde{K}$, the authors then apply PCA and select the leading eigen vectors to construct low-dimensional representations X'_{src} and X'_{tar}. A classifier is learned with X'_{src} and Y_{src} and is applied directly for classification in target domain.

The authors perform experiments on indoor WiFi localization dataset and text classification dataset, the results showed that the proposed MMDE can effectively improve the performance compared to traditional machine learning algorithms.

3.3.1.3 *Transfer Component Analysis*: The previous MMDE suffers from two major limitations: (1) it does not generalize to out-of-sample patterns; (2) it learns the latent space by solving a semi-definite program (SDP), which is a very expensive optimization problem. Furthermore, in order to construct low-dimensional representations, in MMDE the obtained K has to be further post-processed by PCA, this step may discard potentially useful information in K. To get ride of these limitations, the authors further proposed in [19] a new approach, named *transfer component analysis* (TCA), which tries to learn a set of common *transfer components* underlying both domains such that the difference in data distributions in the new subspace of two domains can be reduced, and data properties can be preserved.

Unlike MMDE, this proposed TCA avoids the use of SDP and can be generalized to out-of-sample patterns. Besides, instead of using a two-step approach, they propose a unified kernel learning method which utilizes an explicit low-rank representation.

First note that the kernel matrix K defined in (17) can be decomposed as $K = (KK^{-1/2})(K^{-1/2}K)$, which is often known as the empirical kernel map. Consider the use of a matrix $\widetilde{W} \in \mathbb{R}^{(n_1+n_2)\times m}$ that transforms the empirical kernel map features to an m-dimensional space (where $m \ll n_1 + n_2$). The resultant kernel matrix is then

$$\widetilde{K} = (KK^{-1/2}\widetilde{W})(\widetilde{W}^\top K^{-1/2}K) = KWW^\top K \tag{19}$$

where $W = K^{-1/2}\widetilde{W}$. This kernel $\widetilde{K}$ facilitates a readily parametric form for out-of-sample kernel evaluations. On using the definition of $\widetilde{K}$ in (19), the MMD distance between the two domains X'_{src} and X'_{tar} can be written as:

$$Dist(X'_{src}, X'_{tar}) = \text{trace}((KWW^\top K)L) = \text{trace}(W^\top KLKW) \tag{20}$$

In minimizing (20), a regularization term trace$(W^\top W)$ is usually needed to control the complexity of W. This regularization term can also avoid the rank deficiency of the denominator in the generalized eigenvalue decomposition.

Besides reducing the distance between the two marginal distributions, the projection ϕ should also preserve data properties that are useful for the target supervised learning task. As in PCA or KPCA, this can be done by preserving the data variance. Therefore by combining the minimization of distribution difference and preserving of the data variance, the kernel learning problem becomes:

$$\begin{aligned} \min_{W} \; & \text{trace}(W^\top KLKW) + \mu \, \text{trace}(W^\top W) \\ \text{s.t.} \; & W^\top KHKW = I_m \end{aligned} \tag{21}$$

where $\mu > 0$ is a trade-off parameter, $W^\top KHKW$ is the variance of the projected samples with $H = I_{n_1+n_2} - (1/n_1+n_2)\mathbf{1}\mathbf{1}^\top$ the centering matrix, $\mathbf{1} \in \mathbb{R}^{n_1+n_2}$ is the column vector with all 1's, and $I_{n1+n2} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$ $I_m \in \mathbb{R}^{m\times m}$ are identity matrix. Though this optimization problem involves a non-convex norm constraint $W^\top KHKW = I_m$, the authors proved that it can still be solved efficiently by the following trace optimization problem:

$$\max_{W} \; \text{trace}((W^\top(KLK + \mu I_m)W)^{-1} W^\top KHKW) \tag{22}$$

Similar to kernel Fisher discriminant analysis [20], the W solutions in (22) are the m leading eigenvectors of $(KLK + \mu I)^{-1}KHK$, where $m \leqslant n_1 + n_2 - 1$. This unsupervised approach is named TCA (Transfer Component Analysis), based on this, the authors also proposed a semi-supervised version SSTCA. The effectiveness and efficiency of TCA and SSTCA are verified by experiments on five toy datasets and two real-world applications: cross-domain indoor WiFi localization and cross-domain text classification.

3.3.1.4 *Joint Distribution Adaptation (JDA)*: The previous three works all focus on minimizing the marginal distributions' discrepancy between source and target data, while assuming that the conditional distributions of source and target data in the learned novel feature space are equal so that a classifier learned on source data can be directly applied to target data. However, such equality assumption of conditional distributions is strong and cannot always be respected. In [21], Long *et al.* proposed a *Joint Distribution Adaptation* (JDA), which aims to jointly adapt both the marginal and conditional distributions in a principled dimensionality reduction procedure. Similar to previously introduced MMDE and TCA, JDA also makes use of *Maximum Mean Discrepancy* as the distance measurement between distributions.

Let $(\mathbf{x}_s, y_s)$ be a labeled sample from the source training set, $\mathbf{x}_t$ an unlabeled sample from the target training set, $P(\mathbf{x}_s)$ and $P(\mathbf{x}_t)$ the marginal distributions, and $P(y_s|\mathbf{x}_s)$ and $P(y_t|\mathbf{x}_t)$ the conditional distributions of source and target data, respectively. Long *et al.* propose to adapt the joint distributions by a feature transformation T so that the joint expectations of features $\mathbf{x}$ and labels y are matched between source and target domains:

$$\begin{aligned} \min_{T} \; & \| \mathbb{E}_{P(\mathbf{x}_s,y_s)}[T(\mathbf{x}_s), y_s] - \mathbb{E}_{P(\mathbf{x}_t,y_t)}[T(\mathbf{x}_t), y_t] \|^2 \\ \approx \; & \| \mathbb{E}_{P(\mathbf{x}_s)}[T(\mathbf{x}_s)] - \mathbb{E}_{P(\mathbf{x}_t)}[T(\mathbf{x}_t)] \|^2 \\ + \; & \| \mathbb{E}_{P(y_s|\mathbf{x}_s)}[y_s|T(\mathbf{x}_s)] - \mathbb{E}_{P(y_t|\mathbf{x}_t)}[y_t|T(\mathbf{x}_t)] \|^2 \end{aligned} \tag{23}$$

Similar to TCA, the authors first use PCA and MMD to align the marginal distributions. Denote $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ the input data matrix, and $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}$ the centering matrix, where $n = n_s + n_t$ and $\mathbf{1}$ the $n \times n$ matrix of ones, then the co-variance matrix can be computed as $\mathbf{XHX}^\top$. The learning goal of PCA is to find an orthogonal transformation matrix $\mathbf{A} \in \mathbb{R}^{m\times k}$ such that the embedded data variance is maximized:

$$\max_{\mathbf{A}^\top\mathbf{A}=\mathbf{I}} \; \text{trace}(\mathbf{A}^\top \mathbf{XHX}^\top \mathbf{A}) \tag{24}$$

To reduce the difference between marginal distributions, the empirical MMD, which computes the distance between the sample means of the source and target data in the k-dimensional embeddings, should be minimized:

$$\begin{aligned} \text{Dist}(P(\mathbf{x}_s), P(\mathbf{x}_t)) &= \| \frac{1}{n_s}\sum_{i=1}^{n_s} \mathbf{A}^\top \mathbf{x}_i - \frac{1}{n_t}\sum_{j=n_s+1}^{n_s+n_t} \mathbf{A}^\top \mathbf{x}_j \|^2 \\ &= \text{trace}(\mathbf{A}^\top \mathbf{X}\mathbf{M}_0\mathbf{X}^\top \mathbf{A}) \end{aligned} \tag{25}$$

where $\mathbf{M}_0$ is the MMD matrix and is computed as:

$$(M_0)_{ij} = \begin{cases} \frac{1}{n_s n_s}, & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_s \\ \frac{1}{n_t n_t}, & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_t \\ \frac{-1}{n_s n_t}, & \text{otherwise.} \end{cases} \tag{26}$$

By minimizing Eq.(25) such that Eq.(24) is maximized, the marginal distributions between domains are drawn close under the new representation $\mathbf{Z} = \mathbf{A}^\top \mathbf{X}$.

In [21], Long *et al.* deals with the problem of *unsupervised* domain adaptation and thereby assume that no labeled sample is provided in target training set. As a result, to reduce the mismatch of conditional distributions, the authors propose to explore the *pseudo* labels of the target data, which can be predicted by applying some base classifiers trained on the labeled source data to the unlabeled target data. Furthermore, Long *et al.* propose to explore the sufficient statistics of class-conditional distributions $P(\mathbf{x}_s|y_s)$ and $P(\mathbf{x}_t|y_t)$ instead of the posterior probabilities $P(y_s|\mathbf{x}_s)$ and $P(y_t|\mathbf{x}_t)$. With the true labels on the source data and pseudo labels on the target data, we can match the class-conditional distributions $P(\mathbf{x}_s|y_s = c)$ and $P(\mathbf{x}_t|y_t = c)$ *w.r.t* each class $c \in \{1, \ldots, C\}$ in the label set $\mathcal{Y}$. The authors further modify MMD to compute the distance between the class-conditional distributions:

$$\begin{aligned} & \text{Dist}(P(\mathbf{x}_s|y_s = c), P(\mathbf{x}_t|y_t = c)) \\ & \quad = \| \frac{1}{n_s^{(c)}} \sum_{\mathbf{x}_i \in \mathcal{D}_s^{(c)}} \mathbf{A}^\top \mathbf{x}_i - \frac{1}{n_t^{(c)}} \sum_{\mathbf{x}_j \in \mathcal{D}_t^{(c)}} \mathbf{A}^\top \mathbf{x}_j \|^2 \\ & \quad = \text{trace}(\mathbf{A}^\top \mathbf{X}\mathbf{M}_c\mathbf{X}^\top \mathbf{A}) \end{aligned} \tag{27}$$

where $\mathcal{D}_s^{(c)} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathcal{D}_s \wedge y(\mathbf{x}_i) = c\}$ is the set of data samples belonging to class c in the source data, $y(\mathbf{x}_i)$ is the true label of $\mathbf{x}_i$, and $n_s^{(c)} = |\mathcal{D}_s^{(c)}|$. Correspondingly, $\mathcal{D}_t^{(c)} = \{\mathbf{x}_j \mid \mathbf{x}_j \in \mathcal{D}_t \wedge \widehat{y}(\mathbf{x}_j) = c\}$ is the set of data samples belonging to class c in the target data, $\widehat{y}(\mathbf{x}_j)$ is the pseudo (predicted) label of $\mathbf{x}_j$, and $n_t^{(c)} = |\mathcal{D}_t^{(c)}|$. Thus the MMD matrices $\mathbf{M}_c$ involving class labels are computed as follows:

$$(M_c)_{ij} = \begin{cases} \frac{1}{n_s^{(c)} n_s^{(c)}}, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_s^{(c)} \\ \frac{1}{n_t^{(c)} n_t^{(c)}}, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_t^{(c)} \\ \frac{-1}{n_s^{(c)} n_t^{(c)}}, & \begin{cases} \mathbf{x}_i \in \mathcal{D}_s^{(c)}, \mathbf{x}_j \in \mathcal{D}_t^{(c)} \\ \mathbf{x}_i \in \mathcal{D}_t^{(c)}, \mathbf{x}_j \in \mathcal{D}_s^{(c)} \end{cases} \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

By minimizing Eq. (27) such that Eq.(24) is maximized, the conditional distribution between domains are drawn close under the new representation $\mathbf{Z} = \mathbf{A}^\top \mathbf{X}$. To achieve effective and robust transfer learning, JDA simultaneously minimizes the differences in both the marginal distributions and conditional distributions across domains. By incorporating Eq.(25) and Eq.(27) into Eq.(24), JDA is formulated as the following optimization problem:

$$\min_{\mathbf{A}^\top \mathbf{X}\mathbf{H}\mathbf{X}^\top \mathbf{A}=\mathbf{I}} \sum_{c=0}^{C} \text{trace}(\mathbf{A}^\top \mathbf{X}\mathbf{M}_c \mathbf{X}^\top \mathbf{A}) + \lambda \|\mathbf{A}\|_F^2 \tag{29}$$

where λ is a regularization parameter. Based on the generalized Rayleigh quotient, minimizing Eq.(25) and Eq.(27) such that Eq.(24) is maximized is equivalent to minimizing Eq.(25) and Eq.(27) such that Eq.(24) is fixed. As a result, the previously introduced TCA can be viewed as a special case of JDA with $C = 0$.

For nonlinear problems, consider kernel mapping $\psi : \mathbf{x} \to \psi(\mathbf{x})$, and kernel matrix $\mathbf{K} = \psi(\mathbf{X})^\top \psi(\mathbf{X}) \in \mathbb{R}^{n \times n}$, They use the Representer theorem to formulate the Kernel-JDA as:

$$\min_{\mathbf{A}^\top \mathbf{K}\mathbf{H}\mathbf{K}^\top \mathbf{A}=\mathbf{I}} \sum_{c=0}^{C} \text{trace}(\mathbf{A}^\top \mathbf{K}\mathbf{M}_c \mathbf{K}^\top \mathbf{A}) + \lambda \|\mathbf{A}\|_F^2 \tag{30}$$

The problem of finding an optimal adaptation matrix $\mathbf{A}$ as defined in Eq.(29) can be reformulated as a generalized eigen decomposition problem. The authors proposed an iterative approach where at each iteration they first solve the generalized eigen decomposition problem, then use the novel adaptation matrix $\mathbf{A}$ to get the new features and train a standard classifier. The pseudo labels on target data are then updated using this novel classifier to form a new eigen decomposition problem which can be solved at the next iteration. This EM-like pseudo label refinement procedure continues until convergence of the pseudo labels.

The authors performed experiments for image classification problems to evaluate the JDA approach. The results verified that JDA outperforms other state of the art DA methods by that time, including in particular TCA, on four types of cross-domain image classification problems.

3.3.1.5 *Close yet Discriminative Domain Adaptation (CDDA) and Discriminative and Geometry Aware Domain Adaptation (DGA-DA)*: A cornerstone theoretical result in DA [22], [23] is achieved by Ben-David *et al.*, who estimate an error bound of a learned hypothesis h on a target domain:

$$\begin{aligned} e_{\mathcal{T}}(h) \le e_{\mathcal{S}}(h) + d_{\mathcal{H}}(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}) + \\ \min \{\mathcal{E}_{\mathcal{D}_{\mathcal{S}}} [|f_{\mathcal{S}}(\mathbf{x}) - f_{\mathcal{T}}(\mathbf{x})|], \mathcal{E}_{\mathcal{D}_{\mathcal{T}}} [|f_{\mathcal{S}}(\mathbf{x}) - f_{\mathcal{T}}(\mathbf{x})|]\} \end{aligned} \tag{31}$$

Eq.(31) provides insight on the way to improve DA algorithms as it states that the performance of a hypothesis h on a target domain is determined by: 1) the classification error on the source domain $e_{\mathcal{S}}(h)$; 2) data divergence $d_{\mathcal{H}}(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}})$ which measures the $\mathcal{H}$*-divergence*[23] between two distributions($\mathcal{D}_{\mathcal{S}}$, $\mathcal{D}_{\mathcal{T}}$); 3) the difference in labeling functions across the two domains.

- Close yet Discriminative Domain Adaptation

Inspired by this theoretical analysis [22], [23], Luo *et al.*argue in [24] that previous research TCA[19], JDA[21] only seek to minimize the second term of Eq.(31) in reducing data distribution discrepancies, and ignore the discriminative knowledge which exist between sub-domains with different labels. Thus, in [24], they argue that an effective DA method should search a shared feature subspace where source and target data are not only aligned in terms of distributions as most state of the art DA methods do, *e.g.*, TCA, JDA, but also *discriminative* in that instances of different classes are well separated, and designed an effective DA method, namely *Close yet Discriminative Domain Adaptation* **(CDDA)**.

Similar to JDA, CDDA also tries to minimize Eq.(23) via solving generalized Rayleigh quotient optimization as defined in Eq.(29). To explicitly render data discriminative in the searched feature subspace, CDDA introduces a *Discriminative* domain adaption via a *repulsive force* term, so as to increase the distances of sub-domains with different labels, and improve the discriminative power of the latent shared features, thereby making it possible for a better predictive model for both the source and target data.

- Discriminative and Geometry Aware Domain Adaptation

In [24] , Luo *et al.* further point out that traditional data distribution centered DA methods are weak to preserve the hidden data geometric structure across different domains. Nevertheless, geometric alignment of the underlying data manifold structures across source and target domains is important as it ends to enhance the labeling functions between source and target, thereby potentially optimizing the third term of Eq.(31). To jointly consider CDDA and the hidden geometric structure, Luo *et al.* also propose in [24] a novel DA method, namely *Discriminative and Geometry Aware Domain Adaptation* **(DGA-DA)**.

DGA-DA builds upon CDDA and introduces two additional consistency constraints, namely *label smoothness consistency* and *geometric structure consistency* for both the *pseudo* and final label inference, in order to account for the underlying data manifold structure in data similarity measurement.

It is worth noting that, both CDDA and DGA-DA could solve nonlinear problems through kernelization similar to Eq.(30). The authors performed extensive experiments on 49 image classification DA tasks through 8 popular DA benchmarks and verify the effectiveness of the proposed method. They also carried out in-depth analysis of the proposed DA methods, in particular *w.r.t.* their hyper-parameters and convergence speed. In addition, using both synthetic and real data, the authors also provide insights into the proposed DA model in visualizing the effect of data discriminativeness and geometry awareness.

3.3.2 Learning a shared feature space with metric learning

An alternative way to learn a shared feature space is to cast the representation learning problem into the metric learning scenario. For example, Ding and Fu develop a robust transfer metric learning (RTML) framework in [25] to assist the unlabeled target learning by transferring knowledge from labeled source data.

Given a source domain X_s with n_s labeled samples from c classes: $X_s = \{(x_{s,1}, y_{s,1}), \ldots, (x_{s,n_s}, y_{s,n_s})\}$, and X_t a target domain with n_t unlabeled samples: $X_t = \{x_{t,1}, \ldots, x_{t,n}\}$, where $x_{s,i} \in \mathbb{R}^d$ with $y_{s,i} \in [1, c]$ its class label and $x_{t,i} \in \mathbb{R}^d$.

The objective of the proposed Robust Transfer Metric Learning (RTML) is as follows:

$$\min_{\mathcal{M} \in \mathbb{S}^d_+} \sum_{i=0}^{c} \mathrm{tr}(\Phi^i \mathcal{M}) + \alpha \|\bar{X} - \mathcal{M}\tilde{X}\|_F^2 + \lambda \sum_{i=r+1}^{d} (\sigma_i(\mathcal{M}))^2 \quad (32)$$

where $\Phi^i = (\mu_s^i - \mu_t^i)(\mu_s^i - \mu_t^i)^\top$, μ_s^i is the mean of i-th class in the source domain and μ_t^i the mean of i-th class in the pseudo labeled target domain. $\mathcal{M} \in \mathbb{R}^{d \times d}$ is the positive semi-definite ($\mathcal{M} \in \mathbb{S}^d_+$) distance metric to be learned.

The second term in Eq.(32) is a denoising term for preserving energy of the two domains. Let $X = [X_s, X_t] \in \mathbb{R}^{d \times n}$, $(n = n_s + n_t)$, $\bar{X}$ is m-times repeated version of X, and $\tilde{X}$ is the corrupted version of $\bar{X}$ with different ratios of random corruption. (See marginalized Denoising Auto-Encoder (mDAE) [26] and Denoising Auto-Encoder (DAE) [27]) In this way, the learned metric could not only have denoising property but also reduce the dimensionality of the original data to save computational cost.

The last term is a rank control regularization term, which minimizes the sum of the $d - r$ smallest eigenvalues of $\mathcal{M}$, so that this term would reach its minimum when the rank of $\mathcal{M}$ is less or equal to a pursued rank r. $\sigma_i(\mathcal{M})$ is the i-th eigenvalue of $\mathcal{M}$. $\sum_{i=r+1}^{d}(\sigma_i(\mathcal{M}))^2 = \mathrm{tr}(\Gamma^\top \mathcal{M}\mathcal{M}^\top \Gamma)$, where Γ are the singular vectors which correspond to the $(d-r)$-smallest singular values of $\mathcal{M}\mathcal{M}^\top$.

The optimization is performed as follows: $\mathcal{M}$, Γ and the pseudo labels of the target data are refined iteratively, *i.e.* optimizing one while fixing the others, until the metric $\mathcal{M}$ converges. Since $\mathcal{M}$ is positive semi-definite, it cannot be optimized directly, they adopt a linear approximation to Eq.(32) to solve the problem. The complexity of the resulting algorithm is bounded with $\mathcal{O}\left(T\left((l+2)d^3 + cn^2 + d\right)\right)$ where l is the number of matrix multiplications at each iteration.

Since $\mathcal{M}$ can be decomposed into $\mathcal{M} = PP^\top$, where $P \in \mathbb{R}^{d \times r}$ and $r \leqslant d$ is the rank of metric $\mathcal{M}$, $d_{\mathcal{M}}(x_i, x_j)$ can be rewritten as $\|P^\top(x_i - x_j)\|_2$, which builds the connection between metric learning and subspace learning.

3.3.3 Learning a shared feature space with Deep Neural Networks

The last years have seen breakthroughs enabled by deep learning in an increasing number of domains, in particular for various vision recognition tasks. Recent studies show that deep neural networks (DNN) can also learn transferable feature which generalize well to novel tasks. Therefore, more and more works tend to rely upon Deep Neural Networks to solve DA problems. In the following we show some works making use of DNN to learn a shared feature space for source and target data. The basic idea is similar to those introduce previously in Sect.3.3.1 and Sect.3.3.2. They make use of DNN as the structure for feature learning, and they add regularization on the new feature space either by adding one or multiple adaptation layers through which the embeddings of distributions are matched between source and target [28] [29], or by adding a fully connected subnetwork as a domain discriminator whilst deep features are learned to confuse the domain discriminator in a domain-adversarial training paradigm [30] [31]. We introduce in detail two representative works in the subsequent subsections.

3.3.3.1 *Deep Adaptation Networks*: In [29], Long *et al.* proposed a Deep Adaptation Network (DAN) architecture, which generalizes deep Convolutional Neural Networks (CNN) to the domain adaptation scenario. Similar to previously introduced JDA in Sect.3.3.1, DAN also uses Maximum Mean Discrepancy (MMD) as distance measurement for adapting source and target distributions. Specifically, they used a multi-kernel version of MMD, named MK-MMD as proposed by [32], which is formalized to jointly maximize the two-sample test power and minimize the Type II error, *i.e.*, the failure of rejecting a false null hypothesis.

Given a set of labeled source training samples $\{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and a set of unlabeled target training samples $\{\mathbf{x}_j^t\}_{j=1}^{n_t}$, let $\mathcal{H}_k$ be the reproducing kernel Hilbert space (RKHS) endowed with a characteristic kernel k. The *mean embedding* of distribution p in $\mathcal{H}_k$ is a unique element $\mu_k(p)$ such that $\mathbf{E}_{\mathbf{x}\sim p} f(\mathbf{x}) = \langle f(\mathbf{x}), \mu_k(p)\rangle_{\mathcal{H}_k}$ for all $f \in \mathcal{H}_k$. The MK-MMD $d_k(p,q)$ between probability distributions p and q is defined as the RKHS distance between the mean embeddings of p and q. The squared formulation of MK-MMD is defined as:

$$d_k^2(p,q) \triangleq \|\mathbf{E}_p[\phi(\mathbf{x}^s)] - \mathbf{E}_q[\phi(\mathbf{x}^t)]\|_{\mathcal{H}_k}^2 \quad (33)$$

A property of this distance is that $p = q$ if and only if $d_k^2(p,q) = 0$. The characteristic kernel k associated with the feature map ϕ is defined as the convex combination of m PSD (Positive Semi-Definite) kernels $\{k_u\}_{u=1}^m$. As studied theoretically in [32], the kernel adopted for the mean embeddings of p and q is critical to ensure the test power and low test error. The multi-kernel k can leverage different kernels to enhance MK-MMD test, leading to a principled method for optimal kernel selection.

Using the kernel trick, MK-MMD as in Eq.(33) can be computed as the expectation of kernel functions:

$$d_k^2(p,q) = \mathbf{E}_{\mathbf{x}^s\mathbf{x}'^s} k(\mathbf{x}^s, \mathbf{x}'^s) + \mathbf{E}_{\mathbf{x}^t\mathbf{x}'^t} k(\mathbf{x}^t, \mathbf{x}'^t) - 2\mathbf{E}_{\mathbf{x}^s\mathbf{x}^t} k(\mathbf{x}^s, \mathbf{x}^t) \quad (34)$$

where $\mathbf{x}^s, \mathbf{x}'^s \sim p$, $\mathbf{x}^t, \mathbf{x}'^t \sim q$, and $k \in \mathcal{K}$. However, this computation incurs a complexity of $O(n^2)$ which is undesirable for deep CNNs. Therefore, Long *et al.* propose to adopt the unbiased estimate of MK-MMD [32] which can be computed with linear complexity $O(n)$. Specifically,

$$d_k^2(p,q) = \frac{2}{n_s} \sum_{i=1}^{n_s/2} g_k(\mathbf{z}_i) \quad (35)$$

where $\mathbf{z}_i$ is a quad-tuple defined as: $\mathbf{z}_i \triangleq (\mathbf{x}^s_{2i-1}, \mathbf{x}^s_{2i}, \mathbf{x}^t_{2i-1}, \mathbf{x}^t_{2i})$, and the multi-kernel function k on each quad-tuple $\mathbf{z}_i$ is evaluated by:

$$g_k(\mathbf{z}_i) \triangleq k(\mathbf{x}^s_{2i-1}, \mathbf{x}^s_{2i}) + k(\mathbf{x}^t_{2i-1}, \mathbf{x}^t_{2i}) - k(\mathbf{x}^s_{2i-1}, \mathbf{x}^t_{2i}) - k(\mathbf{x}^s_{2i}, \mathbf{x}^t_{2i-1}) \tag{36}$$

To learn an optimal feature transformation, the authors propose to extend the AlexNet architecture [33], which is comprised of five convolutional layers (*conv1 – conv5*) and three fully connected layers (*fc6 – fc8*). Each *fc* layer ℓ learns a nonlinear mapping $\mathbf{h}^\ell_i = f^\ell(\mathbf{W}^\ell \mathbf{h}^{\ell-1}_i + \mathbf{b}^\ell)$, where $\mathbf{h}^\ell_i$ is the ℓ-th layer hidden representation of point $\mathbf{x}_i$, $\mathbf{W}^\ell$ and $\mathbf{b}^\ell$ are the weights and bias of the ℓ-th layer, and f^ℓ is the activation, taking as rectifier units $f^\ell(\mathbf{x}) = max(\mathbf{0}, \mathbf{x})$ for hidden layers or softmax units $f^\ell(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\sum_{j=1}^{|\mathbf{x}|} e^{x_j}}$ for the output layer. Letting $\Theta = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}^l_{\ell=1}$ denote the set of all CNN parameters, the objective of learning a CNN is to minimize its empirical risk:

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(\mathbf{x}^a_i), y^a_i) \tag{37}$$

where J is the cross-entropy loss function, and $\theta(\mathbf{x}^a_i)$ is the conditional probability that the CNN assigns $\mathbf{x}^a_i$ to label y^a_i. Given that the convolutional layers can learn generic features that tend to be transferable in layers *conv1 – conv3* and are slightly domain-biased in *conv4 – conv5*, while the higher layers *fc6 – fc8* are more domain specific which cannot be directly transferred to the target domain via fine-tuning with limited target supervision [2], the authors therefore propose to freeze *conv1-conv3* and fine-tune *conv4 – conv5* to preserve the efficacy of fragile co-adaptation, while retrain the *fc6 – fc8* layers' parameters in requiring the distributions of the source and target to become similar under the hidden representations of these fully connected layers. This can be realized by adding an MK-MMD-based multi-layer adaptation regularizer , *i.e.*, Eq.(33), to the CNN risk in Eq. (37):

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(\mathbf{x}^a_i), y^a_i) + \lambda \sum_{\ell=l_1}^{l_2} d^2_k(\mathcal{D}^\ell_s, \mathcal{D}^\ell_t) \tag{38}$$

where $\lambda > 0$ is a penalty parameter, l_1 and l_2 are layer indexes between which the regularizer is effective, in DAN the authors set $l_1 = 6$ and $l_2 = 8$. $\mathcal{D}^\ell_* = \{\mathbf{h}^{*\ell}_i\}$ is the ℓ-th layer hidden representation for the source and target examples, and $d^2_k(\mathcal{D}^\ell_s, \mathcal{D}^\ell_t)$ is the MK-MMD between the source and target evaluated on the ℓ-th layer representation.

The authors propose to initialize the parameters in DAN with those of the AlexNet model pre-trained on ImageNet 2012. The training process is therefore a fine-tuning of this pre-trained model on source and target training data.

They perform experiments of DAN compared to other transfer learning methods and deep learning methods on both unsupervised and semi-supervised adaptation problems. The results verified the efficacy of the proposed DAN against previous methods.

This work is further improved in [34], in which the authors proposed a Residual Transfer Network which can adapt both marginal distributions and conditional distributions at the same time.

3.3.3.2 *Adaptation with Adversarial Networks*: Ganin and Lempitsky proposed in [30] a novel approach to domain adaptation using deep architectures. As the training progresses, the approach promotes the emergence of *deep* features that are (i) discriminative for the main learning task on the source domain and (ii) invariant with respect to the shift between the domains. This adaptation behavior is achieved by augmenting a feed-forward model with few standard layers and a new *gradient reversal* layer. The resulting augmented architecture can be trained using standard back-propagation.

Assume that the model works with input samples $\mathbf{x} \in X$ where X is some input space and certain labels (output) y from the label space Y. They consider classification problems where Y is a finite set $Y = \{1, 2, \ldots, L\}$ in the paper, although they claim that their approach is generic and can handle any output label space that other deep feed-forward models can handle. They further assume that there exist two distributions $\mathcal{S}(x, y)$ and $\mathcal{T}(x, y)$ on $X \otimes Y$, which will be referred to as the source and the target distribution, respectively. Both distributions are assumed complex and unknown, and similar but different (in other words, $\mathcal{S}$ is "shifted" from $\mathcal{T}$ by some *domain shift*). Their ultimate goal is to be able to predict labels y given the input $\mathbf{x}$ for the target distribution. At training time, the model has access to a large set of training samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ from both the source and the target domains distributed according to the marginal distributions $\mathcal{S}(\mathbf{X})$ and $\mathcal{T}(\mathbf{x})$, respectively. They denote with d_i, which is a binary variable, the *domain label* for the i-th sample, indicating whether x_i come from the source distribution ($d_i = 0$) or from the target one ($d_i = 1$). It is assumed that the samples from the source domain are given category labels $y_i \in Y$.

They define a deep feed-forward architecture that for each input $\mathbf{x}$ predicts its label $y \in Y$ and its domain label $d \in \{0, 1\}$. They decompose such mapping into three parts. They assume that the input $\mathbf{x}$ is first mapped by a mapping G_f (a *feature extractor*) to a D-dimensional feature vector $\mathbf{f} \in \mathbb{R}^D$. The feature mapping may also include several feed-forward layers and we denote the vector of parameters of all layers in this mapping as θ_f, *i.e.* $\mathbf{f} = G_f(\mathbf{x}; \theta_f)$. Then, the feature vector $\mathbf{f}$ is mapped by a mapping G_y (*label predictor*) to the label y, and they denote the parameters of this mapping with θ_y. Finally, the same feature vector $\mathbf{f}$ is mapped to the domain label d by a mapping G_d (*domain classifier*) with the parameters θ_d.

During the learning stage, they aim to minimize the label prediction loss on the annotated part (*i.e.* the source part) of the training set, and the parameters of both the feature extractor and the label predictor are thus optimized in order to minimize the empirical loss for the source domain samples. This ensures the discriminativeness of the features $\mathbf{f}$ and the overall good prediction performance of the combination of the feature extractor and the label predictor on the source domain.

At the same time, they want to make the features $\mathbf{f}$ domain-variant. That is, they want to make the distributions $S(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) | \mathbf{x} \sim S(\mathbf{x})\}$ and $T(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) | \mathbf{x} \sim T(\mathbf{x})\}$ to be similar. To achieve this, they seek the parameters θ_f of the feature mapping that *maximize* the loss of the domain classifier (by making the two feature distributions as similar as possible), while simultaneously seeking the parameters θ_d of the domain classifier that *minimize* the loss of the domain classifier. In addition, they seek to minimize the loss of the label predictor. More formally, they consider the functional:

$$\begin{aligned} E(\theta_f,\theta_y,\theta_d) = & \sum_{\substack{i=1,\ldots,N \\ d_i=0}} L_y(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y),y_i) \\ & -\lambda \sum_{i=1,\ldots,N} L_d(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d),y_i) \\ = & \sum_{\substack{i=1,\ldots,N \\ d_i=0}} L_y^i(\theta_f,\theta_y) - \lambda \sum_{i=1,\ldots,N} L_d^i(\theta_f,\theta_d) \end{aligned} \tag{39}$$

where $L_y(\cdot,\cdot)$ is the loss for label prediction (*e.g.* multinomial), $L_d(\cdot,\cdot)$ is the loss for the domain classification (*e.g.*, logistic), while L_y^i and L_d^i denote the corresponding loss functions evaluated at the i-th training example. Based on this, they seek the parameters $\hat{\theta}_f$, $\hat{\theta}_y$, $\hat{\theta}_d$ that deliver a saddle point of the functional as defined by Eq.(39):

$$\begin{aligned} (\hat{\theta}_f,\hat{\theta}_y) &= \arg\min_{\theta_f,\theta_y} E(\theta_f,\theta_y,\hat{\theta}_d) \\ \hat{\theta}_d &= \arg\min_{\theta_d} E(\hat{\theta}_f,\hat{\theta}_y,\theta_d) \end{aligned} \tag{40}$$

The authors demonstrate that standard stochastic gradient solvers (SGD) can be adapted for the search of this saddle point. They show that a saddle point for Eq.(39) and Eq. (40) can be found as a stationary point of the following stochastic updates:

$$\theta_f \leftarrow \theta_f - \mu\left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f}\right) \tag{41}$$

$$\theta_y \leftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y} \tag{42}$$

$$\theta_d \leftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d} \tag{43}$$

where μ is the learning rate. As direct implementation of Eq.(41) – Eq.(43) is not possible, they authors propose to reduce the updates to some form of SGD by introducing a special *gradient reversal layer* (GRL). During the forward propagation, GRL acts as an identity transform. During the back-propagation though, GRL takes the gradient from the subsequent level, multiplies it by $-\lambda$ and passes it to the preceding layer. This GRL is inserted between the feature extractor and the domain classifier. Running SGD in this model implements the updates Eq.(41) – Eq.(43) and converges to a saddle point of Eq.(39).

To evaluate the proposed approach, the authors perform experiments on a number of image datasets and their modifications. Results show that their approach outperforms baseline methods and some previous domain adaptation methods.

3.3.4 Subspace alignment

The way to learn a shared feature space for DA demands that the source and target data distributions be enough similar to each other. If not, there might not exist such a common feature transformation that projects two distinct data distributions into two nearby feature distributions. To relax this assumption, one could assume that the source and target data distributions lie in two different subspaces and then find a way to align these two subspaces to achieve adaptation. In the following we show some different ways to make this subspace alignment possible.

3.3.4.1 *Subspace Alignment with dimensionality reduction methods*: Similar to the methods introduced in Sect.3.3.1, Fernando *et al.* also make use of dimensionality reduction methods for subspace learning [35]. The difference is that, they propose to use two PCAs as dimension reduction on both source and target domain, respectively. Following theoretical recommendations of Shai-Ben David's research, this method designs two different subspaces to represent the two different domains, rather than to drag different domains into a common shared subspace. This goal is achieved via optimizing a mapping function that transforms the source subspace into the target one. The authors design a new domain adaptation approach based on subspace alignment.

Firstly, they transform every source and target sample in the form of a D-dimensional z-normalized vector (*i.e.* a vector of zeros mean and unit standard deviation). Then, by using PCA, they select for each domain d eigenvectors corresponding to the d largest eigenvalues. These eigenvectors are selected as bases of the source and target subspaces, which are denoted by X_S and X_T, respectively ($X_S, X_T \in \mathbb{R}^{D\times d}$). Note that X_S' and X_T' are orthonormal, which means $X_S' X_S = \mathbf{I}_d$, $X_T' X_T = \mathbf{I}_d$ ($\mathbf{I}_d$ is the identity matrix of size d).

As projecting two different domains into an intermediate common shared subspace may lead to information loss in both source and target domains, the authors suggest to project source and target data to their corresponding subspaces X_S and X_T, respectively. Suppose a source sample $\mathbf{y}_S \in \mathbb{R}^{1\times D}$ and a target sample $\mathbf{y}_T \in \mathbb{R}^{1\times D}$, the projection is done by $\mathbf{y}_S X_S$ and $\mathbf{y}_T X_T$. They then provide a transformation matrix M from X_S to X_T, which is supposed to connect the two domains and reduce the divergence between the two domains. The transformation matrix M is learned via minimizing the following Bregman matrix divergence:

$$F(M) = \|X_S M - X_T\|_F^2 \tag{44}$$

$$M^* = \arg\min(F(M)) \tag{45}$$

where $\|\cdot\|_F^2$ is the Frobenius norm. Since X_S and X_T are generated from eigenvectors, they are intrinsically regularized. The Frobenius norm is invariant to orthonormal operations, therefore Eq.(44) can be rewritten as follows:

$$F(M) = \|X'_S X_S M - X'_S X_T\|_F^2 = \|M - X'_S X_T\|_F^2 \tag{46}$$

The optimal M^* could therefore be obtained as $M^* = X'_S X_T$, which implies that the new coordinate system is equivalent to $X_a = X'_S X_S X_T$. This X_a is called the *target aligned source coordinate system*. When source and target domains are the same, then $X_S = X_T$ and M^* is the identity matrix.

A novel similarity function $Sim(\mathbf{y}_S, \mathbf{y}_T)$ is defined as follows to compare a source sample $\mathbf{y}_S$ with a target sample $\mathbf{y}_T$:

$$Sim(\mathbf{y}_S, \mathbf{y}_T) = (\mathbf{y}_S X_S M^*)(\mathbf{y}_T X_T)' = \mathbf{y}_S A \mathbf{y}'_T \tag{47}$$

where $A = X_S M^* X'_T$. This $Sim(\mathbf{y}_S, \mathbf{y}_T)$ could be used directly to perform a k-nearest neighbor classification task. An alternative solution is to firstly project the source data via X_a into the target aligned source subspace and project the target data into the target subspace using X_T, then learn a SVM from this d-dimensional space.

In this method, the unique hyper parameter is d, the number of eigenvectors. The authors have derived an upper bound on the similarity function $Sim(\mathbf{y}_S, \mathbf{y}_T)$, which corresponds to d. And they show that d could be efficiently tuned with this bound to guarantee the solution M^* being stable and not over-fitted.

An extension to this work is tensor unsupervised domain adaptation in [36].

3.3.4.2 *Joint Geometrical and Statistical Alignment*: The Subspace Alignment (SA) method which is introduced in the previous subsection does not assume that there exist a unified transformation to reduce the domain shifts. The variance of projected source domain data will be different from that of target domain after the linear mapping of the source subspace because of the domain shift. Therefore, SA fails to minimize the distribution mismatch between domains after aligning the subspaces. In addition, SA cannot deal with situations where the shift between two subspaces is nonlinear. Subspace Distribution Alignment (SDA) [37] improves SA by considering the variance of the orthogonal principal components. However, the variances are considered based on the aligned subspaces. Hence, only the magnitude of each eigen direction is changed and SDA may still fail when the domain shift is large. To solve this problem, Zhang *et al.* in [38] proposed a unified framework, subsequently referred to as Joint Geometrical and Statistical Alignment (JGSA), that reduces the shift between domains both statistically and geometrically.

JGSA reduces the domain divergence both statistically and geometrically by exploiting both shared and domain specific features of two domains. JGSA is formulated by finding two coupled projections, namely A for source and B for target, to obtain new representations of respective domains, such that (1) the variance of target domain is maximized, (2) the discriminative information of source domain is preserved, (3) the divergence of source and target distributions is minimized, and (4) the divergence between source and target subspaces is minimized.

The overall objective function of JGSA is defined as follows:

$$\max \frac{\mu\{\text{Target Var.}\} + \beta\{\text{Between Class Var.}\}}{\{\text{Distribution shift}\} + \lambda\{\text{Subspace shift}\} + \beta\{\text{Within Class Var.}\}} \tag{48}$$

where λ, μ and β are trade-off parameters to balance the importance of each quantity, and Var. indicates variance.

Let the source domain data be $X_s \in \mathbb{R}^{D \times n_s}$ and the target domain data be $X_t \in \mathbb{R}^{D \times n_t}$, where D is the dimension of the data instance, n_s and n_t are number of samples in the source and target domain, respectively. The target variance maximization is formulated as follows:

$$\max_B Tr(B^\top S_t B) \tag{49}$$

where $S_t = X_t H_t X_t^\top$ is the target domain scatter matrix, $H_t = I_t - \frac{1}{n_t} 1_t 1_t^\top$ is the centering matrix, $1_t \in \mathbb{R}^{n_t}$ is the column vector with all ones.

The source discriminative information is preserved by:

$$\max_A Tr(A^\top S_b A) \tag{50}$$

$$\min_A Tr(A^\top S_w A) \tag{51}$$

where S_w is the within class scatter matrix, and S_b is the between class scatter matrix of the source domain data.

They employ the MMD criteria to compare the distributions between domains, which computes the distance between the sample means of the source and target data in the k-dimensional embeddings. Then they follow the idea of JDA as previously introduced in Sect. 3.3.1) to minimize the conditional distribution shift between domains. By combining the marginal and conditional distribution shift minimization terms, the final distribution divergence minimization term is defined as:

$$\min_{A,B} Tr\left(\begin{bmatrix} A^\top & B^\top \end{bmatrix} \begin{bmatrix} M_s & M_{st} \\ M_{ts} & M_t \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}\right) \tag{52}$$

where M_s, M_t, M_{st} and M_{ts} construct the relationships between source domain data and target domain data.

Similar to SA, they also reduce the discrepancy between domains by moving closer the source and target subspaces. The subspace divergence minimization is achieved by:

$$\min_{A,B} \|A - B\|_F^2 \tag{53}$$

By using Eq.(53) along with Eq.(52), both shared and domain specific features are exploited such that the two domains are well aligned geometrically and statistically.

Finally, by incorporating the above five quantities all together, *i.e.*, Eq.(49), Eq.(50), Eq.(51), Eq.(52) and Eq.(53), the objective function Eq.(48) can be rewritten as follows:

$$\max_{A,B} \frac{Tr\left(\begin{bmatrix} A^\top & B^\top \end{bmatrix} \begin{bmatrix} \beta S_b & \mathbf{0} \\ \mathbf{0} & \mu S_t \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}\right)}{Tr\left(\begin{bmatrix} A^\top & B^\top \end{bmatrix} \begin{bmatrix} M_s + \lambda I + \beta S_w & M_{st} - \lambda I \\ M_{ts} - \lambda I & M_t + (\lambda + \mu) I \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}\right)} \tag{54}$$

where $I \in \mathbb{R}^{d \times d}$ is the identity matrix. Minimizing the denominator of Eq.(54) encourages small marginal and conditional distribution shifts, and small within class variance in the source domain. Maximizing the numerator of (54) encourages large target domain variance, and large between class variance in the source domain. Similar to JDA, they also iteratively update the pseudo labels of target domain data using the learned transformations to improve the labeling quality until convergence.

3.3.4.3 *Subspace Alignment with Optimal Transportation*: A different way to achieve subspace alignment is to learn the mapping between source and target subspaces as an optimal transportation [39] [40] [41] [42].

Take the most recent work [42] as an example, the assumption is that there exists a non-linear transformation between the joint feature/label space distributions of the two domains $\mathcal{P}_s$ and $\mathcal{P}_t$ that can be estimated with optimal transport.

The authors propose to find a function f that predicts an output value given an input $\mathbf{x} \in \mathcal{X}$, and that minimizes the optimal transport loss between the joint source distribution $\mathcal{P}_s$ and an estimated target joint distribution $\mathcal{P}_t^f = (X, f(X))$ depending on f. This method is denoted as JDOT for "Joint Distribution Optimal Transport". They show that the resulting optimization problem stands for a minimization of a bound on the target error of f and propose an efficient algorithm to solve it.

Specifically, they look for a transformation $\mathcal{T}$ that will align directly the joint distributions $\mathcal{P}_s$ and $\mathcal{P}_t$. $\mathcal{T}$ is implicitly expressed through a coupling between both joint distributions as:

$$\gamma_0 = \underset{\gamma \in \Pi(\mathcal{P}_s, \mathcal{P}_t)}{\arg\min} \int_{(\Omega \times \mathcal{C})^2} \mathcal{D}(\mathbf{x}_1, y_1; \mathbf{x}_2, y_2) d\gamma(\mathbf{x}_1, y_1; \mathbf{x}_2, y_2) \tag{55}$$

where $\mathcal{D}(\mathbf{x}_1, y_1; \mathbf{x}_2, y_2) = \alpha \mathbf{d}(\mathbf{x}_1, \mathbf{x}_2) + \mathcal{L}(y_1, y_2)$ is a joint cost measure combining both the distances between the samples and a loss function $\mathcal{L}$ measuring the discrepancy between y_1 and y_2. (Matching close source and target samples with similar labels costs few.) α is a positive parameter which balances the metric in the feature space and the loss. When $\alpha \to +\inf$, this cost is dominated by the metric in the input feature space, and the solution of the coupling problem is the same as in [40].

It can be shown that a minimizer to (55) always exists and is unique provided that $\mathcal{D}(\cdot)$ is lower semi-continuous (see [43], Theorem 4.1), which is the case when $d(\cdot)$ is a norm and for every usual loss functions [44].

In unsupervised DA, y_2 is unknown. The authors replace y_2 by a proxy $f(\mathbf{x}_2)$. The joint distribution which uses a given function f as a proxy for y is then defined as:

$$\mathcal{P}_t^f = (\mathbf{x}, f(\mathbf{x}))_{\mathbf{x} \sim \mu_t} \tag{56}$$

In practice they consider empirical versions of $\mathcal{P}_s$ and $\mathcal{P}_t^f$, *i.e.* $\hat{\mathcal{P}}_s = \frac{1}{N_s} \sum_{i=1}^{N_s} \delta_{\mathbf{x}_i^s, y_i^s}$ and $\hat{\mathcal{P}_t^f} = \frac{1}{N_t} \sum_{i=1}^{N_t} \delta_{\mathbf{x}_i^t, f(\mathbf{x}_i^t)}$. γ is then a matrix which belongs to Δ, *i.e.* the transportation polytope of non-negative matrices between uniform distributions. The goal is to estimate a prediction f on the target domain, they propose to find the one that produces predictions that match optimally source labels to the aligned target instances in the transport plan. Therefore they propose to solve the following problem for JDOT:

$$\min_{f, \gamma \in \Delta} \sum_{i,j} \mathcal{D}(\mathbf{x}_i^s, y_i^s; \mathbf{x}_j^t, f(\mathbf{x}_j^t)) \gamma_{i,j} \equiv \min_f W_1(\hat{\mathcal{P}}_s, \hat{\mathcal{P}_t^f}) \tag{57}$$

where W_1 is the 1-Wasserstein distance for the loss $\mathcal{D}(\mathbf{x}_1, y_1; \mathbf{x}_2, y_2) = \alpha \mathbf{d}(\mathbf{x}_1, \mathbf{x}_2) + \mathcal{L}(y1, y2)$.

The authors proved that the function f they retrieve is theoretically sound with respect to the target error. (Chap 4)

Regularization on f: In practice they add a regularization term for function f in order to avoid over-fitting. Assume that the function space $\mathcal{H}$ to which f belongs is either a RKHS or a function space parameterized by some parameters $\mathbf{w} \in \mathbb{R}^p$. (The framework encompasses linear models, neural networks, and kernel methods.) Depending on how $\mathcal{H}$ is defined, $\Omega(f)$ is either a non-decreasing function of the squared-norm induced by the RKHS or a squared-norm on the vector parameter. They further assume that $\Omega(f)$ is continuously differentiable. The objective with regularization is then defined as:

$$\min_{f \in \mathcal{H}, \gamma \in \Delta} \sum_{i,j} \gamma_{i,j} (\alpha \mathbf{d}(\mathbf{x}_i^s, \mathbf{x}_j^t) + \mathcal{L}(y_i^s, f(\mathbf{x}_j^t))) + \lambda \Omega(f) \tag{58}$$

where the loss function $\mathcal{L}$ is continuous and differentiable with respect to its second variable.

Optimization: According to the hypotheses on f and $\mathcal{L}$, Problem (58) is smooth and the constraints are separable according to f and γ. Hence, a natural way to solve this problem is to rely on alternate optimization *w.r.t.* both parameters γ and f. This algorithm is known as Block Coordinate Descent (BCD) or Gauss-Seidel method.

Solving with fixed f boils down to a classical OT problem with a loss matrix $\mathbf{C}$ such that $C_{i,j} = \alpha \mathbf{d}(\mathbf{x}_i^s, \mathbf{x}_j^t) + \mathcal{L}(y_i^s, f(\mathbf{x}_j^t))$.

The optimization problem with fixed γ leads to a new learning problem expressed as:

$$\min_{f \in \mathcal{H}} \sum_{i,j} \gamma_{i,j} \mathcal{L}(y_i^s, f(\mathbf{x}_j^t)) + \lambda \Omega(f) \tag{59}$$

Choice of α: A natural choice of the α parameter is obtained by normalizing the range of values of $\mathbf{d}(\mathbf{x}_i^s, \mathbf{x}_j^t)$ with $\alpha = \frac{1}{\max_{i,j} \mathbf{d}(\mathbf{x}_i^s, \mathbf{x}_j^t)}$. They show that this setting is very good in two out of three experiments. While in some cases, better performances are obtained with a cross-validation of this parameter. (α is strongly linked to the smoothness of the loss $\mathcal{L}$ and of the optimal labeling functions and can be seen as a Lipschitz constant in the bound of Theorem D.1)

Bound on the target error: They give a bound on the target error $err_T(f)$ in Theorem 3.1 (and a complete version with proof in Theorem D.1). There are five terms in this bound, the first two terms correspond to the objective function (57) they propose to minimize accompanied with a sampling bound. The last term assesses the probability under which the probabilistic Lipchitzness (which is defined in this work) does not hold. The remaining two terms involving f^* correspond to the joint error minimizer illustrating that domain adaptation can work only if one can predict well in both domains. If the last terms are small enough, adaptation is possible if one is able to align well $\mathcal{P}_s$ and $\mathcal{P}_t^f$, provided that f^* and Π^* verify the PTL (Probabilistic Transfer Lipschitzness).

Compared to previous works [39] [40] [41], the proposed method do not only differ by the nature of the considered distributions (joint distributions), but also in the way the optimal plan is used to find f. The previous works learn a complex mapping between the source and target distributions when the objective is only to estimate a prediction function f on target. The previous works rely on a barycentric mapping that minimizes only approximately the Wasserstein distance between the distributions, while JDOT uses the optimal plan to propagate and fuse the labels from the source to target.

Another related work is [45], where Thorpe *et al.* introduced the Transportation L^p distance. Their objective is to compute a meaningful distance between multi-dimensional signals. Their distance can be seen as optimal transport between two distributions of the form (56) where the functions are known and the label loss $\mathcal{L}$ is chosen as a L^p distance. In JDOT, the authors introduce a more general class of loss $\mathcal{L}$, and their goal is to estimate the target function f which is not known *a priori*.

3.3.5 Relaxation on shared label space assumption

One of the assumptions of DA, that source and target task share a same label space, restricts its application in reality. When outlier classes appear in source or target data, it will make the equal class number adaptation difficult. Therefore, recently several works start to study the case without this shared label space assumption. The main idea of these works is to identify the outliers and only do adaptation on classes shared by source and target.

For example, in [46] the authors proposed the 'open set Domain Adaptation' problem, where only a few categories of interest are shared between source and target (outliers exist in both source and target). To deal with this problem, they learn a mapping from source to target, which maps source samples to be close to target distribution. This is done iteratively: first assign class labels to some target samples, then minimizing the distance between the samples of the source and the target domain that are labeled by the same category. The assignment problem is defined by a binary linear program that also includes an implicit outlier handling, which discards images that are not related to any image in the source domain.

In [47] the authors deal with a similar problem *partial domain adaptation*, where the target domain has less number of classes compared to the source domain. They extend the adversarial nets-based domain adaptation and proposes a novel adversarial nets-based partial domain adaptation method to identify the source samples that are potentially from the outlier classes and, at the same time, reduce the shift of shared classes between domains. In [48] the authors also deal with the partial domain adaptation problem. They propose Selective Adversarial Network (SAN), which selects out the outlier source classes and maximally matches the data distributions in the shared label space.

4 Knowledge transfer from pre-learned model

In this section we are going to introduce transfer learning works that reuse knowledge from previously learned source tasks. Compared to those introduced in section 3, these methods reuse knowledge from parameters (or outputs) of a model pre-learned for source task. Since they avoid learning from scratch with source raw data, these methods are usually more efficient in training. While as the pre-learned model is more adapted to source task, it makes the result of knowledge transfer more sensitive to distance between target and source.

4.1 From feature extractor parameters (Transferable/shared feature extractor)

The very first group of methods are those who reuse a pre-learned feature extractor for new target tasks.

4.1.1 Transferable feature learning with shallow Neural Networks

In one of the earliest works on transfer learning [49], Thrun introduced the concept of *lifelong learning*. He proposed a transfer algorithm that uses source training data to learn a function, denoted by $g : I \to I'$, which maps input samples in feature space I to a new feature space I'. The main idea is to find a new representation where every pair of positive examples for a task will lie close to each other while every pair of positive and negative examples will lie far from each other.

Let $\mathcal{P}_k$ be the set of positive samples for the k-th task and $\mathcal{N}_k$ the set of negative samples, Thrun's transfer algorithm minimizes the following objective:

$$\min_{g\in\mathcal{G}} \sum_{k=1}^{m} \sum_{\mathbf{x}_i\in\mathcal{P}_k} \Big(\sum_{\mathbf{x}_j\in\mathcal{P}_k} \parallel g(\mathbf{x}_i)-g(\mathbf{x}_j) \parallel - \sum_{\mathbf{x}_j\in\mathcal{N}_k} \parallel g(\mathbf{x}_i)-g(\mathbf{x}_j) \parallel \Big) \tag{60}$$

where $\mathcal{G}$ is the set of transformations encoded by a two layer neural network. The transformation $g(\cdot)$ learned from the source data is then used to project the samples of the target task into the new feature space. Classification for the target task is performed by running a nearest neighbor classifier in the new space.

The paper presented experiments on a small object recognition task. The results showed that when labeled data for the target task is scarce, the representation obtained by running their transfer algorithms on source training data could improve the classification performance of a target task.

This work is further generalized by several authors [50] [51] [52]. The three works can all be casted under the framework of '*structural learning*' proposed in [50].

4.1.2 Structural learning methods (Shared feature extractor)

In a *structural learning* framework we assume the existence of task-specific parameters $\mathbf{w}_k$ for each task and shared parameters θ that parameterize a family of underlying transformations. Both the structural parameters and the task-specific parameters are learned together via joint risk minimization on some supervised training data for m related tasks.

Consider learning linear predictors of the form $h_k(\mathbf{x}) = \mathbf{w}_k^T v(\mathbf{x})$ for some $\mathbf{w} \in \mathbb{R}^z$ and some transformation $v \in \mathcal{V} : \mathbb{R}^d \to \mathbb{R}^z$. In particular, let $\mathcal{V}$ be the family of linear transformations: $v^\theta(\mathbf{x}) = \theta\mathbf{x}$ where θ is a z by d matrix that maps a d dimensional input vector to a z dimensional space.

Define the task-specific parameters matrix: $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ where $\mathbf{w}_k \in \mathbb{R}^z$ are the parameters for the k-th task and $w_{j,k}$ is the parameter value for the j-th hidden feature and the k-th task. A *structural learning* algorithm finds the optimal task-specific parameters W^* and structural parameters θ^* by minimizing a jointly regularized empirical risk:

$$\arg\min_{W,\theta} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \text{Loss}(\mathbf{w}_k^\top \theta \mathbf{x}_i^k, y_i^k) + \gamma \Phi(W) + \lambda \Psi(\theta) \quad (61)$$

The first term in equation (61) measures the mean error of the m classifiers by means of some loss function $\text{Loss}(\cdot)$. The second term is a regularization penalty on the task-specific parameters W and the last term is a regularization penalty on the structural parameters θ. Different choices of regularization functions $\Phi(W)$ and $\Psi(\theta)$ result in different *structural learning* algorithms.

[50] combine a l_2 regularization penalty on the task-specific parameters with an orthonormal constraint on the structural parameters, resulting in the following objective:

$$\arg\min_{W,\theta} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \text{Loss}(\mathbf{w}_k^\top \theta \mathbf{x}_i^k, y_i^k) + \gamma \sum_{k=1}^{m} \parallel \mathbf{w}_k \parallel_2^2, \ \ s.t.\ \theta\theta^T = I \quad (62)$$

where θ is a z by d matrix, z is assumed to be smaller than d and its optimal value is found using a validation set. Therefore knowledge transfer is realized by mapping the high dimensional feature vector $\mathbf{x}$ to a new feature vector $\theta \cdot \mathbf{x}$ in a shared low dimensional feature space. The authors propose to solve (62) using an alternating minimization procedure. This algorithm is applied in the context of *asymmetric transfer* where *auxiliary* (*i.e.* source) training sets are utilized to learn the structural parameter θ. The learned structural parameter is then used to project the samples of the target task and train a classifier on the new feature space. The paper presented experiments on text categorization where the source training sets were automatically derived from unlabeled data (this algorithm can therefore be regarded as a semi-supervised training algorithm). their results showed that the proposed algorithm gave significant improvements over a baseline method that trained on the labeled data ignoring the source training sets.

This work is further applied to image classification in [53] in which they consider having a large set of images with associated captions, while among them only a few images are annotated with story news labels. They take the prediction of content words from the captions to be the source tasks and the prediction of a story label to be the target tasks. The goal is to leverage the source tasks to derive a lower dimensional representation that captures the relevant information necessary to discriminate between different stories. They perform experiments with this method both on synthetic data and real news image data. Results show that when source data labels are suitably related to a target task, the structural learning method can discover feature groupings that speed up learning of the target task.

[51] proposed an alternative model to learn shared representations. In their approach the structural parameter θ is assumed to be a d by d matrix, *i.e.* the linear transformation does not map the inputs $\mathbf{x}$ to a lower dimensional space. Instead, sharing of hidden features across tasks is realized by a regularization penalty imposed on the task-specific parameters W, which requires only a few hidden features to be used by any task (*i.e.* requires the matrix W to be row-sparse). This regularization is achieved by using the following matrix norm: $l_{1,2}(W) = \sum_{j=1}^{z} \parallel \mathbf{w}^j \parallel_2$, which is known to promote row sparsity in W. Therefore the objective can be written as:

$$\arg\min_{W,\theta} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \text{Loss}(\mathbf{w}_k^\top \theta \mathbf{x}_i^k, y_i^k) + \gamma l_{1,2}(W) \quad (63)$$

The authors showed that this problem is equivalent to a convex problem for which they developed an alternating minimization algorithm. The paper presented experiments on a product rating problem where the goal is to predict ratings given by different subjects. In the context of multi-task learning predicting the ratings for a single subject can be regarded as a task. The transfer learning assumption is that predictions made by different subjects are related. The results showed that their algorithm gave better performance than a baseline model where each task was trained independently with an l_1 penalty.

[52] proposed a regularization scheme for transfer learning based on a trace norm regularization penalty. Consider the following m by d parameter matrix $W = [\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^m]$, where each row corresponds to the parameters of one task. The transfer algorithm minimizes the following jointly regularized objective:

$$\arg\min_{W} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \text{Loss}(\mathbf{w}_k^\top \mathbf{x}_i^k, y_i^k) + \gamma \Omega(W) \quad (64)$$

where $\Omega(W) = \sum_i |\gamma_i|$ and γ_i is the i-th eigenvalue of W. This norm is used because it is known to induce low rank on solution matrices W [54]. Recall that the rank of a d by m matrix W is the minimum z such that W can be factored as $W = \theta^\top W'$, for a z by m matrix W' and a z by d matrix θ.

Notice that θ is no longer in equation (64), this is because in this formulation we do not search explicitly for a transformation θ. Instead, we utilize the regularization penalty $\Omega(W)$ to encourage solutions where the task-specific parameters W can be expressed as the combination of a few basis shared across tasks. This optimization problem can be expressed as a semi-definite program and can be solved with an interior-point method. However, the authors argue that interior point methods scale poorly with the size of the training set and proposed a gradient based method to solve this problem by minimizing a smoothed approximation of (64). The authors conducted experiments on a multi-class classification task where the goal is to distinguish between 72 classes of mammals. The performance of their transfer learning algorithm is compared to that of a baseline multi-class SVM classifier. Their results show that the trace-norm penalty can improve multi-class accuracy when only a few samples are available for training.

4.1.3 Transferable feature learning with metric learning

As shown in section 3.3.2, metric learning could provide same effect as subspace learning. Therefore transferable/shared feature learning could also be done with metric learning methods.

One of the first works is [55], which tries to learn a shared feature representation using metric learning disciplines. Similar to the very first transfer learning method [49], this algorithm learns a feature transformation which is later utilized by a nearest neighbor classifier for the target task. Unlike Thrun's transfer algorithm which deploys a neural network to learn the feature transformation, Fink's transfer algorithm follows a max-margin approach to directly learn a distance metric.

Consider learning a function $d : X \times X \rightarrow \mathbb{R}$ which has the following properties: (i) $d(x,x') \geqslant 0$, (ii) $d(x,x') = d(x',x)$ and (iii) $d(x,x') + d(x',x'') \geqslant d(x,x'')$. A function satisfying these three properties is called a pseudo-metric. Ideally, we would like to learn a function d that assigns a smaller distance to pairs having the same label than to pairs with different labels. More precisely, for any positive samples $\mathbf{x}_i$, $\mathbf{x}_j$, and negative sample $\mathbf{x}_k$, we would require the difference in distance to be at least γ, :

$$d(\mathbf{x}_i,\mathbf{x}_j) \leqslant d(\mathbf{x}_i,\mathbf{x}_k) - \gamma \tag{65}$$

Fink's algorithm make use of a pseudo-metric of the form: $d(\mathbf{x}_i,\mathbf{x}_j)^2 = \parallel \theta\mathbf{x}_i - \theta\mathbf{x}_j \parallel_2^2$, the problem is therefore to learn a linear projection θ that achieves γ separation as defined in (65). This projection is learned with source data, and then is deployed for classification of target data. The underlying transfer learning assumption is that a projection θ that achieves γ separation on the source tasks will most likely achieve γ separation on the target task. Therefore, if we project the target samples using θ and run a nearest neighbor classifier in the new space we are likely to get a good performance.

Like the early works introduced in subsection 4.1.1, the strong assumption which requests the source data to be very close to the target restricts the effectiveness of this method for more general situations. In [56] a new method is introduced which combines the large margin nearest neighbor classification with the multi-task learning paradigm. Unlike the previously introduced method, this method learns a specific metric $d_t(\cdot,\cdot)$ for each of the T tasks. They then model the commonalities between various tasks through a shared Mahalanobis metric with $\mathbf{M}_0 \succeq 0$ and the task-specific idiosyncrasies with additional matrices $\mathbf{M}_1,\ldots,\mathbf{M}_T \succeq 0$. The distance for task t is defined as follows:

$$d_t(\mathbf{x}_i,\mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{M}_0 + \mathbf{M}_t)(\mathbf{x}_t - \mathbf{x}_j)} \tag{66}$$

Although there is not a specific projection as θ defined in [55], this distance defined in Eq. (66) could still be considered as a distance in an underlying new feature space. The metric defined by $\mathbf{M}_0$ picks up general trends across multiple data sets and $\mathbf{M}_{t>0}$ specialize the metric further for each particular task. They authors have proved theoretically that the Eq. 66 is a well defined pseudo-metric when the matrices $\mathbf{M}_t$ are constrained to be positive semi-definite (*i.e.* $\mathbf{M}_t \succeq 0$).

To ensure that the learning algorithm does not put too much emphasis onto the shared parameters $\mathbf{M}_0$ or the individual parameters $\mathbf{M}_1,\ldots,\mathbf{M}_T$, they use the regularization term as follows:

$$\min_{\mathbf{M}_0,\ldots,\mathbf{M}_T} \gamma_0\|\mathbf{M}_0 - \mathbf{I}\|_F^2 + \sum_{t=1}^{T}\gamma_t\|\mathbf{M_t}\|_F^2 \tag{67}$$

The trade-off parameter γ_t controls the regularization of $\mathbf{M}_t$ for all $t = 0,1,\ldots,T$. If $\gamma_0 \rightarrow \infty$, the shared metric $\mathbf{M}_0$ reduces to the plain Euclidean metric and if $\gamma_{t>0} \rightarrow \infty$, the task-specific metrics $\mathbf{M}_{t>0}$ become irrelevant zero matrices. Therefore, if $\gamma_{t>0} \rightarrow \infty$ and γ_0 is small, a single metric $\mathbf{M}_0$ across all tasks will be learned, the result is equivalent to applying lmnn (large margin nearest neighbors) on the union of all data sets. In the other extreme case, when $\gamma_0 = 0$ and $\gamma_{t>0} \rightarrow \infty$, the formulation will reduce to T independent lmnn algorithms.

Let S_t be the set of triples restricted to only vectors for task t, *i.e.*, $S_t = \{(i,j,k) \in \mathcal{J}_t^3 : j \rightsquigarrow i, y_k \neq y_i\}$. Where $j \rightsquigarrow i$ indicates that $\mathbf{x}_j$ is a target neighbor of $\mathbf{x}_i$, and $\mathcal{J}_t$ is the set of indexes such that $i \in \mathcal{J}_t$ if and only if the input-label pair $(\mathbf{x}_i, y_i)$ belongs to task t. Combine the regularizer in Eq. 67 with the objective of lmnn applied to each of the T tasks. To ensure well-defined metrics, the authors also add constraints that each matrix is positive semi-definite, *i.e.* $\mathbf{M}_t \succeq 0$. The resulting algorithm is called *multi-task large margin nearest neighbor*(mt-lmnn). The optimization problem is shown in Eq (68) and can be solved after some modifications to the special-purpose solver presented in [57].

$$\min_{\mathbf{M}_0,\ldots,\mathbf{M}_T} \gamma_0\|\mathbf{M}_0 - \mathbf{I}\|_F^2 + \sum_{t=1}^{T}\Bigg[\gamma_t\|\mathbf{M}_t\|_F^2 + \sum_{(i,j)\in\mathcal{J}_{t,j\rightsquigarrow i}} d_t^2(\mathbf{x}_i,\mathbf{x}_j) + \sum$$

subject to: $\forall t, \forall(i,j,k) \in S_t$:

(1) $d_t^2(\mathbf{x}_i,\mathbf{x}_k) - d_t^2(\mathbf{x}_i,\mathbf{x}_j) \geqslant 1 - \xi_{ijk}$

(2) $\xi_{ijk} \geqslant 0$

(3) $\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_T \succeq 0.$

(68)

This regularization term $\gamma_0\|\mathbf{M}_0 - \mathbf{I}\|_F^2$ could be interpreted as learning $\mathbf{M}_0$ while trying to stay close to the Euclidean distance. Another kind of metric regularization for transfer learning is to replace $\mathbf{I}$ with the auxiliary metric $\mathbf{M}_S$ learned from source task. The regularization $\|\mathbf{M} - \mathbf{M}_S\|$ could be interpreted as transferring knowledge brought by $\mathbf{M}_S$ for learning $\mathbf{M}$.

4.1.4 Transferable feature learning with Deep Neural Networks

Recently, the rapid growth of deep learning techniques shows new ways for unsupervised feature learning. Unsupervised deep learning models, such as auto-encoders (AEs), deep belief networks (DBNs) and generative adversarial networks (GANs), could be used for feature learning with unlabeled data. The resulting feature extraction networks could then be applied for new target tasks.

4.2 From classifier parameters

Apart from feature extractor parameters, classifier parameters of a pre-learned model could also be reused for learning a new target task. Here we show two groups of methods, one is based on discriminative classification model (SVM), and another is based on generative classification models.

4.2.1 Knowledge transfer from SVM model parameters

Support Vector Machine (SVM) is a supervised learning method for solving classification and regression problems, and several early works on classifier level knowledge transfer are constructed based on the original SVM classifier. A common form of the objective function of these SVM based transfer learning models could be expressed as follows:

$$\min_{\mathbf{w}^t, b} \Phi(\mathbf{w}^t) + C \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_T} \varepsilon(\mathbf{x}_i, y_i; \mathbf{w}^t, b) \tag{69}$$

where $\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_T} \varepsilon(\mathbf{x}_i, y_i; \mathbf{w}^t, b)$ is the loss on labeled samples in the target domain $\mathcal{D}_T$, and $\Phi(\mathbf{w}^t)$ is the regularization on model parameter $\mathbf{w}^t$ which enforces the margin maximization and the knowledge transfer. The knowledge transfer regularization is usually expressed as a minimization of the distance between the pre-learned source parameter $\mathbf{w}^s$ and the target parameter $\mathbf{w}^t$.

One of the first SVM based transfer learning works is the Adaptive-SVM (A-SVM) proposed in [58], in which Yang *et al.* assume that the decision function $f_T(\cdot)$ for the target classification task can be formulated as:

$$f_T(\mathbf{x}) = f_S(\mathbf{x}) + \Delta f(\mathbf{x}) \tag{70}$$

where $f_S(\cdot)$ is the source decision function and $\Delta f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ is the perturbation function. The perturbation function $\Delta f(\cdot)$ is learned using the labeled data from the target domain $\mathcal{D}_T$ and the pre-learned parameters for the source decision function $f_S(\cdot)$, the objective function is as follows:

$$\begin{gathered}\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{s.t. } \xi_i \geqslant 0,\ y_i f_S(\mathbf{x}_i) + y_i \mathbf{w}^\top \phi(\mathbf{x}_i) \geqslant 1 - \xi_i,\ \forall (\mathbf{x}_i, y_i) \in \mathcal{D}_T\end{gathered} \tag{71}$$

where $\sum_i \xi_i$ measures the total classification error of the adapted classifier $f_T(\cdot)$ in the target domain. The first term in (71) minimizes the deviation between the target decision boundary and the source decision boundary. The cost factor C balances the contribution between the source classifier and the training examples, *i.e.* the larger C is, the smaller the influence of the source classifier is.

This work was improved in [59] for object detection. Aytar and Zisserman fistly show a more general form of the objective function for A-SVM(this form was firstly introduced in [60]) as follows:

$$\min_{\mathbf{w}} \|\mathbf{w}^t - \Gamma \mathbf{w}^s\|^2 + C\sum_{i=1}^{N} \xi_i \tag{72}$$

where $\mathbf{w}^s$ and $\mathbf{w}^t$ are the parameters for source classifier and target classifier respectively. Γ controls the amount of transfer regularization. The authors have shown that Γ is actually a trade-off parameter between margin maximization of the target classifier and the knowledge transfer from source classifier, *i.e.* the larger Γ is, the larger the knowledge transfer is, while the smaller the margin maximization is.

To avoid this trade-off, [59] propose the projective Model Transfer SVM (PMT-SVM), in which they can increase the amount of transfer without penalizing margin maximization. The objective function for PMT-SVM is as follows:

$$\min_{\mathbf{w}^t} \|\mathbf{w}^t\|^2 + \Gamma\|P\mathbf{w}^t\|^2 + C\sum_{i=1}^{N} \xi_i, \ \text{ s.t. } (\mathbf{w}^t)^\top \mathbf{w}^s \geqslant 0 \tag{73}$$

where $P = I - \frac{\mathbf{w}^s (\mathbf{w}^s)^\top}{(\mathbf{w}^s)^\top \mathbf{w}^s}$ is the projection matrix, Γ controls the amount of transfer regularization, and C controls the weight of the loss function $\sum_i \xi_i$. $\|P\mathbf{w}^t\|^2 = \|\mathbf{w}^t\|^2 \sin^2\theta$ is the squared norm of the projection of the $\mathbf{w}^t$ onto the source hyperplane, θ is the angle between $\mathbf{w}^s$ and $\mathbf{w}^t$. $(\mathbf{w}^t)^\top \mathbf{w}^s \geqslant 0$ constrains $\mathbf{w}^t$ to the positive half-space defined by $\mathbf{w}^s$. Experimental results have shown that this PMT-SVM works better compared to A-SVM and SVM when having only a few labeled samples in target domain, especially for one-shot learning when only one labeled sample is available.

In [59] the authors also show a direct generalization of A-SVM to deformable transfer formulation, named Deformable Adaptive SVM (DA-SVM), for object detection with deformable part based models.

In [61] the A-SVM was improved for visual concept classification, where the authors propose the cross-domain SVM (CD-SVM) which makes use of k-nearest neighbors from the target domain to define a weight for each auxiliary pattern, and then the SVM classifier was trained with reweighted patterns.

Tommasi *et al.* [62] proposed a multi-model knowledge transfer algorithm based on the Least Square SVM (LS-SVM). The objective function of the multi-model knowledge transfer is defined as follows:

$$\min_{\mathbf{w}, b} \frac{1}{2}\|\mathbf{w} - \sum_{j=1}^{k} \beta_j \mathbf{w}_j\|^2 + \frac{C}{2}\sum_{i=1}^{N} \zeta_i (y_i - \mathbf{w}\cdot\phi(\mathbf{x}_i) - b)^2 \tag{74}$$

where $\mathbf{w}$ is the parameter of the target model and $\mathbf{w}_j$ are the parameters of the pre-learned source models, the coefficient vector $\boldsymbol{\beta}$ should be chosen in the unitary ball, *i.e.* $\boldsymbol{\beta} \leqslant 1$. The second term in Eq. (74) is the least square loss for training samples in the target domain. The optimal solution of Eq. (74) is as follows:

$$\mathbf{w} = \sum_{j=1}^{k} \beta_j \mathbf{w}'_j + \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \tag{75}$$

where $\mathbf{w}$ is expressed as a weighted sum of the pre-trained models scaled by the parameters β_j, plus the new model built on the incoming training data. The new model parameters α_i could be learned on the target training data. The optimal coefficients β_j could be found by minimizing the LOO (leave one out) error, which is an unbiased estimator of the classifier generalization error and can be used for model selection [63].

In [64] the authors extend this LSSVM based transfer learning to an incremental transfer learning setting, where the source is a pre-learned multi-class classifier for N classes, denoted as $\mathbf{W}' = [\mathbf{w}'_1, \ldots, \mathbf{w}'_N]$, and the target is a small training set composed from samples belonging to the N known classes and a new class. Their aim is to find a new set of hyperplanes $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_N]$ and $\mathbf{w}_{N+1}$, such that: (1) performance on the target $(N+1)$-th class improves by transferring from the source models, and (2) performance on the source N classes should not deteriorate or even improve compared to the former.

They achieve the first goal by using the regularizer $\|\mathbf{w}_{N+1} - \mathbf{W}'\boldsymbol{\beta}\|^2$, which enforces the target model $\mathbf{w}_{N+1}$ to be close to a linear combination of the source models. Negative transfer is prevented by weighting the amount of transfer of each source model using the coefficient vector $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_N]^\top$. To achieve the second objective, they enforce the new hyperplanes $\mathbf{W}$ to remain close to the source hyperplanes $\mathbf{W}'$ using the term $\|\mathbf{W} - \mathbf{W}'\|_F^2$. The final objective function with the two regularizers is as follows:

$$\min_{\mathbf{W},\mathbf{w}_{N+1},\mathbf{b}} \frac{1}{2}\|\mathbf{W}-\mathbf{W}'\|_F^2+\frac{1}{2}\|\mathbf{w}_{N+1}-\mathbf{W}'\boldsymbol{\beta}\|_F^2+\frac{C}{2}\sum_{i=1}^{M}\sum_{n=1}^{N+1}(\mathbf{W}_n^\top \mathbf{x}_i + b_n - Y_{i,n})^2 \tag{76}$$

where $\mathbf{Y}$ is the label matrix, $Y_{i,n}$ is equal to 1 if $y_i = n$ and is equal to -1 otherwise. The solution to this problem is given by:

$$\begin{aligned}
\mathbf{W}_n &= \mathbf{W}'_n + \sum_{i=1}^{M} A_{i,n}\mathbf{x}_i, \; n = 1, \ldots, N \\
\mathbf{w}_{N+1} &= \sum_{n=1}^{N} \beta_n \mathbf{W}'_n + \sum_{i=1}^{M} A_{i,(N+1)}\mathbf{x}_i \\
\mathbf{b} &= \mathbf{b}' - \begin{bmatrix}\mathbf{b}'' & \mathbf{b}''^\top \boldsymbol{\beta}\end{bmatrix}
\end{aligned} \tag{77}$$

where

$$\mathbf{A} = \mathbf{A}' - [\mathbf{A}'' \;\; \mathbf{A}''\boldsymbol{\beta}]$$

,

$$\begin{bmatrix}\mathbf{A}' \\ \mathbf{b}'^\top\end{bmatrix} = \mathbf{M}\begin{bmatrix}\mathbf{Y} \\ \mathbf{0}\end{bmatrix}$$

,

$$\begin{bmatrix}\mathbf{A}'' \\ \mathbf{b}''^\top\end{bmatrix} = \mathbf{M}\begin{bmatrix}\mathbf{X}^\top\mathbf{W}' \\ \mathbf{0}\end{bmatrix}$$

,

$$\mathbf{M} = \begin{bmatrix}\mathbf{X}^\top\mathbf{X} + \frac{1}{C}\mathbf{I} & \mathbf{1} \\ \mathbf{1}^\top & 0\end{bmatrix}^{-1}$$

To tune the parameter $\boldsymbol{\beta}$, they extend the method shown in [62]. They cast the optimization of $\boldsymbol{\beta}$ as the minimization of a convex upper bound of the LOO error and they propose a projected subgradient descent algorithm to find the optimal $\boldsymbol{\beta}$.

As can be seen from the above, most SVM based transfer learning methods enforce knowledge transfer simply by adding a regularization term to minimize the distance between the target model parameters and the source model parameters. This brute-force regularization work well for binary-classification when target positive category and source positive category are as close as possible. The extension to multi-class classification could be done in a one-vs-all manner as shown in [64], and the negative transfer is mainly prevented by tuning the parameter $\boldsymbol{\beta}$ (which could be seen as a model selection process).

4.2.2 *Knowledge transfer from generative models*

SVM is a kind of discriminative model which learn the conditional distribution of labels on knowing input features. Another kind of classification methods are generative models, which learn the joint distribution of the labels and input features. Generative models are also adopted for knowledge transfer, especially in the case of zeros-shot or one-shot learning for object recognition, where no target sample or only one target sample is given for training an object recognition model.

A representative work is proposed in [65], which is a Bayesian-based unsupervised one-shot learning object categorization framework that learns a new object category using a single example (or just a few).

Firstly, to formalize the task of object classification with a generative model, they start with a learned object class model and its corresponding model distribution $p(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of model parameters for the distribution. Given a new image and the objective is to decide if it contains an instance of the target object class or not. In this query image they have identified N interesting features with locations $\mathcal{X}$, and appearances $\mathcal{A}$. They now make a Bayesian decision R. For clarity, they express training images through the detected feature locations $\mathcal{X}_t$ and appearances $\mathcal{A}_t$.

$$\begin{aligned}
R &= \frac{p(\text{Object}|\mathcal{X}, \mathcal{A}, \mathcal{X}_t, \mathcal{A}_t)}{p(\text{No Object}|\mathcal{X}, \mathcal{A}, \mathcal{X}_t, \mathcal{A}_t)} \\
&= \frac{p(\mathcal{X}, \mathcal{A}|\mathcal{X}_t, \mathcal{A}_t, \text{Object})p(\text{Object})}{p(\mathcal{X}, \mathcal{A}|\mathcal{X}_t, \mathcal{A}_t, \text{No Object})p(\text{No object})} \\
&\approx \frac{\int p(\mathcal{X}, \mathcal{A}|\boldsymbol{\theta}, \text{Object})p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t, \text{Object})d\boldsymbol{\theta}}{\int p(\mathcal{X}, \mathcal{A}|\boldsymbol{\theta}_{bg}, \text{No object})p(\boldsymbol{\theta}_{bg}|\mathcal{X}_t, \mathcal{A}_t, \text{No object})d\boldsymbol{\theta}_{bg}}
\end{aligned} \tag{78}$$

Note that the ratio of $\frac{p(\text{Object})}{p(No object)}$ in the second line is usually set manually to 1, hence it is omitted in the third line of Equation (78). The goal of learning in this formulation is to estimate the density of the object model $p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t, \text{Object})$. In other words, in the high dimensional space that characterize the objects, the goal is to find the appropriate distribution that defines the extent of where and how the models occupy this space. This goal is achieved through the usage of prior knowledge.

The representation of the object class model is chosen based on the *constellation model* [66]. A constellation model consists of a number of parts, each encoding information on both the shape and appearance. The appearance of each part is modeled and the shape of the object is represented by the mutual position of the parts [67]. The entire model is generative and probabilistic, so appearance an shape are all modeled by probability density functions, which are Gaussians. Assume a generative object model is learned, with P parts and a posterior distribution on the parameters $\boldsymbol{\theta}$: $p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t)$ where $\mathcal{X}_t$ and $\mathcal{A}_t$ are the location and appearances of interesting features found in the training data. Recall the Bayesian decision rule in Equation (78).

Assume that all non-object images can also be modeled by a background with a single set of parameters $\boldsymbol{\theta}_{bg}$ which are fixed. The ratio of the priors may be estimated from the training set or set manually (usually to 1). The decision then requires the calculation of the ratio of the two likelihood functions, which may be factored as follows:

$$p(\mathcal{X}, \mathcal{A}|\boldsymbol{\theta}) = \sum_{\mathbf{h}\in H} p(\mathcal{X}, \mathcal{A}, \mathbf{h}|\boldsymbol{\theta}) = \sum_{\mathbf{h}\in H} \underbrace{p(\mathcal{A}|\mathbf{h}, \boldsymbol{\theta})}_{\text{Appearance}} \underbrace{p(\mathcal{X}|\mathbf{h}, \boldsymbol{\theta})}_{\text{Shape}} \tag{79}$$

Since the model only has P (typically 3-7) parts while there are N (up to 100) features in the image, the authors introduce an indexing variable $\mathbf{h}$ which they call a *hypothesis* which allocates each image feature either to an object or to the background.

The task in learning is to estimate the density $p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t)$. This is done using the Variational Bayes Procedure. It approximates the posterior distribution $p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t)$ by $q(\boldsymbol{\theta}, \boldsymbol{\omega}, \mathbf{h})$. ω is the mixture component label and $\mathbf{h}$ is the hypothesis. Using Bayes' rule: $q(\boldsymbol{\theta}, \boldsymbol{\omega}, \mathbf{h}) \approx p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t) \propto p(\mathcal{X}_t, \mathcal{A}_t|\boldsymbol{\theta})p(\boldsymbol{\theta})$. The likelihood terms use Gaussian densities and by assuming priors of a conjugate form, int his case a Normal-Wishart, the posterior q-function is also a Normal-Wishart density. The variational Bayes procedure is a variant of EM which iteratively updates the hyper-parameters and latent variables to monotonically reduce the Kullback-Liebler distance between $p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t)$ and $q(\boldsymbol{\theta}, \boldsymbol{\omega}, \mathbf{h})$. Using this approach one is allowed to incorporate prior information in a systematic way. There are two stages to learning: an E-step where the responsibilities of the hidden variables are calculated and an M-step to update the hyper-parameters of $q(\boldsymbol{\theta}, \boldsymbol{\omega}, \mathbf{h})$. This learning approach could be done in an incremental way when doing knowledge transfer. Assume having a model with hyper-parameters $\boldsymbol{\Theta}$, estimated using M previous images, and having N new images for updating the model. From the M previous images, one have retained sufficient statistics for each mixture component ω. Then compute the responsibilities and the sufficient statistics for the new images. In the incremental M-step combine the sufficient statistics from these new images with the existing set of sufficient statistics from the previous images, then compute the overall sufficient statistics. From these one can update the model hyper-parameters. When the model converges, the final value of the sufficient statistics from the new images are combined with the existing set.

This approach allows one to incorporate priors from unrelated object categories. For example, when learning the category motorbikes, priors can be obtained by averaging the learned model parameters from other three categories spotted cats, faces and airplanes.

Another work is [68], where the authors propose an attribute-based transfer learning framework for zero-shot and one-shot learning. They firstly build a generative attribute model to learn the probabilistic distributions of image features for each attribute, which they consider as attribute priors. These attribute priors can then be used to (1) classify unseen images of target categories (for zero-shot learning), or (2) facilitate learning classifiers for target categories when there is only one training example per target category (for one-shot learning).

Specifically, the category-attribute relationship is represented by a category-attribute matrix $\mathcal{M}$, where the entry at the m-th row and the l-th column is a binary value indicating whether category m has the l-th attribute. Each object category thus has a list of attributes whose corresponding values in $\mathcal{M}$ equal to "yes". Given an object category, the list of associated attributes $\mathbf{a}$ is deterministic. For example, for category *cow*, the attribute list is $\mathbf{a} = \{black, white, brown, spots, furry, domestic\}$. This information is supposed to be available for both source categories and target categories.

The attribute model and the target classifier belong to an extension of topic models. In a topic model, a document $\mathbf{w}$ is modeled by a mixture of topics, and each topic z is represented by a probability distribution of words (noted as w). In computer vision, a quantized image feature is often analogous to a word, a group of co-occurred image features to a topic, and an image to a document. The authors choose the bag-of-features as image representation, *i.e.* the spatial information of image features is discarded, and an image is represented as a collection of orderless visual words.

They employ the Author-Topic (AT) model as the attribute model. The AT model is a generative model which is originally designed to model the interests of authors from a given document corpus. They extend the AT model to describe the distribution of images features related to attributes. An image j has a list of attributes, denoted by $\mathbf{a}_j$. An attribute l in $\mathbf{a}_j$ is modeled by a discrete distribution of K topics, which parameterized by a K-dim vector $\boldsymbol{\theta}_l = [\theta_{l1}, \dots, \theta_{lK}]$ with topic k receiving weight θ_{lk}. The topic k is modeled by a discrete distribution of W codewords in the lexicon, which is parameterized by a W-dim vector $\boldsymbol{\phi}_k = [\phi_{k1}, \dots, \phi_{kW}]$ with code word v receiving weight ϕ_{kv}. Symmetric Dirichlet priors are placed on θ and ϕ, with $\theta_l \sim$ Dirichlet(α), and $\phi_k \sim$ Dirichlet(λ), where α and λ are hyper-parameters that affect the sparsity of these distributions. Given an attribute list $\mathbf{a}_j$ and a desired number N_j of visual words in image j. To generate each visual word, they firstly condition on $\mathbf{a}_j$, choose an attribute $x_{ji} \sim$ Uniform($\mathbf{a}_j$); then they condition on x_{ji}, choose a topic $z_{ji} \sim$ Discrete($\theta_{x_{ji}}$), where θ_l defines the distribution of topics for attribute $x = l$; finally, they condition on z_{ji} and choose a visual word $w_{ji} sim$Discrete($\phi_{z_{ji}}$), where ϕ_k defines the distribution of visual words for topic $z = k$.

Given a training corpus, the goal of inference in an AT model is to identify the values of ϕ and θ. To achieve this goal, the authors make use of a collapsed block Gibbs sampling method [69]. Here "collapse" means that the parameters ϕ and θ are analytically integrated out, and the "block" means that the pair of (x_{ji}, z_{ji}) are drawn together. To run the Gibbs sampling algorithm, they firstly initialize $\mathbf{x}$ and $\mathbf{z}$ with random assignments. In each Gibbs sampling iteration, they draw samples of x_{ji} and z_{ji} for all visual words in the training corpus in a randomly permuted order of i and j. The samples of $\mathbf{x}$ and $\mathbf{z}$ are recorded after the burn-in period. According to experimental results, 200 iterations are sufficient for the sampler to be stable. The posterior means of θ and ϕ can then be estimated using the recorded samples.

If there is only one attribute in each image and the attribute is the object category label, the AT model can be used in object categorization problems, they call this approach Category-Topic (CT) model and use it as the target classifier. (It is also possible to employ other types of target classifier, for example, the authors also evaluate SVM as a target classifier. Although experiment show that the CT model outperforms SVM by a large margin.) After learning a CT model, it can be used to classify a test image by choosing the target classifier that yields the highest likelihood.

If the attribute list is unique in each category, an AT model can also be used to classify a new image by the maximum likelihood criterion in the case of zero-shot learning problem. An approximate likelihood could be calculated with a pseudo weight for the category-specified topic distribution of a new category, this pseudo weight can be viewed as a prior before seeing the real training examples of the new category.

To deal with the one-shot learning, the main problem is that the number of training samples in source is higher than the one of target by several orders, therefore one need to control the balance between the prior information from source categories and the new information in target categories. The authors propose two approaches to achieve this goal.

The first one is knowledge transfer by synthesis of training samples. Firstly, they learn the attribute model from the training samples of the source categories; then, for each target category, they run the generative process to produce S synthesized training samples using the estimated $\hat{\theta}$ and $\hat{\phi}$ as well as the attribute list associated to this target category. The number of visual words for each generated training sample is chosen as the average number of visual words per image in the source categories. In this procedure, the number of synthesized training samples S represent the confidence about the attribute priors. It can be used to adjust the balance between the attribute priors and new observations from the training images of target categories.

The second approach is to give parameters of the CT model in the target classifiers informative priors. These priors could be estimated with the source data, the scaling factors of these priors represent the confidence on them, which can be used to control the balance between attribute priors and the new observations.

4.3 Knowledge distillation for DNN (Knowledge from both feature extractor and classifier parameters)

A special group of works is the knowledge distillation methods for Deep Neural Networks. As its name 'distillation' suggests, these works try to learn a small student network which could give equal performance as a big teacher network. These works not only benefit knowledge transfer from a big pre-learned DNN, but also try to make knowledge more compact by putting the transferred knowledge into a new DNN with smaller amount of parameters.

[70] first proposed the concept of Knowledge distillation(KD) in the teacher-student framework by introducing the teacher's softened output. They introduced the model compression method based on the concept of dark knowledge. It uses a softened version of the final output of a teacher network to teach information to a small student network.

[71] used not only the final output but also intermediate hidden layer values of the teacher network to train the student network and showed that using these intermediate layers can improve the performance of deeper and thinner student networks.

Net2Net [72] also uses a teacher-student network system with a function-preserving transform to initialize the parameters of the student network according to the parameters of the teacher network.

A recent work proposed in [73], which is inspired by [74], uses the Gramian matrix to represent the texture information of the input image. The Gramian matrix is generated by computing the inner product of feature vectors, it can contain the directionality between features, which can be thought of as texture information. Similar to this work, [73] also represent the flow of solving a problem by using Gramian matrix consisting of the inner products between features from two layers. The key difference is that Yim et al. compute the Gramian matrix across layers, whereas Gatys et al. compute the inner products between features within a layer.

- The extracted feature maps from two layers are used to generate the flow of solution procedure (FSP) matrix. The student DNN is trained to make its FSP matrix similar to that of the teacher DNN.

Suppose one of the selected layers generate the feature map $F^1 \in \mathbb{R}^{h\times w\times m}$, where h, w, and m represent the height, width, and number of channels respectively. The other selected layer generates the feature map $F^2 \in \mathbb{R}^{h\times w\times n}$. Then, the FSP matrix $G \in \mathbb{R}^{m\times n}$ is calculated by:

$$G_{i,j}(x; W) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F^1_{s,t,i}(x; W) \times F^2_{s,t,j}(x; W)}{h \times w} \tag{80}$$

where x and W represent the input image and the weights of the DNN respectively.

Assume that there are n FSP matrices $G_i^T, i = 1, \ldots, n$, which are generated by the teacher network, and n FSP matrices $G_i^S, i = 1, \ldots, n$, which are generated by the student network. The cost function of transferring the distilled knowledge task is defined as:

$$L_{FSP}(W_t, W_s) = \frac{1}{N}\sum_{x}\sum_{i=1}^{n} \lambda_i \times \|G_i^T(x; W_t) - G_i^S(x; W_s)\|_2^2 \tag{81}$$

where λ_i and N represent the weight for each loss term and the number of data points, respectively.

The authors also show the three usefulness of knowledge distillation: (1) Fast Optimization by finding good initial weight with distilled knowledge. (2) Improve the performance of a small network with fewer parameters. (3) Transfer knowledge from a deep and heavy DNN pretrained with a huge dataset to a small DNN.

Another recent work [75]: "Classical distillation methods transfer representations from a "teacher" neural network to a "student" network by matching their output activations. Recent methods also match the Jacobians, or the gradient of output activations with the input. However, this involves making some ad hoc decisions, in particular, the choice of the loss function. In this paper, we first establish an equivalence between Jacobian matching and distillation with input noise, from which we derive appropriate loss functions for Jacobian matching. We then rely on this analysis to apply Jacobian matching to transfer learning by establishing equivalence of a recent transfer learning procedure to distillation. We then show experimentally on standard image datasets that Jacobian-based penalties improve distillation, robustness to noisy inputs, and transfer learning."

4.4 From model predictions (Hypothesis transfer learning)

Another special group of methods is the so-called *hypothesis transfer learning*.

Stability and hypothesis transfer learning [76]

In [77] the authors have explored the setting *Metric Hypothesis Transfer Learning*, in which they assume that the source training samples are not accessible so one can only make use of the pre-learned source metric $\mathbf{M}_S$ to help learning the target metric $\mathbf{M}$. They have mainly provided some theoretical analysis showing that supervised regularized metric learning approaches using a biased regularization are well-founded. Their analysis is based on algorithmic stability arguments allowing one to derive generalization guarantees when a learning algorithm does not suffer too much from a little change in the training sample. Firstly they introduced a new notion of stability called *on-average-replace-two-stability* that is well-suited to regularized metric learning formulations. This notion allows one to prove a high probability generalization bound for metric hypothesis transfer learning achieving a fast converge rate in $O(\frac{1}{n})$ in the context of admissible, lipschitz and convex losses. Secondly, they provided a consistency result from which they justify the interest of *weighted biased regularization* of the form $\|\mathbf{M}-\beta\mathbf{M}_S\|$ where β is a parameter to set. They derive an approach for assessing this parameter without resorting to a costly parameter tuning procedure. They also provided an experimental study showing the effectiveness of transfer metric learning with weighted biased regularization in the presence of few labeled data both on standard metric learning an transfer learning tasks.

References

[1] V. Vapnik, "Principles of risk minimization for learning theory," in *Advances in neural information processing systems*, pp. 831–838, 1992.

[2] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[3] Y. Tang, J. Wang, X. Wang, B. Gao, E. Dellandréa, R. Gaizauskas, and L. Chen, "Visual and semantic knowledge transfer for large scale semi-supervised object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[4] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, pp. 1345–1359, Oct. 2010.

[5] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. PP, pp. 1–1, July 2014.

[6] J. Zhang, W. Li, and P. Ogunbona, "Transfer learning for cross-dataset recognition: a survey," *arXiv preprint arXiv:1705.04396*, 2017.

[7] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. 24th Int. Conf. Machine Learning*, (New York, NY, USA), pp. 193–200, ACM, 2007.

[8] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *Proc. 2010 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1855–1862, June 2010.

[9] G.-J. Qi, C. Aggarwal, Y. Rui, Q. Tian, S. Chang, and T. Huang, "Towards cross-category knowledge propagation for learning visual concepts," in *Proc. 2011 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 897–904, June 2011.

[10] Y. Lu, L. Chen, A. Saidi, E. Dellandrea, and Y. Wang, "Discriminative transfer learning using similarities and dissimilarities," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 3097–3110, July 2018.

[11] W. Ge and Y. Yu, "Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI*, vol. 6, 2017.

[12] V. Manjunatha, S. Ramalingam, T. K. Marks, and L. Davis, "Class subset selection for transfer learning using submodularity," *arXiv preprint arXiv:1804.00060*, 2018.

[13] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Int. Conf. Machine Learning*, 2007.

[14] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, p. 607, 1996.

[15] H. Wang, F. Nie, and H. Huang, "Robust and discriminative self-taught learning," in *International Conference on Machine Learning*, pp. 298–306, 2013.

[16] S. Li, K. Li, and Y. Fu, "Self-taught low-rank coding for visual learning," *IEEE Trans. Neural Netw. Learn. Syst.*, 2017.

[17] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, pp. 929–942, July 2010.

[18] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction.," in *AAAI*, vol. 8, pp. 677–682, 2008.

[19] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[20] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE transactions on neural networks*, vol. 12, no. 2, pp. 181–201, 2001.

[21] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *2013 IEEE Int. Conf. Computer Vision*, pp. 2200–2207, Dec. 2013.

[22] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[23] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 180–191, VLDB Endowment, 2004.

[24] L. Luo, L. Chen, S. Hu, Y. Lu, and X. Wang, "Discriminative and geometry aware unsupervised domain adaptation," *CoRR*, vol. abs/1712.10042, 2017.

[25] Z. Ding and Y. Fu, "Robust transfer metric learning for image classification," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 660–670, 2017.

[26] M. Chen, Z. Xu, K. Q. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1627–1634, Omnipress, 2012.

[27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[28] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[29] M.-S. Long, Y. Cao, J.-M. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. 32nd Int. Conf. Machine Learning*, pp. 97–105, 2015.

[30] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*, pp. 1180–1189, 2015.

[31] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4068–4076, 2015.

[32] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur, "Optimal kernel choice for large-scale two-sample tests," in *Advances in neural information processing systems*, pp. 1205–1213, 2012.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[34] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Adv. Neural Inf. Process Syst. (NIPS)*, 2016.

[35] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pp. 2960–2967, 2013.

[36] H. Lu, L. Zhang, Z. Cao, W. Wei, K. Xian, C. Shen, and A. van den Hengel, "When unsupervised domain adaptation meets tensor representations," in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 2017.

[37] B. Sun and K. Saenko, "Subspace distribution alignment for unsupervised domain adaptation.," in *BMVC*, pp. 24–1, 2015.

[38] J. Zhang, W. Li, and P. Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," *arXiv preprint arXiv:1705.05498*, 2017.

[39] N. Courty, R. Flamary, and D. Tuia, "Domain adaptation with regularized optimal transport," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 274–289, Springer, 2014.

[40] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[41] M. Perrot, N. Courty, R. Flamary, and A. Habrard, "Mapping estimation for discrete optimal transport," in *Advances in Neural Information Processing Systems*, pp. 4197–4205, 2016.

[42] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Advances in Neural Information Processing Systems*, pp. 3733–3742, 2017.

[43] C. Villani, *Optimal transport: old and new*, vol. 338. Springer Science & Business Media, 2008.

[44] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?," *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[45] M. Thorpe, S. Park, S. Kolouri, G. K. Rohde, and D. Slepčev, "A transportation l^ p lp distance for signal analysis," *Journal of Mathematical Imaging and Vision*, vol. 59, no. 2, pp. 187–210, 2017.

[46] P. P. Busto and J. Gall, "Open set domain adaptation," in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 1, p. 3, 2017.

[47] J. Zhang, Z. Ding, W. Li, and P. Ogunbona, "Importance weighted adversarial nets for partial domain adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[48] Z. Cao, M. Long, J. Wang, and M. I. Jordan, "Partial transfer learning with selective adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[49] S. Thrun, "Is learning the n-th thing any easier than learning the first?," in *Advances in Neural Information Processing Systems*, pp. 640–646, The MIT Press, 1996.

[50] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, vol. 6, no. Nov, pp. 1817–1853, 2005.

[51] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Advances in neural information processing systems*, pp. 41–48, 2007.

[52] Y. Amit, M. Fink, N. Srebro, and S. Ullman, "Uncovering shared structures in multiclass classification," in *Proceedings of the 24th international conference on Machine learning*, pp. 17–24, ACM, 2007.

[53] A. Quattoni, M. Collins, and T. Darrell, "Learning visual representations using images with captions," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[54] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 720–727, 2003.

[55] M. Fink, "Object classification from a single example utilizing class relevance metrics," in *Advances in neural information processing systems*, pp. 449–456, 2005.

[56] S. Parameswaran and K. Q. Weinberger, "Large margin multi-task metric learning," in *Advances in neural information processing systems*, pp. 1867–1875, 2010.

[57] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[58] J. Yang, R. Yan, and A. G. Hauptmann, "Cross-domain video concept detection using adaptive svms," in *Proc. 15th Int. Conf. Multimedia*, (New York, NY, USA), pp. 188–197, ACM, 2007.

[59] Y. Aytar and A. Zisserman, "Tabula rasa: Model transfer for object category detection," in *Proc. 2011 Int. Conf. Computer Vision*, (Washington, DC, USA), pp. 2252–2259, IEEE Computer Society, 2011.

[60] X. Li, *Regularized adaptation: Theory, algorithms and applications*, vol. 68. Citeseer, 2007.

[61] W. Jiang, E. Zavesky, S.-F. Chang, and A. Loui, "Cross-domain learning methods for high-level visual concept classification," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 161–164, IEEE, 2008.

[62] T. Tommasi, F. Orabona, and B. Caputo, "Safety in numbers: Learning categories from few examples with multi model knowledge transfer," in *Proc. 2010 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3081–3088, June 2010.

[63] G. 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*, pp. 1661–1668, IEEE, 2006.

[64] I. Kuzborskij, F. Orabona, and B. Caputo, "From n to n+1: Multiclass transfer incremental learning," in *Proc. 2013 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3358–3365, June 2013.

[65] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[66] M. C. Burl and P. Perona, "Recognition of planar object classes," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pp. 223–230, IEEE, 1996.

[67] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, pp. II–II, IEEE, 2003.

[68] X. Yu and Y. Aloimonos, "Attribute-based transfer learning for object categorization with zero/one training example," *Computer Vision–ECCV 2010*, pp. 127–140, 2010.

[69] M. Rosen-Zvi, C. Chemudugunta, T. Griffiths, P. Smyth, and M. Steyvers, "Learning author-topic models from text corpora," *ACM Transactions on Information Systems (TOIS)*, vol. 28, no. 1, p. 4, 2010.

[70] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[71] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[72] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

[73] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[74] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[75] S. Srinivas and F. Fleuret, "Knowledge transfer with Jacobian matching," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 4730–4738, PMLR, 10–15 Jul 2018.

[76] I. Kuzborskij and F. Orabona, "Stability and hypothesis transfer learning," in *International Conference on Machine Learning*, pp. 942–950, 2013.

[77] M. Perrot and A. Habrard, "A theoretical analysis of metric hypothesis transfer learning," in *International Conference on Machine Learning*, pp. 1708–1717, 2015.