Transferability in Deep Learning: A Survey

Junguang Jiang jiangjunguang1123@outlook.com
School of Software, BNRist, Tsinghua University
Beijing 100084, China

Yang Shu * shu-y18@mails.tsinghua.edu.cn
School of Software, BNRist, Tsinghua University
Beijing 100084, China

Jianmin Wang jimwang@tsinghua.edu.cn
School of Software, BNRist, Tsinghua University
Beijing 100084, China

Mingsheng Long † mingsheng@tsinghua.edu.cn
School of Software, BNRist, Tsinghua University
Beijing 100084, China

Editor: Leslie Pack Kaelbling

Abstract

The success of deep learning algorithms generally depends on large-scale data, while humans appear to have inherent ability of knowledge transfer, by recognizing and applying relevant knowledge from previous learning experiences when encountering and solving unseen tasks. Such an ability to acquire and reuse knowledge is known as *transferability* in deep learning. It has formed the long-term quest towards making deep learning as *data-efficient* as human learning, and has been motivating fruitful design of more powerful deep learning algorithms. We present this survey to connect different isolated areas in deep learning with their relation to transferability, and to provide a unified and complete view to investigating transferability through the whole *lifecycle* of deep learning. The survey elaborates the fundamental goals and challenges in parallel with the core principles and methods, covering recent cornerstones in deep architectures, pre-training, task adaptation and domain adaptation. This highlights unanswered questions on the appropriate objectives for learning transferable knowledge and for adapting the knowledge to new tasks and domains, avoiding catastrophic forgetting and negative transfer. Finally, we implement a benchmark and an open-source library, enabling a fair evaluation of deep learning methods in terms of transferability.

Keywords: Deep learning, transferability, pre-training, adaptation, library, benchmark

∗. Equal contribution
†. Correspondence to: Mingsheng Long <mingsheng@tsinghua.edu.cn>.

Contents

1 Introduction **3**
1.1 Terminology . . . 4
1.2 Overview . . . 5

2 Pre-Training **7**
2.1 Pre-Training Model . . . 7
2.2 Supervised Pre-Training . . . 9
2.2.1 Meta-Learning . . . 10
2.2.2 Causal Learning . . . 12
2.3 Unsupervised Pre-Training . . . 14
2.3.1 Generative Learning . . . 14
2.3.2 Contrastive Learning . . . 16
2.4 Remarks . . . 19

3 Adaptation **21**
3.1 Task Adaptation . . . 21
3.1.1 Catastrophic Forgetting . . . 22
3.1.2 Negative Transfer . . . 24
3.1.3 Parameter Efficiency . . . 26
3.1.4 Data Efficiency . . . 27
3.1.5 Remarks . . . 29
3.2 Domain Adaptation . . . 30
3.2.1 Statistics Matching . . . 34
3.2.2 Domain Adversarial Learning . . . 36
3.2.3 Hypothesis Adversarial Learning . . . 39
3.2.4 Domain Translation . . . 40
3.2.5 Semi-Supervised Learning . . . 42
3.2.6 Remarks . . . 44

4 Evaluation **45**
4.1 Datasets . . . 45
4.2 Library . . . 46
4.3 Benchmark . . . 47
4.3.1 Pre-Training . . . 47
4.3.2 Task Adaptation . . . 49
4.3.3 Domain Adaptation . . . 49

5 Conclusion **50**

1. Introduction

Deep learning (LeCun et al., 2015) is a class of machine learning algorithms that utilize multiple processing layers to learn representations of data with multiple levels of abstraction. These multiple processing layers, also called deep neural networks (DNNs), are empowered with the ability to discover different explanatory factors of variation behind the intricate structured data (Bengio et al., 2013). With essential advances in network architectures, training strategies and computation devices, deep learning has made breakthroughs or even revolutions in various areas, such as computer vision (Krizhevsky et al., 2012; He et al., 2016), natural language processing (Radford et al., 2018), speech processing (Amodei et al., 2016), computational biology (Senior et al., 2020), games (Silver et al., 2016; Vinyals et al., 2019) and so forth. Despite its great success in these important areas, deep learning is still faced with the grand challenge of *data efficiency*. Most mainstream deep learning methods require big datasets in the order of millions or even trillions to achieve good performance, yet collecting and annotating such huge amount of data for each new task or domain are expensive and even prohibitive. This data efficiency challenge heavily impedes the adoption of deep learning to a wider spectrum of application scenarios.

An effective solution to this challenge is to explore the *transferability* in deep learning. Transferability is a foundational ability of human learning: human beings can gain relevant knowledge from other related problems and apply it to handle new problems with extremely few samples (Thrun and Pratt, 1998). In deep learning, transferability refers to the ability of deep neural networks to extract transferable representations from some source tasks and then adapt the gained representations to improve learning in related target tasks (Bengio, 2012). Recent advances in deep learning reveal that deep models trained via upstream tasks on large-scale data tend to yield good transferability to a variety of downstream tasks, such as visual object detection (Ren et al., 2015), natural language understanding (Devlin et al., 2019), to name a few. Transferability has become the central property of deep learning for improving data efficiency. It is on par with generalizability, interpretability, and robustness for bridging the gap between machine learning and human learning.

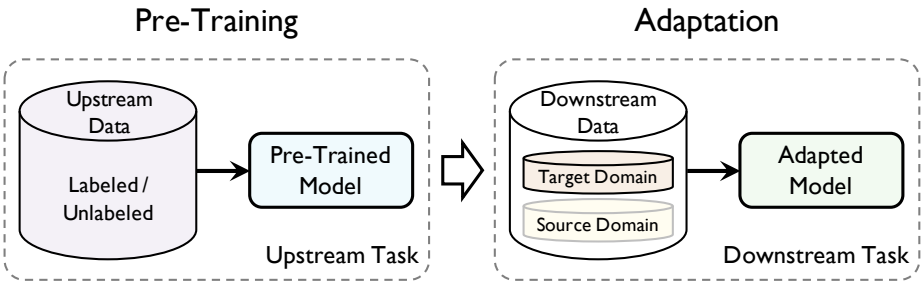


Figure 1: The two-stage lifecycle of most deep learning applications. In the first stage, the deep model is *pre-trained* on an upstream task with large-scale data (labeled or unlabeled) for gaining transferable knowledge. In the second stage, the pre-trained model is *adapted* to a downstream task in the target domain with labeled data; If the downstream task only has unlabeled data, then additional labeled data from another source domain of identical learning task but different data distribution will be used to improve performance.

Towards gaining and applying knowledge with good transferability, the lifecycle of many deep learning applications is divided into two stages: pre-training and adaptation (Figure 1). The goal of the *pre-training* stage is to gain the transferable knowledge. The deep models are pre-trained on an upstream task with large-scale data (either labeled or unlabeled) to learn disentangled representations or reusable parameters that are transferable to a variety of downstream tasks. The goal of the *adaptation* stage is to reuse the transferable knowledge. The pre-trained models are adapted to a downstream task in the target domain with labeled data, and the previously learned knowledge enables better generalization with fewer labeled samples. When the downstream task only has unlabeled data, additional labeled data from another source domain of identical learning task but different data distribution will be used to improve the data efficiency of the adapted model (Ganin and Lempitsky, 2015).

It is helpful to highlight the difference underlying the transferability in the two stages. The pre-training stage focuses mainly on the *generic* transferability, i.e., obtaining a general transferable representation that can improve the performance of as many downstream tasks as possible. In contrast, the adaptation stage pays attention to the *specific* transferability, i.e., how to exploit the transferable knowledge in pre-trained models for a specific kind of downstream tasks, or how to improve the transferability between related domains of the same downstream task. The generic transferability is attractive since it may benefit many downstream tasks without additional cost or special design. Yet it may ignore the special structures of downstream tasks that are crucial for stronger transferability, thus the specific transferability is still necessary in many cases. Recently, the gap between the pre-training stage and the adaptation stage is getting closer. Several pre-training methods are designed to obtain fast model adaptation ability in the adaptation stage (Finn et al., 2017), while some adaptation methods try to convert downstream tasks into pre-training tasks to make full use of the generic transferability of pre-trained models (Brown et al., 2020).

Transferability lies at the core of the whole lifecycle of deep learning, yet different areas such as domain adaptation (Zhuang et al., 2021) and continual learning (Delange et al., 2021), mainly explore transferability in a partial regime of the lifecycle. This is not enough to achieve a complete picture of transferability. Thereby, we present this survey to connect different isolated areas in deep learning with their relation to transferability, and to provide a unified and complete view to investigate transferability through the whole lifecycle of deep learning. Due to the broadness of the scope and the limitation of the space, we do not aim to cover all methods towards transferability. Instead, we elaborate on the core principles and methods and then give a brief review of the expanded literature. We further implement `TLlib`, a high-quality open library to provide a fair evaluation of typical methods. We hope this survey can highlight the grand picture of transferability in deep learning, and provide a useful navigation to researchers interested in improving the data efficiency of deep learning.

1.1 Terminology

Foremost, we give several definitions related to transferability, and the summary of notations and their descriptions used in this survey can be found in Table 1. Denote the input space as $\mathcal{X}$ and the output space as $\mathcal{Y}$, and assume that there exists an unknown labeling function $f : \mathcal{X} \mapsto \mathcal{Y}$. Formally, a *task* corresponds to learning an underlying labeling function f. To learn a task, we first collect a set of samples $\widehat{\mathcal{D}} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, which are drawn independently

Table 1: Notations and descriptions used in the survey.

$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$\mathcal{D}$	A fixed but unknown distribution over $\mathcal{X}$
$\widehat{\mathcal{D}}$	Empirical distribution of a sample drawn i.i.d. from $\mathcal{D}$
$P(\cdot)$	Probability of an event
$\mathbb{E}(\cdot)$	Expectation of a random variable
$\mathcal{U}$	Upstream data
$\mathcal{S}$	Source domain in downstream data
$\mathcal{T}$	Target domain in downstream data
$\mathcal{H}$	Hypothesis space
h	A hypothesis in the hypothesis space $\mathcal{H}$
ψ	Feature generator
θ	Hypothesis parameter
$\mathbf{x}$	Model input
$\mathbf{y}$	Model output
$\mathbf{z}$	Hidden activation of the feature generator
D	A discriminator to distinguish different distributions

and identically distributed (i.i.d.) from some fixed but unknown distribution $\mathcal{D}$. Formally, a *domain* is a marginal probability distribution $P(\mathbf{X})$ defined on a certain input space $\mathcal{X}$. Consider a set of hypotheses $\mathcal{H}$ and a specific loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$, the objective of the learner is to select a hypothesis $h \in \mathcal{H}$ that yields the lowest generalization error, $\min_{h\in\mathcal{H}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \ell(h(\mathbf{x}), f(\mathbf{x}))$.

Definition 1 (Transferability) *Given a source domain $\mathcal{S}$ with learning task $t_\mathcal{S}$ and a target domain $\mathcal{T}$ with learning task $t_\mathcal{T}$, transferability is the ability of gaining transferable knowledge from $t_\mathcal{S}$ on $\mathcal{S}$ and reusing the knowledge to decrease the generalization error of $t_\mathcal{T}$ on $\mathcal{T}$, under the distribution shift $\mathcal{S} \neq \mathcal{T}$ or the task discrepancy $t_\mathcal{S} \neq t_\mathcal{T}$.*

In the deep learning lifecycle (Figure 1), the pre-training stage aims to *gain* transferable knowledge via learning on upstream task with large-scale data, while the adaptation stage aims to *reuse* the pre-trained knowledge to improve the data efficiency in downstream tasks. The upstream and downstream are different in both learning tasks and data distributions. To conform with the literature, in the pre-training stage, we will replace the notions of source domain/task with the widely-used upstream data/task, denoted as $\mathcal{U}$ and $t_\mathcal{U}$ respectively.

1.2 Overview

The survey is organized around how to acquire and utilize the transferability in deep learning throughout its whole lifecycle, including *pre-training*, *adaptation*, and *evaluation* (Figure 2).

- **Pre-Training.** We first briefly discuss some important model *architectures* that make pre-trained representations transferable. Then we elaborate on *supervised pre-training* and *unsupervised pre-training*, which are distinguished by the availability of labeled or unlabeled data for pre-training. In *supervised pre-training*, we cover both standard practices commonly used in the industry and research advances in academia to acquire

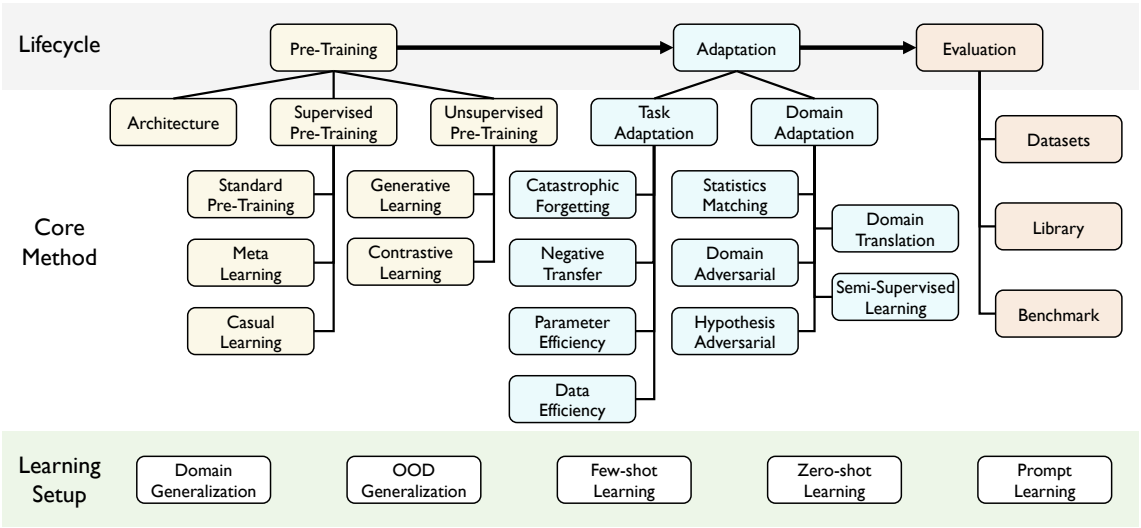


Figure 2: Overview of this survey. The survey is organized around the lifecycle (pre-training, adaptation, and evaluation) of deep learning applications and focuses on the core problems and methods towards transferability. Besides, we briefly review related learning setups.

transferability on the labeled data. In *unsupervised pre-training*, we cover the latest designs of proper pre-training tasks on unlabeled data to gain transferability.

- **Adaptation.** We mainly elaborate on *task adaptation* and *domain adaptation*, which are divided by whether there exists another related source domain in addition to the pre-trained model for boosting the downstream task performance. In *task adaptation*, we first pinpoint several open problems caused by the discrepancy between upstream tasks and downstream tasks, then illustrate how different task adaptation paradigms (Yosinski et al., 2014; Brown et al., 2020) close the task discrepancy to better utilize the transferability. In *domain adaptation*, we first pinpoint the most influential theories for closing the distribution shift (Ben-David et al., 2006, 2010a), then elaborate how to derive solid learning algorithms (Long et al., 2015; Ganin and Lempitsky, 2015) from these theories to enhance the transferability of deep models across domains.

- **Evaluation.** We mainly investigate the transferability gained and reused by different pre-training and adaptation methods on several large-scale datasets released recently in the literature. Note that we omit some small-scale and relatively obsolete datasets to make our benchmark concise and easy to report. To facilitate fair evaluation and full reproduction of existing algorithms, we open source `TLlib`, a high-quality library along with this survey at https://github.com/thuml/Transfer-Learning-Library.

Pre-training and adaptation lie at the core methods towards transferability. In parallel with them, there are some fields that are also closely related to the transferability in deep learning, such as *domain generalization* (Gulrajani and Lopez-Paz, 2021), *out-of-distribution (OOD) generalization* (Bengio et al., 2021), *few-shot learning* (Chen et al., 2019a), etc. Recent evaluation shows that these learning setups can largely benefit from the advancement in pre-training and adaptation and we will give a brief review to them in the related sections.

2. Pre-Training

Despite yielding unprecedented performances on various machine learning tasks, the deep learning methods require large amounts of labeled data to generalize well. This *data hungry* nature limits their application to a wide variety of domains and tasks, especially to scenarios short of data and annotations. Pre-training, which obtains transferable representations or models from upstream tasks with large-scale data to boost the performance on downstream tasks, is one of the most common and practical solutions to the problem of data scarcity. In this section, we will first review some important model architectures that have a great impact on the transferability of pre-trained representations in Section 2.1. Then we elaborate on how to *gain knowledge* of improved transferability via supervised pre-training on large-scale labeled data in Section 2.2 and via unsupervised pre-training on much larger unlabeled data in Section 2.3. Figure 3 overviews the recent cornerstones of pre-training methods.

2.1 Pre-Training Model

Pre-training has a big interplay with the model architecture. On the one hand, pre-training techniques, such as greedy layerwise unsupervised pre-training (Bengio et al., 2007), have eased the training of many deep architectures. On the other hand, as neural networks evolve from shallow to deep, they have a larger capacity to capture knowledge by pre-training from large-scale data, which increases their transferability to downstream tasks.

Model architecture has a great influence on the transferability of knowledge obtained via pre-training. Kornblith et al. (2019) find that the performance of the pre-trained models on the downstream tasks is highly correlated with the accuracy on the pre-training tasks, which suggests that improving the performance on the pre-training task serves as a direct way for improving transferability. The depth of the architecture, or more precisely, the *capacity* of

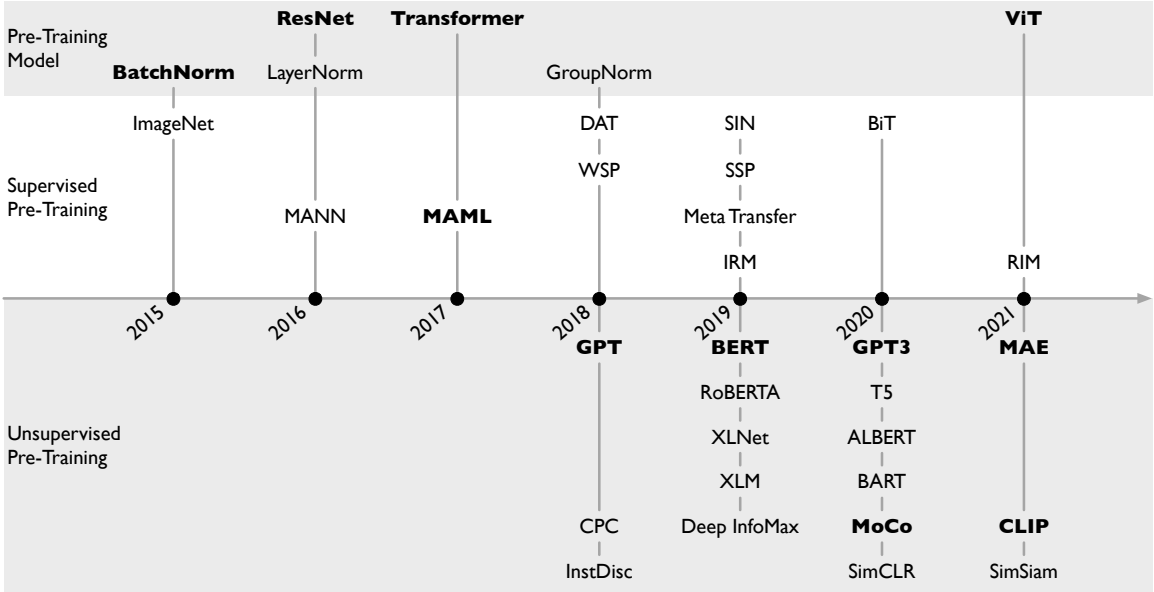


Figure 3: Cornerstones of pre-training methods for *gaining* knowledge of transferability.

the model, is deemed the most critical factor to its transferability. However, training very deep neural networks have remained a grand difficulty for decades. He et al. (2016) observe a degradation of training accuracy by increasing the network depth, which implies that deeper models are more difficult to optimize. Instead of fitting a desired mapping $h(\mathbf{x})$ by a few stacked layers, they proposed Residual Network (ResNet) to explicitly fit a residual mapping $\delta(\mathbf{x}) := h(\mathbf{x}) - \mathbf{x}$ and then recast the original mapping into $\delta(\mathbf{x}) + \mathbf{x}$. As a result, ResNet improves feature and gradient flows and enables end-to-end training of hundreds of and even thousands of layers, allowing the capacity of pre-trained models to scale up easily. Ioffe and Szegedy (2015) hypothesize that the optimization difficulty also comes from the internal covariate shift caused by layerwise transformation. To stabilize training very deep models, they proposed Batch Normalization (BatchNorm) (Ioffe and Szegedy, 2015), which performs normalization for each training mini-batch within the architecture. This design is extensively used by ResNet. Kolesnikov et al. (2020) find that BatchNorm is suboptimal for transfer due to the requirement of distribution-dependent moving averaged statistics. They proposed Big Transfer (BiT) to replace BatchNorm by GroupNorm (Wu and He, 2018), which generates pre-trained models of strong performance on downstream tasks.

The pre-training paradigm also reshapes the design of model architectures. In classic supervised learning, models usually have strong *inductive bias* such as the local connectivity assumption in Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). A strong inductive bias makes pre-training of deep models more data-efficient and generalize better when training data is scarce, yet on the other hand, it also limits the expressiveness and transferability of the deep models when there is large-scale data for pre-training. Thus, Transformer (Vaswani et al., 2017) removes the local connectivity assumption and models the global dependencies between every two tokens. The connection weights are dynamically computed by the self-attention mechanism and then the feature aggregation in Transformer depends on these attentions calculated from the input sequence, while the token positions in the sequence are encoded by positional embedding. Transformers are powerful for sequence modeling in natural language processing, and Vision Transformer (ViT) (Dosovitskiy et al., 2021) extends them to computer vision. ViT splits an image into fixed-size patches, linearly embeds each of them, adds positional embeddings, and feeds the resulting sequence of vectors to a standard Transformer encoder. In summary, Transformer makes least assumptions on the structural information of data, which makes Transformer an *expressive* architecture for storing the transferable knowledge extracted by pre-training on large amounts of training data (Devlin et al., 2019; Radford et al., 2018).

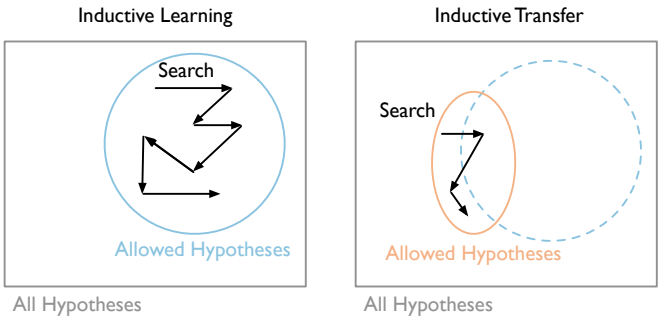


Figure 4: Designed inductive bias (left) and learned inductive bias from pre-training (right).

In some sense, pre-training provides a *learned* inductive bias for the downstream tasks (Torrey and Shavlik, 2010). Many downstream tasks only have hundreds or thousands of labeled samples, yet the pre-trained Transformers with hundreds of millions of parameters can generalize well after fine-tuning on such small data. To explain this phenomenon, Aghajanyan et al. (2021) empirically show that pre-training minimizes the intrinsic dimension (Li et al., 2018), which measures the number of parameters required to closely approximate the optimization problem. Further, an intrinsic-dimension generalization bound is given, indicating that the pre-trained parameters implicitly affect the *inductive bias* of models and a larger pre-trained model might correspond to a smaller allowed hypothesis space during fine-tuning (see Figure 4). The success of Transformer reveals that as the amount of pre-training data increases, the learned inductive bias is able to outperform the manually designed inductive bias in terms of transferability.

2.2 Supervised Pre-Training

Supervised pre-training aims to obtain models on large-scale labeled data and then transfers these models to boost downstream tasks (see Figure 5). Supervised pre-training is commonly employed in computer vision, where image classification on ImageNet (Deng et al., 2009; Russakovsky et al., 2015) is often used as the pre-training task. The pre-trained models can be transferred to downstream tasks by reusing the representations from the feature generator (Sermanet et al., 2013). Donahue et al. (2014) find that the generic visual representations pre-trained on ImageNet outperforms many conventional feature descriptors on various object recognition tasks. Yosinski et al. (2014) find that transferring the pre-trained models by fine-tuning the whole models yields better generalization performance on new tasks.

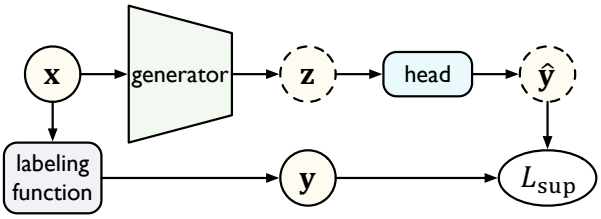


Figure 5: Standard supervised pre-training. The model is composed of a feature generator and a task-specific head. The goal is to obtain a feature generator capturing transferable knowledge from large-scale labeled data. After pre-training, the feature generator is adapted to downstream tasks, while the task-specific head is usually discarded.

Among the factors that influence the transferability of pre-trained models, the quantity and quality of the pre-training data might be the most important. BiT (Kolesnikov et al., 2020) emphasizes that training on larger datasets is vital for better transferability. Yet the data labeling is labor-exhaustive and time-consuming, which limits the possible size of the annotation data. To break this limitation, Mahajan et al. (2018) explore Weakly Supervised Pre-training (WSP) on IG-1B-Targeted, a dataset of billions of images with social media hashtags. Yalniz et al. (2019) further explore web-scale Semi-Supervised Pre-training (SSP) on YFCC100M, a dataset of billions of unlabeled images along with a relatively smaller set of task-specific labeled data. These methods improve clearly against the counterpart trained

with only clean labeled data and achieve stronger transfer performance. On the other hand, Domain Adaptive Transfer (DAT) (Ngiam et al., 2018) studies the influence of data quality and finds that using more data does not necessarily lead to better transferability, especially when the dataset is extremely large. Thus, an importance weighting strategy is proposed to carefully choose the pre-training data that are most relevant to the target task. Cui et al. (2018) also find that pre-training on more similar upstream data improves transferability to fine-grained downstream tasks. They propose to estimate domain similarity via the Earth Mover's Distance to choose proper pre-training data. Geirhos et al. (2019) find that models trained supervisedly on ImageNet are biased towards textures in images, and propose to pre-train with a Stylized ImageNet (SIN), which fixes the texture bias and encourages the models to learn shape-based representations of better transferability.

While standard supervised pre-training is powerful when there are enough labeled data, it still has drawbacks that may limit the transferability of the model. For instance, standard supervised pre-trained models are vulnerable to adversarial examples (Goodfellow et al., 2015), and Salman et al. (2020) enhance the adversarial robustness of the pre-trained models to achieve better transferability. In addition, there are alternative pre-training methods for improving the transferability of deep models. Section 2.2.1 will elaborate on meta-learning, which aims to obtain pre-trained models that adapt to downstream tasks with less training time and less training data. Section 2.2.2 will review causal learning, which aims to obtain distributionally robust and generalizable pre-trained models.

2.2.1 Meta-Learning

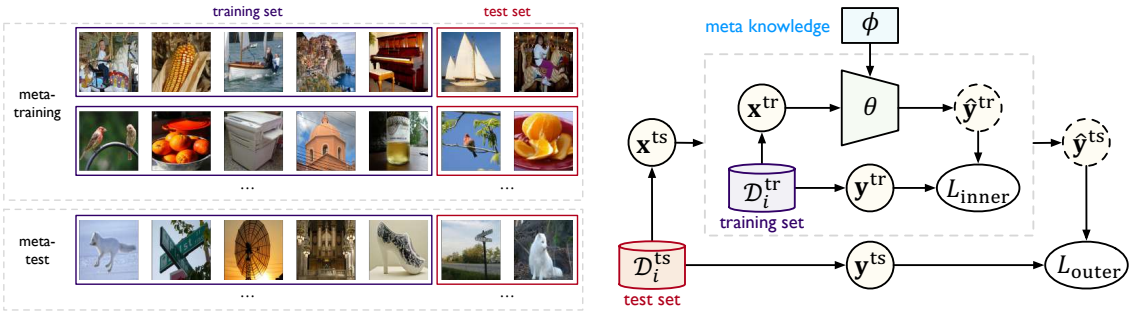
Standard supervised pre-training gains transferable representations to boost the learning of new tasks. However, it still requires to fine-tune the pre-trained models with hundreds or thousands of labeled data and with many gradient updates when adapting to the new task. In contrast, people have the ability to quickly adapt to different related new tasks with few labeled data. Meta-learning, also known as learning to learn (Schmidhuber, 1987), aims to pursue such kind of *efficient* transferability in the pre-training stage.

The core idea of meta-learning is to equip the model with some *meta knowledge* ϕ that captures intrinsic properties of different learning tasks, which is called *meta-training*. When facing a new task, the learned meta knowledge could help the target model θ adapt to the task faster, which is called *meta-testing*. Meta-learning is based on a simple machine learning principle that test and training conditions should be matched. As shown in Figure 6(a), to simulate the fast adaptation condition during meta-testing, the meta-training data is constructed into a collection of n learning tasks, and each task $i \in [n]$ contains a training set $\mathcal{D}_i^{\text{tr}}$ for adaptation to this task and a test set $\mathcal{D}_i^{\text{ts}}$ for evaluation[1]. As shown in Figure 6(b), the learning objective of meta-training is a bi-level optimization problem,

$$\phi^* = \arg\max_{\phi} \sum_{i=1}^{n} \log P(\theta_i(\phi)|\mathcal{D}_i^{\text{ts}}), \quad \text{where } \theta_i(\phi) = \arg\max_{\theta} \log P(\theta|\mathcal{D}_i^{\text{tr}}, \phi). \tag{1}$$

Here the inner level optimization updates the model θ with the training set $\mathcal{D}_i^{\text{tr}}$ using meta knowledge ϕ, and the outer level optimization evaluates the updated model with the test

1. $\mathcal{D}^{\text{ts}}$ is a *surrogate* test set used during meta-training to simulate different tasks and improve the model. It is different from the true test set in the general setting in machine learning.



(a) Learning Setup (b) Architecture

Figure 6: Learning setup and architecture for meta-learning. (a) Meta-learning consists of two phases, *meta-training* and *meta-testing*. Meta-training gains meta knowledge ϕ from training tasks to help the model θ adapt quickly to a new task in meta-testing, where each task consists of a training set and a test set. (b) In the inner level optimization, the model θ is updated with the training set $\mathcal{D}_i^{\text{tr}}$ using meta knowledge ϕ. In the outer level optimization, the updated model is evaluated on the test set $\mathcal{D}_i^{\text{ts}}$ to find better meta knowledge ϕ.

set $\mathcal{D}_i^{\text{ts}}$ to find better meta knowledge of stronger transferability. The key to enhancing the transferability of meta-learning methods is to design a proper form of meta knowledge.

Memory-Based Meta-Learning considers *memory mechanisms* as the meta knowledge. A controller writes knowledge extracted from training data $\mathcal{D}_i^{\text{tr}}$ into the memory, and reads from the memory to adapt the base learner θ to make predictions on test data $\mathcal{D}_i^{\text{ts}}$. The parameter of the controller is updated to find transferable knowledge. Memory-Augmented Neural Network (MANN) (Santoro et al., 2016) stores bound sample representation-class label information in the external memory, which can then be retrieved as features for making predictions when a sample from the same class is presented. Meta Network (Munkhdalai and Yu, 2017) designs another memory mechanism where a base learner provides information about the status of the current task while the meta learner interacts with the external memory to generate parameters for the base learner to quickly learn the new task. Memory-based meta-learning methods improve transferability in various downstream tasks, such as few-shot classification and reinforcement learning. However, they require a careful design of the black-box architecture to incorporate the memory mechanism, and it is unclearer what is stored and retrieved in the memory and why it helps adapt the model.

Optimization-Based Meta-Learning considers a good *initialization* of the model as the meta knowledge. The motivation of Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) is to explicitly seek for an initialization that is most transferable for fine-tuning, i.e., only a small amount of gradient steps and a few labeled data are needed for the model to generalize to a new task. To learn such an initialization, for each sampled task $i \in [n]$, the model ϕ is first updated on its training data $\mathcal{D}_i^{\text{tr}}$ using one gradient step of size α,

$$\theta_i = \phi - \alpha \nabla_\phi L(\phi, \mathcal{D}_i^{\text{tr}}). \tag{2}$$

which mimics the situation of fine-tuning the model from the starting point of ϕ. As meta knowledge, ϕ should have good transferablity, such that for all tasks $i \in [n]$, the fine-tuned

parameters θ_i could perform well on the test set $\mathcal{D}_i^{\text{ts}}$,

$$\min_{\phi} \sum_{i=1}^{n} L(\theta_i(\phi), \mathcal{D}_i^{\text{ts}}) = \sum_{i=1}^{n} L(\phi - \alpha \nabla_{\phi} L(\phi, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}}). \tag{3}$$

The meta knowledge of MAML is high-dimensional, hindering MAML from deeper models. To tackle it, Meta Transfer (Sun et al., 2019a) uses standard pre-training for initialization and performs meta-training with light-weight neuron operations (e.g. scaling and shifting over tasks), which reduces the training tasks needed to acquire the meta knowledge. Raghu et al. (2020) find that feature reuse of the backbone is the predominant reason for efficient learning on downstream tasks with MAML. They thus propose the Almost No Inner Loop algorithm, which performs inner loop updates and task adaptation only on the task-specific head layer. Another limitation of MAML is that the fixed meta knowledge is globally shared by all tasks. To break this, Latent Embedding Optimization (Rusu et al., 2019) performs gradient-based meta-learning in a low-dimensional latent space, and learns data-dependent latent embedding as meta knowledge to generate target model parameters. Yao et al. (2019) perform Hierarchically Structured Meta-Learning over hierarchical tasks based on clustering structures and learns to tailor transferable meta knowledge to different tasks.

While meta-learning methods enable fast model adaptation across tasks, they are weak in transferring to data from different domains, and some sophisticated methods even perform worse than standard pre-training baselines (Chen et al., 2019a). Thus, Omni-Training (Shu et al., 2021a) incorporates both standard pre-training and meta-training in a framework with a tri-flow architecture to equip the pre-trained model with both domain transferability across different distributions and task transferability for fast adaptation across related tasks.

2.2.2 Causal Learning

It remains difficult for supervised pre-training to obtain a transferable representation that generalizes well to an out-of-distribution (OOD) domain (Bengio et al., 2021). In contrast, humans have the ability to adapt to different domains or new environments. Causal learning aims to pursue such kind of *extrapolated* transferability in the pre-training stage.

The core idea of causal learning is to equip the model with some *causal mechanisms* that capture independent and disentangled aspects of the complex real-world distributions. When the distribution changes, only one or several causal mechanisms change, with others remaining invariant, which could result in better *out-of-distribution* (OOD) generalization. The causal mechanisms are described by Structural Causal Models. As shown in Figure 7, causal mechanisms consider a set of variables as the vertices of a directed acyclic graph, and each edge represents a mechanism of direct causation in that the parents directly affect the assignment of the child. This induces a canonical factorization of the joint distribution of these variables into the disentangled distribution of them conditioned on their parents. The independent causal mechanism principle states that given its mechanism, the conditional distribution of each variable does not inform or influence the other mechanisms (Schölkopf et al., 2012; Peters et al., 2017). This implies that small distribution changes should only affect the causal mechanisms along with the disentangled factorization in a sparse and local way (Schölkopf et al., 2021), thereby enabling transferability towards different distributions. The key problem of causal learning is to obtain the variables governed by independent causal

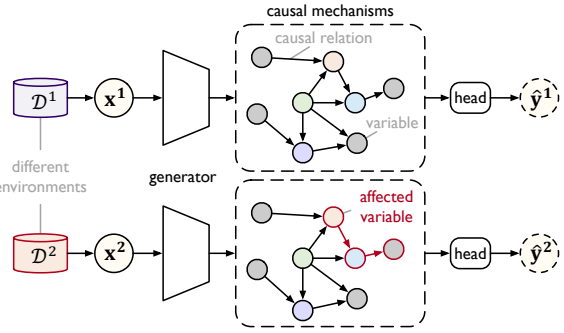


Figure 7: Causal mechanisms consider a set of observations or variables as the vertices of a directed acyclic graph, where each edge corresponds to a mechanism of direct causation. Causal learning seeks a model with variables governed by certain causal mechanisms, and if the environment or distribution changes, only part of the causal mechanisms will be affected.

mechanisms. One way is to explicitly introduce independence with the *modular* models. Another common practice is to leverage the *invariance* assumption that causal relationships remain invariant across distributions.

Modular Model. Recurrent Independent Mechanism (RIM) (Goyal et al., 2021) takes a modular model composed of several modules of different functions, where each module is a recurrent cell such as LSTM or GRU (Cho et al., 2014) and represents a causal mechanism. To obtain independence in distinct modules, RIM introduces attention between the hidden states of each module and the current inputs. For specific inputs, only the most relevant modules with larger attention are activated and updated, which forms competition between different modules and encourages their independence. RIM is shown to capture independent causal mechanisms and generalize well over different temporal patterns.

Invariant Learning. The invariance assumption indicates that the conditional probability of the target output given its direct cause should be invariant across all environments or distributions. Invariant Causal Prediction (ICP) (Peters et al., 2016) uncovers independent causal mechanisms by performing a statistical test to find the subset of the variables satisfying the invariance assumption. Invariant Risk Minimization (IRM) (Arjovsky et al., 2019) extends this idea to representation learning and learns a good representation such that the conditional probability of the target output given the representation should be invariant across training environments. Formally, given a data representation $\psi : \mathcal{X} \to \mathcal{Z}$ and training environments $\mathcal{E}^{\text{tr}}$, the conditional probability between the representation and the output is invariant if there is a classifier $h : \mathcal{Z} \to \mathcal{Y}$ simultaneously optimal for all the environments. This can be formalized as the following constrained optimization problem,

$$\min_{\psi:\mathcal{X}\to\mathcal{Z},h:\mathcal{Z}\to\mathcal{Y}} \sum_{e\in\mathcal{E}^{\text{tr}}} \epsilon^e(h\circ\psi), \quad \text{subject to } h \in \arg\min_{\bar{h}:\mathcal{Z}\to\mathcal{Y}} \epsilon^e(\bar{h}\circ\psi), \text{ for all } e \in \mathcal{E}^{\text{tr}}, \tag{4}$$

where $\epsilon^e(h\circ\psi)$ refers to the expected error of the predictor $h\circ\psi$ on the environment e. The transferability across environments relies on how the invariance across training environments implies invariance across all environments. Thus, the diversity of training environments is important for gaining transferability. IRM can be extended to complex situations where the causal relations are defined on some latent variables that need to be extracted from data.

2.3 Unsupervised Pre-Training

Being a canonical successful approach, supervised pre-training still requires a large amount of labeled data which are expensive to annotate and only available in certain fields. This hinders pre-training on huge-scale data and limits its transferability to particular tasks. To break this shackle, unsupervised learning (Bengio, 2012), typically in the form of self-supervised learning, is used for pre-training on very large unlabeled data to acquire generally transferable knowledge. To improve the transferability on downstream tasks, it is crucial to design a proper self-supervised task for pre-training. According to the type of task, we can divide common unsupervised pre-training methods into generative learning and contrastive learning, which will be discussed in Sections 2.3.1 and 2.3.2 respectively.

2.3.1 Generative Learning

Generative learning is underpinned by the idea of learning to generate data distribution $P(\mathbf{X})$ for unsupervised pre-training. It aims to learn the intrinsic representation in data and has been commonly used for pre-training deep neural networks (Bengio et al., 2007). As shown in Figure 8, we employ an encoder f_θ that maps the perturbed input $\tilde{\mathbf{x}}$ into a latent representation $\mathbf{z} = f_\theta(\tilde{\mathbf{x}})$ and a decoder g_θ that maps the representation back to derive a reconstructed version of the input $\widehat{\mathbf{x}} = g_\theta(\mathbf{z})$. The model is then optimized by minimizing the reconstruction error $L_{\text{gen}}(\widehat{\mathbf{x}}, \mathbf{x})$. Most generative pre-training methods are based on two models: *Autoregressive* Model, which generates future inputs given only past inputs, and *Autoencoding* Model, which generates full inputs given partial inputs.

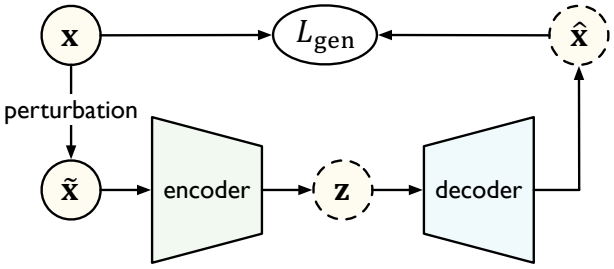


Figure 8: Generative pre-training tries to reconstruct the original input $\mathbf{x}$ from a perturbed input $\tilde{\mathbf{x}}$. The generative learning task shall encourage the learned representation $\mathbf{z}$ to capture the intrinsic and transferable explanatory factors from the data.

Autoregressive Model approximates the distribution of a sequence by predicting each entry conditioned on its previous context, which is called Language Modeling (LM) task in NLP. As shown in Figure 9, given a text sequence $\mathbf{x}_{1:T} = [x_1, x_2, ..., x_T]$, the learning objective of LM is to maximize the conditional probability of each entry x_t,

$$\max_\theta \sum_{t=1}^{T} \log P_\theta(x_t | x_{t-k}, \cdots, x_{t-1}), \tag{5}$$

where k is the size of the context window and θ is the parameter of the neural network. Generative Pre-Training (GPT) (Radford et al., 2018) explores unsupervised pre-training of Transformer with LM on the BooksCorpus (Zhu et al., 2015) dataset with over 7000

unpublished books. This equips the model with great transferability to various NLP tasks, such as question answering, commonsense reasoning, and so on. The advantage of LM is that it models the context dependency while the drawback is that it only encodes contextual information from one direction, yet contextual representations encoded in both directions may be more suitable to many downstream tasks, such as natural language inference.

Autoencoding Model approximates the data distribution by generating original data from encoded representations. Vincent et al. (2008) hypothesize that a good representation should also be robust to partial corruption of the input. Thus Denoising Autoencoder (Vincent et al., 2008) is trained to reconstruct the original input $\mathbf{x}$ with the corrupted input $\tilde{\mathbf{x}}$. Inspired from Denoising Autoencoder, BERT (Devlin et al., 2019) adopts the Masked Language Modeling (MLM) task as a pre-training task to overcome the drawback of the unidirectional LM. As shown in Figure 9, MLM first randomly masks out some tokens $m(\mathbf{x})$ from the input sentences $\mathbf{x}$ with a special `[MASK]` token and then trains the models to predict the masked tokens by the rest of the tokens $\mathbf{x}_{\setminus m(\mathbf{x})}$,

$$\max_{\theta} \sum_{x \in m(\mathbf{x})} \log P_\theta(x | \mathbf{x}_{\setminus m(\mathbf{x})}). \tag{6}$$

Masked pre-training has also been used in many other areas. For instance, Masked Autoencoders (MAE) (He et al., 2021) pre-trains vision transformers on large-scale unlabeled image datasets using the image generation task. The difficulty is that the signals are highly redundant in images, thus it is hard for generative tasks, such as filling a few missing pixels, to capture high-level knowledge from data. To tackle this issue, MAE randomly masks a very large portion of patches, forcing the model to go beyond low-level understanding and reconstruct the whole image based on a small subset of visible patches, which improve its transferability to semantic-level tasks. For another instance, to pre-train Graph Neural Network (GNN) (Garcia and Bruna, 2018) for transferable representations, Attribute Masking (Hu et al., 2020) conceals node or edge attributes and asks GNNs to predict those attributes based on neighboring structures, which can capture the regularities of attributes distribution over different graph structures, such as the chemistry rules in molecular graphs, and improve transferability on the downstream node or edge classification tasks.

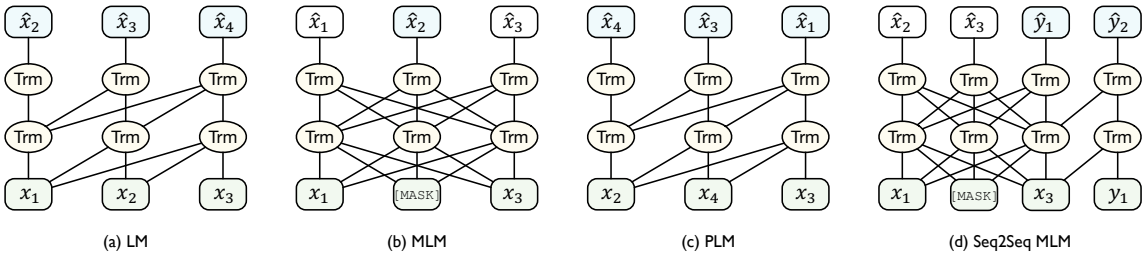


Figure 9: Attention visibility in Transformer (Trm) for language models. (a) LM maximizes the probabilities of all words conditioned on their previous words. (b) MLM maximizes the probabilities of random masked words conditioned on all unmasked words. (c) PLM permutes the original sequence and then performs autoregression. (d) Seq2Seq MLM encodes the input masked sequence x and then decodes the output masked tokens y sequentially.

Combining Autoregressive and Autoencoding Models. In MLM, some special tokens, such as `[MASK]`, are only used in pre-training while absent in the downstream tasks, leading to the mismatch between the pre-training phase and the fine-tuning phase. To mitigate this discrepancy, Permuted Language Modeling (PLM) (Yang et al., 2019) randomly samples a permutation of the sequence and then performs autoregression on the permuted sequence to predict the last few tokens. To explore the limits of transferability of knowledge gained in different generative pre-training methods, T5 (Raffel et al., 2020) unifies all text-based language tasks into the text-to-text format and then adopts a Sequence-to-Sequence MLM (Seq2Seq MLM), where the encoder processes a masked sequence and the decoder sequentially generates the masked tokens in an autoregression manner.

The design of unsupervised pre-training tasks has a great influence on the transferability to the downstream tasks, thus many efforts have been made to optimize the pre-training tasks and exploit better training objectives. RoBERTa (Liu et al., 2019b) explores the under-training issue of BERT and highlights that training with more data, longer sequences, and dynamically changed masking patterns helps the model transfer better. Besides, MLM randomly masks out some independent words, which are the smallest semantic units in English but may not have complete semantics in other languages, such as Chinese. Thus, ERNIE (Baidu) (Sun et al., 2019b) introduces entity-level and phrase-level masking, where multiple words that represent the same semantic meaning are masked. This achieves good transferability on Chinese NLP tasks. To improve transferability to tasks where span selection is important, such as question answering and coreference resolution, SpanBERT (Joshi et al., 2020) masks a random variable length of span in the text and trains the span boundary representations to predict the entire content of the masked span. BART (Lewis et al., 2020) introduces more perturbation functions such as sentence permutation, document rotation, token deletion, and text infilling for more transferable pre-trained models.

The generative pre-training on large-scale data greatly improves the transferability of models and even enables *few-shot* task transfer. By scaling up the model size to 175B and pre-training on the corpus over 500GB, GPT-3 (Brown et al., 2020) obtains impressive transferability. Using only task demonstrations and a few examples, GPT-3 achieves better performance than prior state-of-the-art fine-tuning approaches on some tasks. The success of GPT-3 comes from the fact that the web-scale corpus contains a vast amount of natural language sentences, which potentially demonstrate different tasks without explicit task symbols. A high-capacity language model trained on such data would perform unsupervised multi-task learning and absorb transferable knowledge to handle downstream tasks. The generative pre-training on large-scale data also improves the transferability across domains. Multilingual BERT (Pires et al., 2019) is pre-trained with MLM on Wikipedia texts from 104 languages and then achieves great cross-lingual transferability in the downstream tasks, where each language can be considered as a domain. Further, XLM (Lample and Conneau, 2019) introduces the translation language modeling task, which extends MLM to parallel bilingual sentence pairs, encouraging more transferable representations across language.

2.3.2 Contrastive Learning

Contrastive learning utilizes the idea of learning to compare for unsupervised pre-training. As shown in Figure 10, two different views, query $\mathbf{x}^q$ and key $\mathbf{x}^k$, are constructed from the

original data $\mathbf{x}$. Encoders will map different views into latent representations and decoders will further map the representation to the metric space. The model is learned by minimizing the distance between query and key of the same instance. We will review three typical contrastive learning methods widely used in pre-training: Mutual Information Maximization (uses the global context and the local features as different views), Relative Position Prediction (uses different local components as different views), and Instance Discrimination (uses data augmentations to generate different views of the same instance). Different ways of generating and comparing different views encourage these methods to respectively capture the *global-local* relation, *local-local* relation and *global-global* relation of the training data.

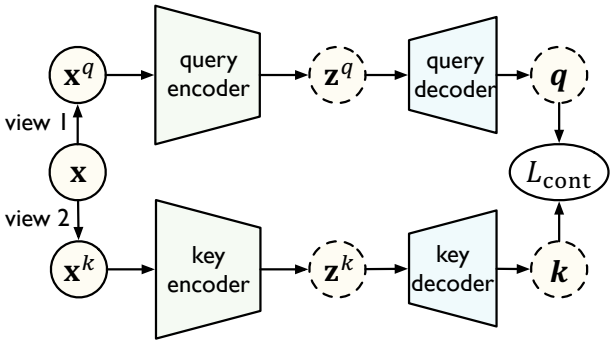


Figure 10: Contrastive pre-training aims to minimize the similarity between the query $\mathbf{q}$ and the key $\mathbf{k}$ that are generated from different views of the same data input $\mathbf{x}$.

Mutual Information Maximization. Deep InfoMax (Hjelm et al., 2019) aims to acquire transferable representations from the relation between the high-level global context and the low-level local features. Given input $\mathbf{x}$, Deep InfoMax learns an encoder ψ to maximize the mutual information between its input and output of the same instance. The mutual information can be estimated and bounded by training a discriminator to distinguish between their joint distribution and the product of their marginals. Using Noise-Contrastive Estimation (NCE), the training objective of Deep InfoMax becomes,

$$\max_{\psi} \mathbb{E}_{\mathbf{x}\sim\mathcal{U}} \left[D(\mathbf{x}, \psi(\mathbf{x})) - \mathbb{E}_{\mathbf{x}'\sim\widetilde{\mathcal{U}}} \Big(\log \sum_{\mathbf{x}'} e^{D(\mathbf{x}', \psi(\mathbf{x}))} \Big) \right], \tag{7}$$

where $\mathbf{x}$ is the input sampled from the training distribution $\mathcal{U}$ of upstream task, $\mathbf{x}'$ is another input sampled from $\widetilde{\mathcal{U}} = \mathcal{U}$, and D is the discriminator to distinguish between the joint distribution and the product of marginals. A parallel work, Contrastive Predictive Coding (CPC) (Oord et al., 2019), also maximizes the mutual information between pairs of global representation and local representation. Given a sequence input, CPC processes it with an encoder and summarizes the results into a context by an autoregression model. Then it maximizes the mutual information between the summarized context and the hidden representation of the future observation in the sequence, which guides the learned representations to capture information for predicting future samples.

Mutual information maximization has been used to obtain pre-trained models on many data formats, such as Deep InfoMax on image data and CPC on sequence data. On graph data, Deep Graph Infomax (Veličković et al., 2019) maximizes the mutual information between a node's local representations and the k-hop neighborhoods' context representations.

On multimodal data, Contrastive Language-Image Pre-training (CLIP) (Radford et al., 2021) maximizes the mutual information between the image and the corresponding text in a multimodal embedding space. After training with a large-scale dataset of image-text pairs from the Internet, it enables the *zero-shot* transfer of the model to downstream tasks, competitive with the prior task-specific supervised models.

Relative Position Prediction. Next Sentence Prediction (NSP) (Devlin et al., 2019), which is first introduced in BERT, acquires transferable representations from the relation between *local parts*. Specifically, NSP uses a binary classifier to predict whether two sentences are coherent from the training corpus, aiming to enhance the transferability to tasks with multiple sentences, such as question answering and natural language inference. However, subsequent work questions the necessity of NSP tasks (Yang et al., 2019; Liu et al., 2019c) and Lan et al. (2020) conjecture that NSP only forces the model to learn topic prediction, rather than more difficult coherence prediction. Since inter-sentence coherence is important to many downstream tasks, ALBERT (Lan et al., 2020) introduces a sentence-order prediction task, where two consecutive segments from the same document are taken as positive examples, and the same segments with order swapped are taken as negative examples. Similar ideas are also explored in vision, where the pre-training task is to predict relative positions of two patches from an image (Doersch et al., 2015).

Instance Discrimination. InstDisc (Wu et al., 2018) aims to learn transferable representations from the relation between *instances*. Given n instances, an encoder ψ is trained to distinguish each instance from others, i.e., minimize the distance between the query $\mathbf{q}$ and key $\mathbf{k}_+$ from the same instance (also called positive samples) and maximize the distance between that of different instances (also called negative samples),

$$\min_{\psi} -\log \frac{\exp(\mathbf{q}\cdot\mathbf{k}_+/\tau)}{\sum_{j=0}^{K}\exp(\mathbf{q}\cdot\mathbf{k}_j/\tau)}, \tag{8}$$

where τ is a temperature hyper-parameter and the sum is over one positive and K negative samples. Note that the computation of the features of all samples and the non-parametric softmax is costly especially when the number of training instances n is extremely large. To tackle this issue, negative sampling is used to approximate the softmax, i.e., $K < n$.

The discriminability of representations to contrast one instance from another instance is closely related to the transferability on downstream tasks. Thus, many efforts have been made to increase the number and improve the quality of keys. As shown in Figure 11, InstDisc (Wu et al., 2018) uses a memory bank to store the latest updated representations for each key, which increases the number of negative samples, yet may result in less consistent representations. Momentum Contrast (MoCo) (He et al., 2020) maintains a dynamic queue of encoded features to enlarge the size of negative samples and encodes the keys with a momentum-updated encoder, which increases encoding consistency between different samples in the queue and improves the quality of keys. The way how the positive samples and negative samples are constructed is also important for transferability. Contrastive Multiview Coding (CMC) (Tian et al., 2020) takes multiple views, rather than multiple augmentations, of the same instance as positive samples and achieves better transferability. SimCLR (Chen et al., 2020) emphasizes that data augmentations play a crucial role in implicitly defining

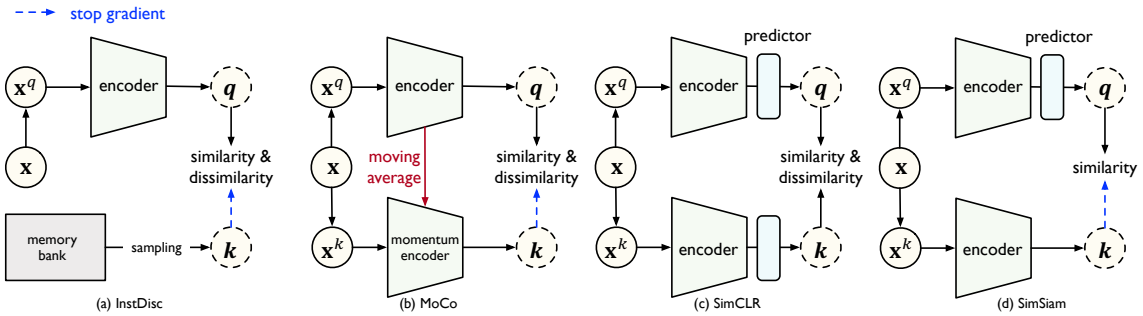


Figure 11: Comparison of different contrastive learning mechanisms. (a) InstDisc samples the keys from a memory bank. (b) MoCo encodes the new keys on the fly by a momentum encoder and maintains a queue of keys. (c) SimCLR encodes the keys and queries in the same batch with the same encoder and adds a nonlinear predictor to improve the representation. (d) SimSiam applies an MLP predictor on one side and applies a stop-gradient operation on the other side, and maximizes the similarity in two views without using negative pairs.

different pretext tasks, and the composition of stronger augmentations leads to better transferability even without the need for a memory bank or a queue. The introduction of negative samples is to avoid trivial solutions that all outputs collapse to a constant. However, BYOL (Grill et al., 2020) finds that when maximizing the similarity between two augmentations of one image, negative sample pairs are not necessary. Further, SimSiam (Chen and He, 2021) finds that momentum encoder is also not necessary while a stop-gradient operation applied on one side is enough for learning transferable representations.

Compared with supervised pre-training, contrastive pre-training leads to competitive performance on downstream classification tasks and even better performance on various other downstream tasks, such as object detection and semantic segmentation. To explain the stronger transferability of contrastive pre-training, Zhao et al. (2021) observe that standard supervised pre-training usually transfers high-level semantic knowledge, while contrastive pre-training usually transfers low-level and mid-level representations. When the target tasks are different from the supervised pre-trained tasks, the supervised pre-training methods have the risk of over-fitting the semantic discriminative parts of objects defined by the class labels, which hurts the transferability. On the contrary, the contrastive pre-training tasks lead to more holistic modeling of the objects, which relaxes the task misalignment issue and achieves better transferability for widespread downstream tasks.

2.4 Remarks

While standard supervised pre-training is well established, its transferability also depends on the relationship between the pre-training task and the target task, and no pre-training task can dominate all downstream tasks. He et al. (2019) show that compared with the random initialization, supervised pre-training on ImageNet only speeds up the convergence of object detection on the COCO dataset, but does not lead to better final accuracy. Raghu et al. (2019) observe similar phenomena in medical imaging, where training lightweight models from scratch perform comparably with transferring from ImageNet pre-trained models. Abnar et al. (2022) explore the limits of large-scale supervised pre-training and find that as the pre-training accuracy increases by scaling up data, model size and training time,

the performance of downstream tasks gradually saturates and there are even some extreme scenarios where performance on pre-training and downstream tasks are at odds with each other. These controversial results encourage us to *rethink* the common practice of supervised pre-training and design new supervised pre-training strategies for specific fields, especially when large gaps exist between the pre-training and target tasks.

Table 2: Comparison between different pre-training methods.

Method	Modality Scalability[1]	Task Scalability[2]	Data Efficiency[3]	Labeling Cost[4]
Standard Pre-Training	★★★	★★	★★★	★
Meta-Learning	★★★	★	★	★
Causal Learning	★★	★	★	★
Generative Learning	★★	★★★	★★★	★★★
Contrastive Learning	★	★★★	★★★	★★★

[1] Modality Scalability: whether models can be pre-trained on various modalities, such as text, graph.
[2] Task Scalability: whether pre-trained models can be easily transferred to different downstream tasks.
[3] Data Efficiency: whether stronger transferability can be yielded from large-scale pre-training.
[4] Labeling Cost: whether relies on manual data labeling.

Table 2 compares pre-training methods from four perspectives: modality scalability, task scalability, data efficiency, and labeling cost. Though meta-learning enables fast adaptation to new tasks, it mainly considers related tasks such as reinforcement learning under environments with small changing factors, while standard pre-training can transfer to broader task gaps such as from image classification to object detection. Besides, the existing meta-learning and causal learning methods are empirically verified only on small datasets, and it remains unclear whether they can acquire stronger transferability via pre-training on large-scale data. Despite the promising performance without manual labeling, unsupervised pre-trained models require a large number of gradient steps for fine-tuning to downstream tasks. Also, strong data augmentations are required by contrastive learning to gain transferability, but they are not easy to design in other modalities, such as text and graphs. Finally, the design of unsupervised pre-training tasks remains heuristic, lacking solid analysis on how the task shift is bridged and what enables the transferability of these models.

Acquiring transferability only through the pre-training stage may limit our horizon. As the shift in tasks and domains naturally exists between the pre-training and adaptation stages, many pre-training methods are tailored to adaptation. Unsupervised pre-training aims to improve the transferability to downstream tasks by exploring different kinds of self-supervised tasks to reduce the task discrepancy between pre-training and adaptation, or by enlarging the size and diversity of the upstream data to reduce the upstream-downstream discrepancy. The distribution shift commonly tackled by domain adaptation (Ganin and Lempitsky, 2015) also influences the transferability of the pre-trained model. For instance, the data distribution in a specific domain, such as biological and scientific literature, is quite different from that in the general pre-training domain and may degrade transferability, thus BioBert (Lee et al., 2020b) and SciBERT (Beltagy et al., 2019) perform pre-training on domain-specific data to improve the transferability on domain-specific tasks.

3. Adaptation

While pre-training on large-scale datasets can gain transferable knowledge in deep models, performing task adaptation with the target data is still necessary for most applications, as the target task is usually different from the pre-training task. When the labeled data for the target task is not enough, domain adaptation from a related source domain with labeled data to boost the performance on the target domain is also necessary in many applications. We will review task adaptation and domain adaptation in Sections 3.1 and 3.2 respectively.

3.1 Task Adaptation

In task adaptation, there exist a pre-trained model h_{θ^0} and a target domain $\widehat{\mathcal{T}} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$ of m labeled samples. The goal is to find a hypothesis $h_\theta : \mathcal{X} \mapsto \mathcal{Y}$ in the space $\mathcal{H}$ using the pre-trained model and target data to achieve a low generalization risk $\epsilon_{\mathcal{T}}(h_\theta)$. In general, there are two simple ways to *adapt* a pre-trained model to the downstream tasks: feature transfer and fine-tuning. Feature transfer freezes the weights of the pre-trained models and trains a linear classifier on top of that. In contrast, fine-tuning uses the pre-trained models to initialize the target model parameters and update these parameters during training. Feature transfer is fast in training and efficient in parameter storage, yet fine-tuning yields better performance (Yosinski et al., 2014), and has become a common practice for task adaptation in both vision and NLP (Girshick et al., 2014; Devlin et al., 2019).

Vanilla fine-tuning, which tunes the pre-trained models by empirical risk minimization on the target data, has been widely used in various downstream tasks and scenarios. However, vanilla fine-tuning still suffers from several issues, including *catastrophic forgetting* and *negative transfer*. We will introduce how to alleviate these issues in Sections 3.1.1 and 3.1.2. Besides, as the parameters of deep models keep increasing and some of them reach billions or trillions, parameter efficiency and data efficiency have become increasingly important in task adaptation. We will give an introduction on how to explore the transferability in the pre-trained models to solve these problems in Sections 3.1.3 and 3.1.4. Overall, Figure 12 shows the progress made by the task adaptation algorithms to solve different problems.

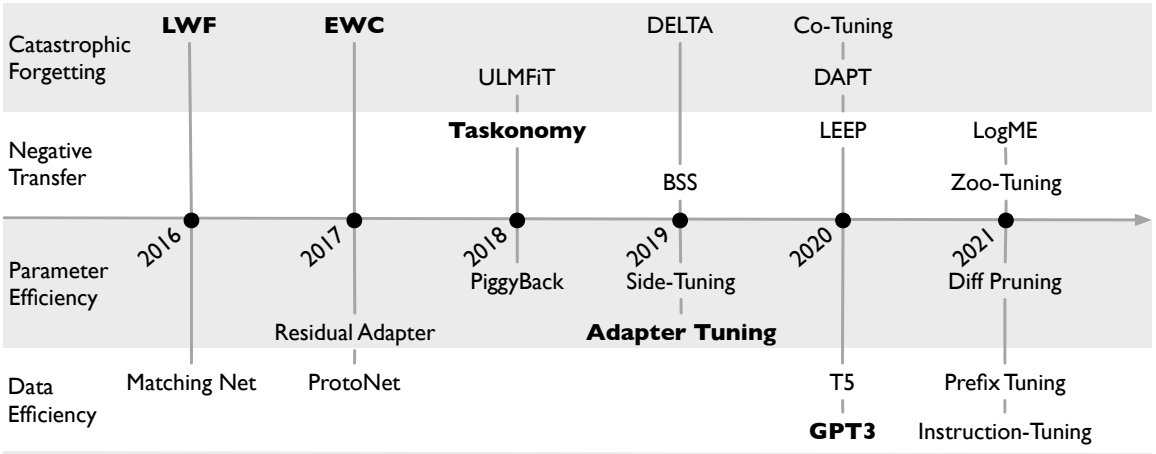


Figure 12: Cornerstones of task adaptation methods for *applying* transferable knowledge.

3.1.1 Catastrophic Forgetting

Catastrophic forgetting, which was first studied in *lifelong learning*, refers to the tendency of neural networks to lose knowledge acquired from previous tasks when learning new tasks (Kirkpatrick et al., 2017). In the fine-tuning scenario where labeled data is usually scarce, it will lead to the overfitting of models on the target data. This phenomenon is also called *representational collapse*, i.e., the degradation of generalizable representations during the fine-tuning stages (Aghajanyan et al., 2021). The most simple way to avoid catastrophic forgetting might be selecting a small learning rate and adopting an early-stopping strategy, which avoids updating the parameters too much. However, this strategy may lead the model to falling into the local minimal, especially when there is a large gap between the pre-training parameters and the optimal parameters for the downstream task.

Yosinski et al. (2014) find that the transferability of different layers is not the same — the first layers learn general features, the middle layers learn semantic features and the last layers learn task-specific features. Thus, to make the model retain the knowledge acquired in the pre-training task and fit the target task well at the same time, different layers should not be treated the same. Specifically, the first layers should retain more pre-trained knowledge while the last layers should adapt more to the downstream tasks. Inspired by this finding, DAN (Long et al., 2015) sets the learning rate of the task-specific head to be 10 times larger than that of the lower layers, which is simple yet effective when the labeled data is scarce or the target domain is close with the pre-training domain. ULMFiT (Howard and Ruder, 2018) gradually unfreezes the model starting from the last layers to the first layers, which effectively retains general knowledge in the first layers. To automatically determine which layers should be fine-tuned or frozen for each sample, Spottune (Guo et al., 2019) proposes a policy network that is deployed to output the routing decision based on the input of each sample and is jointly trained with the main model during fine-tuning.

Domain Adaptive Tuning reveals that an important source of catastrophic forgetting is the dataset shift between the pre-training and the target domain. To bridge such shift, ULMFiT (Howard and Ruder, 2018) and DAPT (Gururangan et al., 2020) first tune the pre-trained model on data related to the target domain, or simply data of the target task, with the pre-training task. Then they fine-tune the adaptive-tuned model on the target task (Figure 13). Usually, the pre-training task is unsupervised, thus further pre-training with in-domain data can provide rich information about the target data distribution for better task adaptation with no additional labeling costs. The two stages, domain adaptive tuning and regular fine-tuning, in the above methods can also be done jointly via multi-task learning. SiATL (Chronopoulou et al., 2019) adds an auxiliary language model loss to the task-specific optimization function, which alleviates catastrophic forgetting and learns task-specific features at the same time.

Regularization Tuning is another way to prevent the models from deviating far away from the pretrained ones. The optimization objective with a general regularization is

$$\min_{\theta} \sum_{i=1}^{m} L(h_\theta(\mathbf{x}_i), \mathbf{y}_i) + \lambda \cdot \Omega(\theta), \tag{9}$$

where L is the loss function, Ω is a general form of regularization, and λ is the trade-off between them. A typical regularization in supervised learning is L_2 penalty, $\Omega(\theta) =$

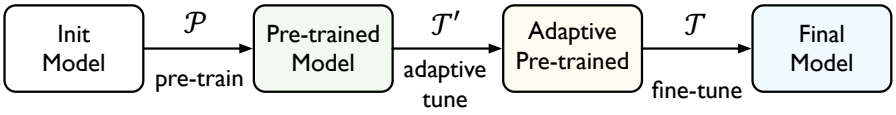


Figure 13: Domain Adaptive Tuning often consists of two consecutive steps: first, adaptive-tune on an auxiliary domain $\mathcal{T}'$ that is related to the target domain using the pre-training task; second, fine-tune on the target domain $\mathcal{T}$ using the target learning task.

$\frac{1}{2}||\theta||_2^2$, which drives the weights θ to zero to control the model complexity. Different from typical supervised learning, in fine-tuning, there exists a pre-trained model h_{θ^0} setting a reference that can be used to define the hypothesis space (Figure 4). Thus, Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) constrains the distance between the weights of the pre-trained and fine-tuned networks (Figure 14) to overcome catastrophic forgetting,

$$\Omega(\theta) = \sum_j \frac{1}{2} F_j \left\| \theta_j - \theta_j^0 \right\|_2^2, \tag{10}$$

where F is the estimated Fisher information matrix. EWC is based on the assumption that the networks with similar weights should produce similar outputs. However, due to the complex structures of deep networks, similar parameters do not necessarily produce the same output, and the same output may also come from completely different model parameters. Thus, DELTA (Li et al., 2019) constraints the behavior, i.e., the feature maps of the model by selecting the discriminative features with a supervised attention mechanism and regularizing the distance of these features between pre-trained and fine-tuned networks. Learning Without Forgetting (LWF) (Li and Hoiem, 2018) constrains the output prediction of the model by encouraging the model's response for old tasks to keep the same throughout the fine-tuning process (Figure 14). Regularization on the output often performs better than regularization on the parameters or the features, yet the latter two have better scalability and versatility to more complex downstream tasks.

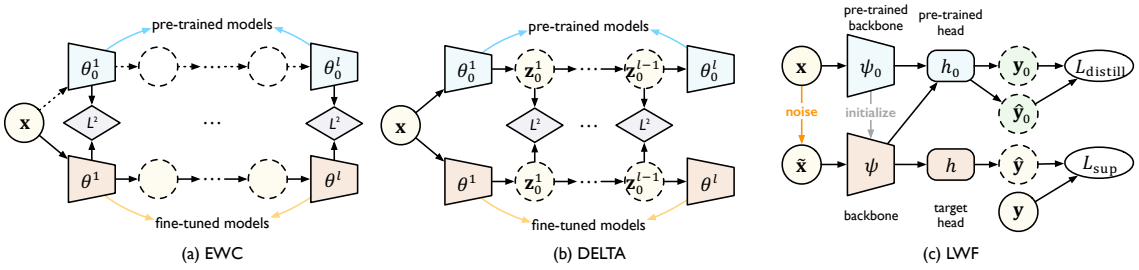


Figure 14: Regularization methods for task adaptation that avoids catastrophic forgetting. Blue: pre-trained parameters; Red: fine-tuned parameters. (a) EWC regularizes the *parameters* of the new models θ and that of the pre-trained models θ_0 with weighted L_2-penalty. (b) DELTA regularizes the *feature maps* of the new models $\mathbf{z}$ and that of the pre-trained models $\mathbf{z}_0$. (c) LWF enforces the *output* of the old tasks $\widehat{\mathbf{y}}_0$ close to the initial response $\mathbf{y}_0$.

An explanation for the effect of regularization is that it makes the hypothesis smoother. Therefore, TRADES (Zhang et al., 2019a) and SMART (Jiang et al., 2020) directly enforce the smoothness of the hypothesis by encouraging the output of the model to not change much when injecting a small perturbation to the input,

$$\Omega(\theta) = \sum_{i=1}^{m} \max_{||\widetilde{\mathbf{x}}_i - \mathbf{x}_i||_p \leq \epsilon} L_s(h_\theta(\widetilde{\mathbf{x}}_i), h_\theta(\mathbf{x}_i)), \tag{11}$$

where $\epsilon > 0$ is a small positive, L_s is the distance between two predictions, such as the symmetrized KL-divergence in classification and the squared loss in regression.

Apart from the training objective, an alternative regularization approach is through the parameter updating strategy. Stochastic Normalization (Kou et al., 2020) randomly replaces the target statistics in the batch-normalization layer (Ioffe and Szegedy, 2015) with their pre-trained statistics, which serves as an implicit regularization by avoiding over-depending on the target statistics. Mixout (Lee et al., 2020a) randomly replaces part of the model parameters with their pre-trained weights during fine-tuning to mitigate catastrophic forgetting. Child-Tuning (Xu et al., 2021) selects a subset of parameters (child network) by some criterion and only updates them during fine-tuning. In some senses, the above methods decrease the hypothesis space to preserve the transferability in pre-trained models.

3.1.2 Negative Transfer

Although the paradigm of pre-training and fine-tuning has been used in various downstream tasks, it does not necessarily produce a positive effect, which is known as *negative transfer* (Rosenstein, 2005). Wang et al. (2019d) propose to quantitatively measure the degree of negative transfer across different domains and we extend this idea to the paradigm of pre-training and fine-tuning.

Definition 2 (Negative Transfer Gap) *Let $h_\theta(\mathcal{U}, \mathcal{T})$ be a hypothesis obtained by adapting the model pre-trained from the upstream data $\mathcal{U}$ to the target data $\mathcal{T}$, and $h_\theta(\emptyset, \mathcal{T})$ be a hypothesis obtained by training from scratch on $\mathcal{T}$, then negative transfer gap is defined as*

$$NTG = \epsilon_{\mathcal{T}}(h_\theta(\mathcal{U}, \mathcal{T})) - \epsilon_{\mathcal{T}}(h_\theta(\emptyset, \mathcal{T})), \tag{12}$$

and negative transfer occurs if NTG is positive and vice versa.

First, negative transfer will happen when the relatedness between the upstream task and downstream task is not strong, e.g. Next Sentence Prediction pre-training task will hurt token-level classification tasks (Liu et al., 2019b). Negative transfer will also happen when there is a large shift between the pre-training domain and the target domain, e.g. for legal documents classification, pre-training only on legal documents is better than pre-training on more diverse datasets (Zheng et al., 2021). Second, negative transfer depends on the size of the labeled target dataset (Wang et al., 2019d). For example, He et al. (2019) empirically show that on large-scale object detection datasets (e.g. COCO), ImageNet pre-training is not beneficial when training for enough iterations. Third, negative transfer depends on the task adaptation algorithms. An ideal adaptation algorithm should promote positive transfer between related tasks while avoiding negative transfer between unrelated tasks. In practice, however, these two goals are often contradictory and result in the *dilemma*: approaches that promote larger positive transfer will suffer from severer negative transfer (Figure 15).

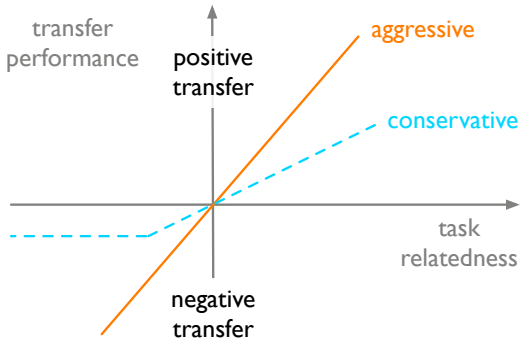


Figure 15: Dilemma to promote positive transfer and avoid negative transfer: Aggressive strategies that promote larger positive transfer will suffer from severer negative transfer; Conservative strategies can decrease negative transfer, yet lead to smaller positive transfer.

Enhancing Safe Transfer. One way to avoid negative transfer is to recognize and reject harmful knowledge in the pre-trained model. Chen et al. (2019b) observe that with sufficient training data, the spectral components with small singular values vanish during fine-tuning, indicating that small singular values correspond to detrimental pre-trained knowledge and may cause negative transfer. Thus BSS penalizes smaller singular values to suppress untransferable spectral components for safe transfer. Jang et al. (2019) meta-learns the weights determining which pairs of layers should be matched and to what extent the knowledge should be transferred, which rejects irrelevant information during transfer. Zoo-tuning (Shu et al., 2021b) enables adaptive transfer from a zoo of models, which adaptively aggregates multiple pre-trained models to derive the target model using data-dependent gating mechanisms to highlight transferable parameters. Another way to mitigate negative transfer of the pre-trained model is to fully explore the target data. Self-Tuning (Wang et al., 2021) proposes a pseudo group contrastive mechanism to explore the intrinsic structure of the target data in the process of fine-tuning with standard supervised objective.

Choosing Pre-trained Models. With the fast development of deep learning, a large zoo of pre-trained models are available, thus a simpler way to avoid negative transfer is to select a model that is pre-trained on the upstream data/task relevant to the downstream data/task. The most common practice to choose pre-trained models is based on rich past experience or through heavy experiments. To facilitate faster selection, Taskonomy (Zamir et al., 2018) proposes a fully computational approach for explicitly modeling the relationship between 26 different visual tasks. Another more efficient strategy to select pre-trained models is to predict the transferability of pre-trained models. LEEP (Nguyen et al., 2020) constructs an empirical predictor by estimating the joint distribution over pre-trained labels and the target labels, and then uses the log expectation of the empirical predictor to measure the transferability. LogME (You et al., 2021) proposes to predict the fine-tuning performance from the compatibility of features $\{\mathbf{z}_i = \psi(\mathbf{x}_i)\}_{i=1}^m$ and labels $\{\mathbf{y}_i\}_{i=1}^m$. Still, these methods may underestimate strong but non-linear features. He et al. (2021) show that features from contrastive pre-training, such as MoCo v3 (Chen et al., 2021a), have higher linear probing accuracy while worse fully fine-tuning results than generative pre-training, such as MAE (He et al., 2021), indicating that the linear separability of the pre-trained features is not the sole metric for evaluating transferability.

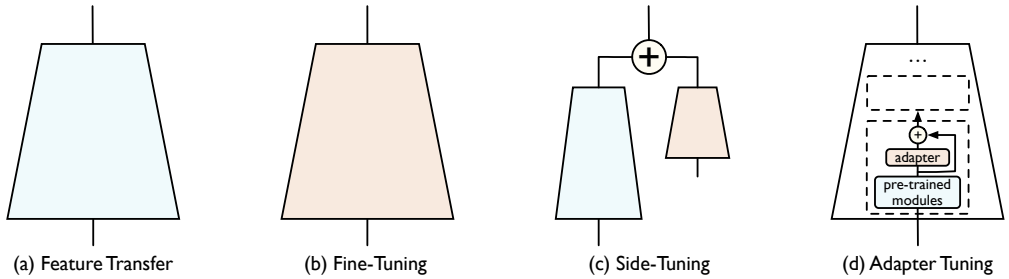


Figure 16: Comparison on how different adaptation methods freeze (blue) and tune (red) pre-trained parameters. (a) Feature transfer freezes all the pre-trained parameters. (b) Fine-tuning re-trains all the pre-trained parameters. (c) Side-tuning trains a lightweight conditioned side network that is fused with the fixed pre-trained network using summation. (d) Adapter-Tuning inserts adapter modules for tuning into each frozen pre-trained layer.

3.1.3 Parameter Efficiency

Fine-tuning large pre-trained models yields strong performances on many downstream tasks (Radford et al., 2018; Devlin et al., 2019), yet it is not parameter efficient since it generates a full set of model parameters for each downstream task, which will cause unacceptable storage cost as the number of tasks increases. The simplest solution is *Multi-task Learning* (Caruana, 1997), i.e., fine-tuning a single model to solve multiple target tasks, which might be mutually beneficial to each other (He et al., 2017; Liu et al., 2019a). Yet when different target tasks are weakly related, multi-task learning will underperform fine-tuning for each task separately. Also, multi-task learning requires simultaneous access to all target tasks, which is not feasible in online scenarios where the target tasks arrive in sequence. Hereafter, we will introduce new tuning paradigms proposed to improve parameter efficiency.

Residual Tuning. Inspired by the fact that approximation to a difference is easier than the original function (He et al., 2016), Side-Tuning (Zhang et al., 2019b) adds a small side network h_{side} to adapt the frozen pre-trained model $h_{\text{pretrained}}$ for the target task and obtain a combined model $h(x) = \alpha h_{\text{pretrained}}(x) + (1-\alpha)h_{\text{side}}(x)$, where α is a weight that changes during training. When there is a big gap between the pre-trained model and the downstream task, it may be difficult to learn the residuals of the entire model. Thus, Adapter Tuning (Houlsby et al., 2019) inserts residual adapter modules into each frozen layer. Residual Adapter was first introduced for learning multiple visual domains (Rebuffi et al., 2017) and consists of a skip connection, such that it is set as a near-identity function and will not impair the whole model when the training starts. By choosing a much smaller amount of parameters for the adapters, Adapter Tuning can extend pre-trained models to new tasks without increasing much storage cost. Houlsby et al. (2019) find that Adapter Tuning with only 3.6% tunable parameters can match the performance of the fully fine-tuned BERT on the GLUE benchmark (Wang et al., 2019a), revealing the great potential of this method.

Parameter Difference Tuning. While residual adapter tuning changes the model activations by adding new modules, parameter difference tuning extends the pre-trained models through a task-specific parameter difference vector,

$$\theta_{\text{task}} = \theta_{\text{pretrained}} \oplus \delta_{\text{task}}, \tag{13}$$

where $\oplus$ is the element-wise addition function, $\theta_{\text{pretrained}}$ is the fixed pre-trained parameters and δ_{task} is the tuned task-specific difference vector. Instead of storing a copy of θ_{task} for every task, difference tuning only needs to store a single copy of $\theta_{\text{pretrained}}$ and a copy of δ_{task} for every task. As long as the size of δ_{task} can be reduced, we can achieve parameter efficient models. To this end, Diff Pruning (Guo et al., 2021) utilizes L_0-norm penalty (Louizos et al., 2018) to encourage sparsity of the difference vector δ_{task}. Aghajanyan et al. (2021) adopt FastFood transform M (Li et al., 2018) to convert δ_{task} into a low-dimensional vector δ_{low}, i.e., $\delta_{\text{task}} = \delta_{\text{low}} M$. The element-wise addition can also be replaced by element-wise multiplication. For instance, Piggyback (Mallya and Lazebnik, 2018) multiplies real-valued mask weights to the pre-trained parameters, i.e., $\theta_{\text{task}} = \theta_{\text{pretrained}} \odot \delta_{\text{task}}$ during training. After training, the mask weights δ_{task} are passed through a thresholding function to obtain binary-valued masks, further reducing the parameter storage of δ_{task} at inference.

The essential difference between the above two tuning methods lies in their different assumptions about the root of transferability. Residual tuning assumes that transferability is encoded in the *behaviors* of each module, i.e., the features output by each module. When adapting to the downstream tasks, we only need to add some task-specific behaviors by stacking the pre-trained modules with the residual adapter modules. In contrast, parameter difference tuning assumes that transferability lies in the pre-trained *parameters*. Most of the pre-trained parameters can be reused, and only a small part of them need to be adapted to the downstream tasks, thus we only need to store the increment. Another thing to mention is that when limiting the size of the residual adapters or the complexity of the difference vector, these methods naturally overcome the catastrophic forgetting issue in Section 3.1.1.

3.1.4 Data Efficiency

Currently, when fine-tuning large pre-trained models, hundreds or even thousands of labeled samples are still required to achieve strong performance on a specific downstream task, which limits the application of the "pre-training and fine-tuning" paradigm to wider range of tasks where labeled data are expensive to collect. In contrast, people can adapt to a new task with extremely few labeled samples, which is known as *few-shot learning*, or even with no labeled samples, which is known as *zero-shot learning*. Considering the lifecycle of deep learning, we can tackle this problem in three ways. The first is to improve the cross-task transferability of the pre-trained models, such as by increasing the model capacity or the pre-training dataset size, which is mentioned in Section 2. The second is to transfer from another labeled source domain where labeled data is cheaper to collect, which will be discussed in Section 3.2. The last is to reformulate the target task to close its gap with the pre-trained models, which is the focus of this part.

Metric Learning. Fine-tuning in low data regimes will easily cause over-fitting as updating the model of large-scale parameters using few labeled samples is ill-posed. In contrast, many *non-parametric* methods, such as nearest neighbors, can deal with low-sample regimes without suffering from catastrophic forgetting. To combine the advantages of parametric and non-parametric methods, Matching Net (Vinyals et al., 2016) uses an attention mechanism over the learned representations to predict the classes for the query samples, which can be interpreted as weighted nearest neighbors. Since labeled data is severely limited, ProtoNet (Snell et al., 2017) adds a stronger inductive bias that there exists a single pro-

totype representation for each class, where each prototype is the mean of the features of the labeled samples in each class, and classification boils down to finding the nearest prototype. Since no gradient update is performed on the feature representation, choosing a proper distance metric that has good transferability across tasks plays an important role. A common choice is the cosine distance, which explicitly reduces the intra-class variations and improves the cross-task transferability. Chen et al. (2019a) find that by replacing the linear classifier with a cosine-distance based classifier, the naive feature transfer method without fine-tuning serves as a strong baseline in few-shot learning.

Prompt Learning. Prompting, firstly proposed in GPT-3 (Brown et al., 2020), is another way to reformulate the downstream task to make it similar to the solved pre-training task. In fine-tuning, models will take input $\mathbf{x}$ and predict an output $\mathbf{y}$ as $P(\mathbf{y}|\mathbf{x})$. In prompting, the original input $\mathbf{x}$ is modified by a prompt template into a new string $\tilde{\mathbf{x}}$ that has unfilled slots, then the pre-trained language model will fill $\tilde{\mathbf{x}}$ to obatin a final string $\widehat{\mathbf{x}}$ and derive the output $\mathbf{y}$ from $\widehat{\mathbf{x}}$ (Liu et al., 2021b). Table 3 provides an example of prompting methods.

Table 3: An example of prompting method from Liu et al. (2021b).

Name	**Notation**	**Example**
Input	$\mathbf{x}$	I love this film.
Output	$\mathbf{y}$	`positive`
Prompt Template	$f_{\text{prompt}}(\mathbf{x})$	[X] Overall, it was a [Z] film.
Prompt	$\tilde{\mathbf{x}}$	I love this film. Overall, it was a [Z] film.
Filled Prompt	$\widehat{\mathbf{x}}$	I love this movie. Overall, it was a good movie.

The advantage of prompting is that it enables few-shot or even zero-shot task adaptation. The strong cross-task transferability stems from the implicit multiple tasks such as question-answering that language models are forced to learn on the large-scale pre-training corpus. However, this transferability requires a large model capacity to deal with potential implicit tasks and is also very sensitive to the choice of prompts. Thus, the disadvantage is that it introduces the necessity for prompt engineering, i.e., finding the best prompts to solve each downstream task, which is work-heavy and time-consuming especially on large datasets.

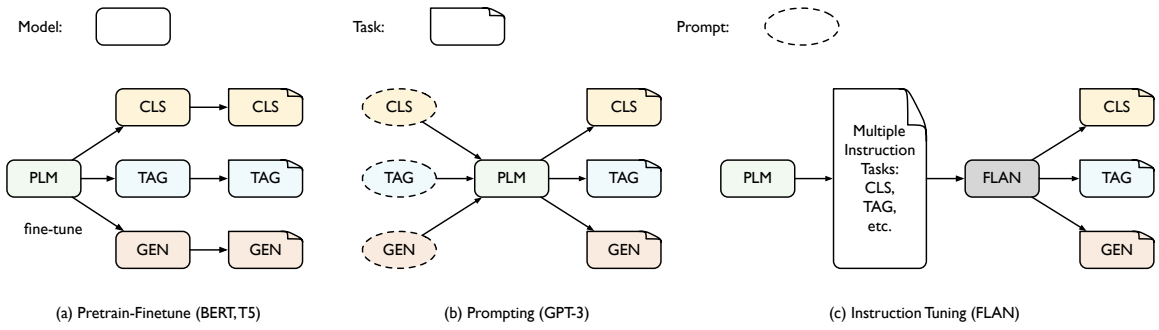


Figure 17: (a) Fine-tuning generates a new set of parameters for each downstream task. (b) Prompting fixes the pre-trained parameters and finds task-specific prompts to solve each downstream task. (c) Instruction Tuning tunes the pre-trained models on instruction-format dataset and uses the obtained model to do inference with multiple downstream tasks.

Combining prompting and fine-tuning may tackle this problem (Figure 17). PET-TC (Schick and Schütze, 2020) tunes the parameters of pre-trained language models in prompt learning while Prefix-Tuning (Li and Liang, 2021) adds additional prompt-related parameters and tunes these parameters. Instruction Tuning (Wei et al., 2022) explicitly fine-tunes the pre-trained models on a mixture of datasets expressed through natural language instructions (similar to the filled prompt) and obtain Fine-tuned LAnguage Model (FLAN), which largely increases the models' transferability to unseen tasks. In summary, prompt learning has provided a revolutionary way on how to utilize the transferability of pre-trained models.

3.1.5 Remarks

Table 4 gives a comparison of different task adaptation methods. Fine-tuning (including vanilla fine-tuning, domain adaptive tuning, and regularization tuning) has better performance when there are enough labeled data in the downstream tasks. In contrast, prompt learning requires much fewer labeled data to achieve decent performance, yet its applications are still limited to NLP and it is still non-trivial to extend it to vision or other areas. Fine-tuning is the most parameter-inefficient since it generates a full set of model parameters for each downstream task, while residual tuning, difference tuning, and prompt learning are all parameter efficient. Also, these latter methods naturally mitigate the catastrophic forgetting problem, but *negative transfer* is still a hard problem to be resolved.

Table 4: Comparison between different task adaptation methods.

	Adaptation Performance[1]	Data Efficiency[2]	Parameter Efficiency[3]	Modality Scalability[4]	Task Scalability[5]
Feature Transfer	★	★★	★★★	★★★	★★★
Vanilla Fine-tuning	★★★	★	★	★★★	★★★
Domain Adaptive Tuning	★★★	★★	★	★★	★★★
Regularization Tuning	★★★	★★	★	★★★	★
Residual Tuning	★★	★★	★★	★★	★★
Parameter Difference Tuning	★★	★★	★★	★★★	★★★
Metric Learning	★	★★★	★★★	★★★	★
Prompt Learning	★★	★★★	★★★	★	★

[1] Adaptation Performance: performance when there are large-scale labeled data in downstream tasks.
[2] Data Efficiency: performance when there are only small-scale labeled data in downstream tasks.
[3] Parameter Efficiency: whether can control parameters when the number of downstream tasks increases.
[4] Modality Scalability: whether can adapt pre-trained models to various modalities, such as text, graph.
[5] Task Scalability: whether can adapt pre-trained models to different downstream tasks, such as detection.

The motivation of many task adaptation methods can be understood from the perspective of *transferability*. For instance, domain adaptive tuning aims to bridge the domain discrepancy between the pre-training task and the downstream task by further obtaining a pre-trained model on the target data distribution. Prompt learning aims to bridge the task discrepancy between the pre-training task and the downstream task by reformulating all the tasks to the same format. In this scenario, when all tasks can be expressed in the same form, the difference between the pre-training task and the downstream task is only the shift in data distributions, i.e., task adaptation becomes the domain adaptation problem.

3.2 Domain Adaptation

The pre-training and fine-tuning paradigm has greatly improved the state-of-the-arts for diverse machine learning problems and applications, and the pre-trained deep networks can be easily adapted to the tasks at hand even with a small amount of labeled data. However, in many practical scenarios, there is no labeled training data and thus there is the demand to transfer a deep network from a source domain where labeled training data is available to a target domain where only unlabeled data exists (Chen et al., 2012; Glorot et al., 2011). In this situation, the deep models still suffer from performance degradations due to *distribution shift* (Quionero-Candela et al., 2009). Thus, domain adaptation is proposed to reduce the distribution shift between training and testing domains (Pan and Yang, 2010).

Many methods have been proposed for domain adaptation in the shallow regime, either by re-weighting or selecting samples from the source domain (Sugiyama et al., 2008) or seeking an explicit feature space transformation from the source distribution into the target distribution (Gong et al., 2013). As seminal methods, Huang et al. (2007); Pan et al. (2011); Long et al. (2013) explicitly match the distributions in the kernel-reproducing Hilbert space, while Gong et al. (2012) map the principal axes associated with each of the distributions. This survey will focus on deep domain adaptation, where adaptation modules are embedded in deep architectures to match data distributions across domains.

In unsupervised domain adaptation (UDA), there is a source domain $\widehat{\mathcal{S}} = \{(\mathbf{x}_i^s, \mathbf{y}_i^s)\}_{i=1}^n$ of n labeled samples and a target domain $\widehat{\mathcal{T}} = \{\mathbf{x}_i^t\}_{i=1}^m$ of m unlabeled samples. The goal of a learning algorithm is to find a hypothesis $h : \mathcal{X} \mapsto \mathcal{Y}$ in the hypothesis space $\mathcal{H}$ with a low target risk $\epsilon_{\mathcal{T}}(h) = \mathbb{E}_{(\mathbf{x}^t, \mathbf{y}^t) \sim \mathcal{T}}[\ell(h(\mathbf{x}^t), \mathbf{y}^t)]$ with no access to the labels of $\mathcal{T}$, where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a loss function. Several seminal theories have been proposed to tackle this problem and the the main idea of them is to bound the target risk $\epsilon_{\mathcal{T}}(h)$ by the source risk $\epsilon_{\mathcal{S}}(h)$ and a distribution distance. In this survey, we will focus on the theory of $\mathcal{H}\Delta\mathcal{H}$-Divergence (Ben-David et al., 2006, 2010a; Mansour et al., 2009) and Disparity Discrepancy (Zhang et al., 2019c) and illustrate how to derive different algorithms from these theories. First, using triangle inequalities, we can relate the target risk to the source risk as follows.

Theorem 3 (Bound with Disparity) *Assume that the loss function ℓ is symmetric and obeys the triangle inequality. Define the* disparity *between any two hypotheses h and h' on distribution $\mathcal{D}$ as*

$$\epsilon_{\mathcal{D}}(h, h') = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}}[\ell(h(\mathbf{x}), h'(\mathbf{x}))]. \tag{14}$$

Then the target risk $\epsilon_{\mathcal{T}}(h)$ can be bounded by

$$\epsilon_{\mathcal{T}}(h) \leqslant \epsilon_{\mathcal{S}}(h) + [\epsilon_{\mathcal{S}}(h^*) + \epsilon_{\mathcal{T}}(h^*)] + |\epsilon_{\mathcal{S}}(h, h^*) - \epsilon_{\mathcal{T}}(h, h^*)|, \tag{15}$$

where $h^* = \arg\min_{h \in \mathcal{H}} [\epsilon_{\mathcal{S}}(h) + \epsilon_{\mathcal{T}}(h)]$ is the ideal joint hypothesis, $\epsilon_{ideal} = \epsilon_{\mathcal{S}}(h^*) + \epsilon_{\mathcal{T}}(h^*)$ is the ideal joint error, $|\epsilon_{\mathcal{S}}(h, h^*) - \epsilon_{\mathcal{T}}(h, h^*)|$ is the disparity difference *between $\mathcal{S}$ and $\mathcal{T}$.*

It is a common *assumption* in domain adaptation that the ideal joint error ϵ_{ideal} shall be sufficiently small, otherwise domain adaptation will be infeasible, the *impossibility* theorem (Ben-David et al., 2010b). The goal is reduced to bound the disparity difference. However, since the ideal hypothesis h^* is unknown due to the unavailability of labeled target data, the disparity difference cannot be estimated directly. To this end, $\mathcal{H}\Delta\mathcal{H}$-Divergence (Ben-David et al., 2006, 2010a) is proposed to measure the upper bound of the disparity difference.

Definition 4 ($\mathcal{H}\Delta\mathcal{H}$-Divergence) *Define* $\mathcal{H}\Delta\mathcal{H} \triangleq \{h|h = h_1 \otimes h_2, h_1, h_2 \in \mathcal{H}\}$ *as the* symmetric difference hypothesis space *of* $\mathcal{H}$*, where* $\otimes$ *stands for the XOR operator. Then the* $\mathcal{H}\Delta\mathcal{H}$-Divergence *between* $\mathcal{S}$ *and* $\mathcal{T}$ *is*

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) \triangleq \sup_{h,h'\in\mathcal{H}} \left|\epsilon_{\mathcal{S}}\left(h,h'\right) - \epsilon_{\mathcal{T}}\left(h,h'\right)\right|.$$

For binary classification problem with the 01*-loss,* $\ell(y,y') = \mathbb{1}(y \neq y')$*, we have*

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) = \sup_{\delta\in\mathcal{H}\Delta\mathcal{H}} \left|\mathbb{E}_{\mathcal{S}}\left[\delta(\mathbf{x}) \neq 0\right] - \mathbb{E}_{\mathcal{T}}\left[\delta\left(\mathbf{x}\right) \neq 0\right]\right|.$$

The main advantage of the $\mathcal{H}\Delta\mathcal{H}$-Divergence is that it can be estimated from *finite* unlabeled samples of source and target domains. However, it is generally hard to compute and optimize. Thus, it is approximated by training a domain discriminator D that separates the source and target samples (Ben-David et al., 2006; Ganin and Lempitsky, 2015). Assume that the family of the discriminators is rich enough, such as the multilayer perceptrons (MLP) that is universal approximator to any functions, to contain $\mathcal{H}\Delta\mathcal{H}$, i.e., $\mathcal{H}\Delta\mathcal{H} \subset \mathcal{H}_D$. The $\mathcal{H}\Delta\mathcal{H}$-Divergence can be further bounded by $\sup_{D\in\mathcal{H}_D} \left|\mathbb{E}_{\mathcal{S}}\left[D(\mathbf{x}) = 1\right] + \mathbb{E}_{\mathcal{T}}\left[D\left(\mathbf{x}\right) = 0\right]\right|$, which gives rise to the *domain adversarial* methods in Section 3.2.2. The $\mathcal{H}\Delta\mathcal{H}$-Divergence can also be estimated in a nonparametric way by replacing $\mathcal{H}\Delta\mathcal{H}$ with a proper function space $\mathcal{F}$, which induces the *statistics matching* methods in Section 3.2.1.

The following theorem is the earliest theory in domain adaptation, which establishes the generalization bound based on the $\mathcal{H}\Delta\mathcal{H}$-Divergence for binary classification problems.

Theorem 5 (Ben-David et al. (2010a)) *Let* $\mathcal{H}$ *be a binary hypothesis space of VC dimension* d*. If* $\widehat{\mathcal{S}}$ *and* $\widehat{\mathcal{T}}$ *are samples of size* m *each, then for any* $\delta \in (0,1)$*, with probability at least* $1-\delta$*, for every* $h \in \mathcal{H}$*,*

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h) + d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{S}},\widehat{\mathcal{T}}) + \epsilon_{ideal} + 4\sqrt{\frac{2d\log(2m) + \log(\frac{2}{\delta})}{m}}. \tag{16}$$

This bound sheds key insights into algorithm designs. However, it has the limit of being based on the particular 01-loss. Thus, Mansour et al. (2009) extend the domain adaptation theory to a general class of loss functions satisfying the symmetry and subadditivity.

Theorem 6 (Mansour et al. (2009)) *Assume that the loss function* ℓ *is symmetric and obeys the triangle inequality, and define* $h^*_{\mathcal{S}} = \arg\min_{h\in\mathcal{H}} \epsilon_{\mathcal{S}}(h)$ *and* $h^*_{\mathcal{T}} = \arg\min_{h\in\mathcal{H}} \epsilon_{\mathcal{T}}(h)$ *as the ideal hypotheses for the source and target domains, then for every* $h \in \mathcal{H}$*,*

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h, h^*_{\mathcal{S}}) + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) + \epsilon, \tag{17}$$

where $\epsilon_{\mathcal{S}}(h, h^*_{\mathcal{S}})$ *stands for the source risk and* $\epsilon = \epsilon_{\mathcal{T}}(h^*_{\mathcal{T}}) + \epsilon_{\mathcal{S}}(h^*_{\mathcal{T}}, h^*_{\mathcal{S}})$ *for the capacity to adapt. Further, let* ℓ *be bounded,* $\forall(y,y') \in \mathcal{Y}^2, \ell(y,y') \leq M$ *for some* $M > 0$*, and defined as* $\ell(y,y') = |y-y'|^q$ *for some* q*. If* $\widehat{\mathcal{S}}$ *and* $\widehat{\mathcal{T}}$ *are samples of size* n *and* m *each, with probability at least* $1-\delta$*, we have*

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) \leq d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{S}},\widehat{\mathcal{T}}) + 4q(\mathfrak{R}_{n,\mathcal{S}}(\mathcal{H}) + \mathfrak{R}_{m,\mathcal{T}}(\mathcal{H})) + 3M\left(\sqrt{\frac{\log\frac{4}{\delta}}{2n}} + \sqrt{\frac{\log\frac{4}{\delta}}{2m}}\right), \tag{18}$$

where $\mathfrak{R}_{n,\mathcal{D}}$ is the expected Rademacher Complexity (Bartlett and Mendelson, 2002) with respect to distribution $\mathcal{D}$ and sample size n.

Note that the $\mathcal{H}\Delta\mathcal{H}$-Divergence bounds are still *loose* since the supremum is taken over both $h' \in \mathcal{H}$ and $h \in \mathcal{H}$. Observing that h is known as the source classifier, the Disparity Discrepancy (Zhang et al., 2019c) provides a tighter bound by computing directly on $\mathcal{H}$.

Definition 7 (Disparity Discrepancy) *Given a binary hypothesis space $\mathcal{H}$ and a* specific hypothesis $h \in \mathcal{H}$*, the* Disparity Discrepancy *induced by $h' \in \mathcal{H}$ is defined by*

$$d_{h,\mathcal{H}}(\mathcal{S},\mathcal{T}) = \sup_{h' \in \mathcal{H}} \left(\mathbb{E}_{\mathcal{T}} \mathbb{1}[h' \neq h] - \mathbb{E}_{\mathcal{S}} \mathbb{1}[h' \neq h] \right) \tag{19}$$

Since the supremum is only take over $h' \in \mathcal{H}$, estimating and minimizing the disparity discrepancy jointly through a minimax game can be done much more easily. The disparity discrepancy can well measure the distribution shift and yields a tighter generalization bound.

Theorem 8 (Zhang et al. (2019c)) *Let $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{T}}$ be samples of size n and m each. For any $\delta > 0$ and every binary classifier $h \in \mathcal{H}$, with probability at least $1 - 3\delta$, we have*

$$\begin{aligned} \epsilon_{\mathcal{T}}(h) \leq \epsilon_{\widehat{\mathcal{S}}}(h) + d_{h,\mathcal{H}}(\widehat{\mathcal{S}},\widehat{\mathcal{T}}) + \epsilon_{ideal} + 2\mathfrak{R}_{n,\mathcal{S}}(\mathcal{H}) \\ + 2\mathfrak{R}_{n,\mathcal{S}}(\mathcal{H}\Delta\mathcal{H}) + 2\sqrt{\frac{\log\frac{2}{\delta}}{2n}} + 2\mathfrak{R}_{m,\mathcal{T}}(\mathcal{H}\Delta\mathcal{H}) + \sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \end{aligned} \tag{20}$$

The disparity discrepancy can be further extended to the *multiclass* classification problem with hypothesis space $\mathcal{F}$ of scoring functions $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and margin loss, which is going beyond existing bounds and closer to the choices for real tasks (Zhang et al., 2019c).

Definition 9 (Margin Disparity Discrepancy) *Given a scoring hypothesis space $\mathcal{F}$, denote the margin of a real hypothesis f at a labeled example (x, y) as $\rho_f(x, y) \triangleq \frac{1}{2}(f(x, y) - \max_{y' \neq y} f(x, y'))$, the labeling function induced by f as $h_f : x \mapsto \arg\max_{y \in \mathcal{Y}} f(x, y)$, and the margin loss as*

$$\Phi_\rho(x) \triangleq \begin{cases} 0 & \rho \leq x \\ 1 - x/\rho & 0 \leq x \leq \rho \\ 1 & x \leq 0 \end{cases}, \tag{21}$$

then the margin disparity *between f and f' on distribution $\mathcal{D}$ is*

$$\epsilon_{\mathcal{D}}^{(\rho)}(f', f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\Phi_\rho(\rho_{f'}(x, h_f(x))]. \tag{22}$$

Given a specific hypothesis $f \in \mathcal{F}$*, the* Margin Disparity Discrepancy *is defined by*

$$d_{f,\mathcal{F}}^{(\rho)}(\mathcal{S},\mathcal{T}) = \sup_{f' \in \mathcal{F}} [\epsilon_{\mathcal{T}}^{(\rho)}(f', f) - \epsilon_{\mathcal{S}}^{(\rho)}(f', f)]. \tag{23}$$

Note that the margin disparity satisfies the nonnegativity and subadditivity, but not the symmetry. Thus Theorem 6 cannot apply here and a new generalization bound is derived.

Theorem 10 (Zhang et al. (2019c)) *Given the same settings with Definition 9, for any $\delta > 0$, with probability at least $1 - 3\delta$, the following* margin bound *holds for all scoring functions $f \in \mathcal{F}$,*

$$\begin{aligned}\epsilon_{\mathcal{T}}(f) \leq \epsilon_{\widehat{\mathcal{S}}}^{(\rho)}(f) + d_{f,\mathcal{F}}^{(\rho)}(\widehat{\mathcal{S}}, \widehat{\mathcal{T}}) + \epsilon_{ideal} + \frac{2k^2}{\rho}\mathfrak{R}_{n,\mathcal{S}}(\Pi_1 \mathcal{F}) \\ + \frac{k}{\rho}\mathfrak{R}_{n,\mathcal{S}}(\Pi_{\mathcal{H}}\mathcal{F}) + 2\sqrt{\frac{\log\frac{2}{\delta}}{2n}} + \frac{k}{\rho}\mathfrak{R}_{m,\mathcal{T}}(\Pi_{\mathcal{H}}\mathcal{F}) + \sqrt{\frac{\log\frac{2}{\delta}}{2m}},\end{aligned} \tag{24}$$

where $\Pi_{\mathcal{H}}\mathcal{F} \triangleq \{x \mapsto f(x, h(x)) | h \in \mathcal{H}, f \in \mathcal{F}\}$ is the scoring version of the symmetric hypothesis space $\mathcal{H}\Delta\mathcal{H}$, $\Pi_1\mathcal{F} \triangleq \{x \mapsto f(x, y) | y \in \mathcal{Y}, f \in \mathcal{F}\}$ and $\epsilon_{ideal} = \min_{f^* \in \mathcal{F}}\{\mathrm{err}_{\mathcal{S}}^{(\rho)}(f^*) + \mathrm{err}_{\mathcal{T}}^{(\rho)}(f^*)\}$ is the ideal joint error in terms of margin loss, k is the number of classes.

This margin bound suggests that a proper margin ρ could yield better generalization on the target domain. Theorems 8 and 10 together form the theoretical basis of the *hypothesis adversarial* methods in Section 3.2.3. Note that the supremum in both the $\mathcal{H}\Delta\mathcal{H}$-Divergence and Disparity Discrepancy will become meaningless, when the allowed hypothesis space $\mathcal{H}$ is too large, which is common in deep neural networks. Thus, pre-training the deep neural networks on large-scale upstream data to decrease the allowed hypotheses is still necessary for the domain adversarial methods and hypothesis adversarial methods.

A final important note is that while there are no theoretical guarantees for some well-established methods, they have also achieved quite strong performance in practice, such as the *domain translation* methods in Section 3.2.4 and the *semi-supervised learning* methods in Section 3.2.5. Figure 18 highlights the cornerstones of domain adaptation methods in deep learning, which rely on the reuse of transferability gained in pre-trained deep models.

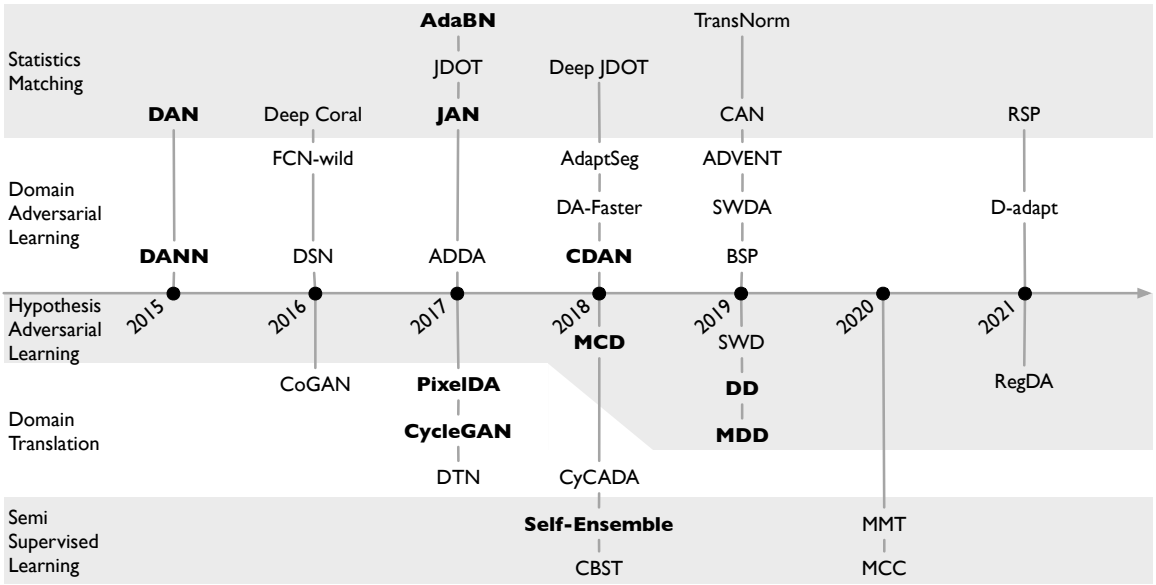


Figure 18: The cornerstones of domain adaptation methods in deep learning.

3.2.1 Statistics Matching

We have introduced several seminal theories on the generalization bounds for domain adaptation, which are all based on the *hypothesis-induced* distribution distances. These distances are less intuitive because they rely on unknown hypotheses and cannot be computed before learning the hypotheses. In this section, we first introduce another family of metrics on the space of *probability measures* well-studied in probability theory, which provide interpretable and complementary properties to the hypothesis-induced distribution distances and relate closely to a large set of domain adaptation algorithms (Long et al., 2015, 2017).

Definition 11 (Maximum Mean Discrepancy) *Given two probability distributions $\mathcal{S}$ and $\mathcal{T}$ on a measurable space $\mathbf{X}$, the* integral probability metric *(Redko et al., 2020) is defined as $d_{\mathcal{F}}(\mathcal{S},\mathcal{T}) \triangleq \sup_{f\in\mathcal{F}} \left|\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathcal{T}}[f(\mathbf{x})]\right|$, where $\mathcal{F}$ is a class of bounded functions on $\mathbf{X}$. Sriperumbudur et al. (2010) further restrict $\mathcal{F}$ as the unit ball in Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ endowed with a* characteristic *kernel k, $\mathcal{F} = \{f \in \mathcal{H}_k : ||f||_{\mathcal{H}_k} \leq 1\}$, leading to the* maximum mean discrepancy (MMD) *(Gretton et al., 2012a),*

$$d^2_{\text{MMD}}(\mathcal{S},\mathcal{T}) = \left\|\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}[\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathcal{T}}[\phi(\mathbf{x})]\right\|^2_{\mathcal{H}_k}, \tag{25}$$

where $\phi(x)$ is a feature map associated with kernel k such that $k(\mathbf{x},\mathbf{x}') = \langle\phi(\mathbf{x}),\phi(\mathbf{x}')\rangle$. It can be proved from probability theory that $\mathcal{S} = \mathcal{T}$ if and only if $d_{\mathcal{F}}(\mathcal{S},\mathcal{T}) = 0$ or $d^2_{\text{MMD}}(\mathcal{S},\mathcal{T}) = 0$.

Theorem 12 (Redko et al. (2020)) *Given the same settings with Definition 11, let ℓ be a convex loss function with a parametric form $\ell(y,y') = |y-y'|^q$ for some q. Then for any $\delta > 0$, with probability at least $1-\delta$, the following bound holds for all $h \in \mathcal{F}$,*

$$\begin{aligned}\epsilon_{\mathcal{T}}(h) \leq\ & \epsilon_{\mathcal{S}}(h) + d_{\text{MMD}}(\widehat{\mathcal{S}},\widehat{\mathcal{T}}) + \epsilon_{ideal} \\ & + \frac{2}{n}\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}[\sqrt{\text{tr}(\mathbf{K}_{\mathcal{S}})}] + \frac{2}{m}\mathbb{E}_{\mathbf{x}\sim\mathcal{T}}[\sqrt{\text{tr}(\mathbf{K}_{\mathcal{T}})}] + \sqrt{\frac{\log\frac{2}{\delta}}{2n}} + \sqrt{\frac{\log\frac{2}{\delta}}{2m}},\end{aligned} \tag{26}$$

where $\mathbf{K}_{\mathcal{S}}$ and $\mathbf{K}_{\mathcal{T}}$ are the kernel matrices computed on samples from $\mathcal{S}$ and $\mathcal{T}$, respectively.

This bound has several advantages compared to previous theories. First, it is *hypothesis-free* and does not require estimating hypotheses to measure the distribution distance. Second, the complexity term does not depend on the Vapnik-Chervonenkis dimension. Third, the unbiased estimate of MMD can be computed in linear time. Fourth, minimizing MMD has a nice interpretation of *statistics matching* in the probability space. These advantages make the bound particularly useful to underpin several seminal algorithms.

Deep Domain Confusion (DDC) (Tzeng et al., 2014) applies MMD with a linear kernel to a single feature layer of the deep network, yet it has limited power for closing the domain gap since linear kernel is not characteristic and cannot ensure MMD to be a probability metric. Thereby, Deep Adaptation Network (DAN) (Long et al., 2015, 2019) introduces the multiple-kernel variant of MMD (MK-MMD) (Gretton et al., 2012b,a) to measure the domain relatedness, employing a convex combination of multiple characteristic kernels such as Gaussian kernel to make the function space $\mathcal{F}$ rich enough and enhance the distinguishing power of MK-MMD. Besides, as shown in Figure 19, multiple domain-specific layers are adapted by MK-MMD, which enables learning transferable features for domain adaptation.

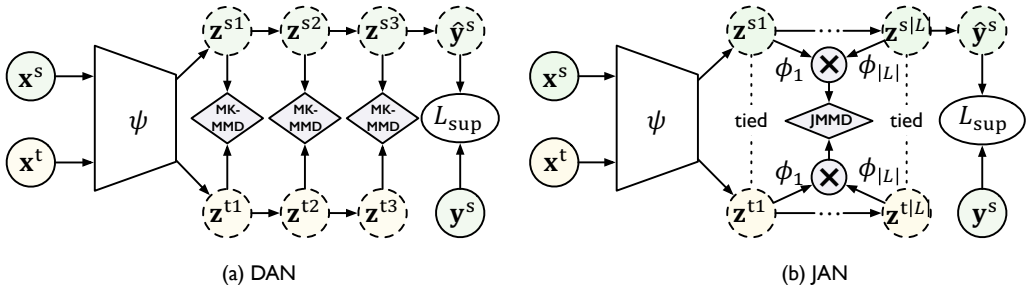


Figure 19: The cornerstone methods of statistics matching: (a) DAN adapts the *marginal* distributions of activations in multiple task-specific layers with MK-MMD. (b) JAN adapts the *joint* distributions of the feature activations and classification predictions with JMMD.

DAN mainly reduces the shift in the feature distribution and ignores that in the label distribution. Take AlexNet as example, the feature distribution shift mainly exists in layers $fc6$ and $fc7$ while the label distribution shift mainly exists in layer $fc8$. Joint Adaptation Network (JAN) (Long et al., 2017) proposes Joint Maximum Mean Discrepancy (JMMD) to measure the shift in joint distributions $P(\mathbf{X}^s, \mathbf{Y}^s)$ and $Q(\mathbf{X}^t, \mathbf{Y}^t)$. Denoting the activations of adapted layers $\mathcal{L}$ as $\{(\mathbf{z}_i^{s1}, \ldots, \mathbf{z}_i^{s|\mathcal{L}|})\}_{i=1}^n$ and $\{(\mathbf{z}_j^{t1}, \ldots, \mathbf{z}_j^{t|\mathcal{L}|})\}_{j=1}^m$, JMMD is defined as

$$d^2_{\text{JMMD}}(\widehat{\mathcal{S}}, \widehat{\mathcal{T}}) = \left\| \mathbb{E}_{i\in[n]} \otimes_{l\in\mathcal{L}} \phi^l(\mathbf{z}_i^{sl}) - \mathbb{E}_{j\in[m]} \otimes_{l\in\mathcal{L}} \phi^l(\mathbf{z}_j^{tl}) \right\|^2_{\mathcal{H}_k} \tag{27}$$

where ϕ^l is the feature map associated with kernel k^l for layer l and $\otimes$ is the outer product.

A characteristic kernel widely used in MMD is the Gaussian kernel, or $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-||\mathbf{x}_1 - \mathbf{x}_1||^2/2\sigma^2)$. After Taylor expansion, MMD can be considered as a weighted sum of distances between all orders of statistic moments. Thus, the statistic moments can be directly used to measure the distribution distance. For instance, deep CORAL (Sun and Saenko, 2016) uses the second-order statistics (covariance) to measure distribution distance, which is frustratingly easy yet useful. Center moment discrepancy (CMD) (Zellinger et al., 2017) further considers an explicit order-wise matching of higher-order moments.

One disadvantage of MMD is that it cannot take into account the geometry of the data distribution when estimating the discrepancy between two domains. Thus, Joint Distribution Optimal Transport (JDOT) (Courty et al., 2017) is introduced into domain adaptation, and Deep JDOT (Damodaran et al., 2018) further extends it to deep networks. Another disadvantage is that minimizing MMD on the instance representation has the risk of changing the feature scale, while regression tasks are fragile to feature scaling. Thus, Representation Subspace Distance (RSD) (Chen et al., 2021b) closes the domain shift through orthogonal bases of the representation spaces, which are free from feature scaling.

Instead of explicitly matching the statistics moments of feature distributions, Adaptive Batch Normalization (AdaBN) (Li et al., 2017) implicitly minimizes domain discrepancy by aligning BatchNorm Ioffe and Szegedy (2015) statistics. The hypothesis is that task-related knowledge is stored in the weight matrix while domain-related knowledge is represented by BatchNorm statistics. Thus AdaBN replaces the mean and variance of all BatchNorm layers with those estimated on the target domain at inference to reduce the domain shift. However,

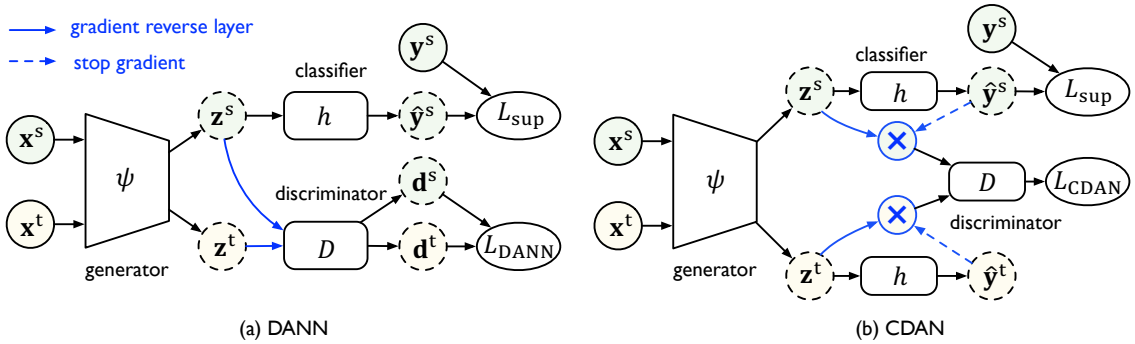


Figure 20: Both DANN and CDAN have a feature generator network ψ, a classifier h, and a domain discriminator D connected to ψ via a *gradient reversal layer*. (a) In DANN, the discriminator D is trained to distinguish between domains while the generator ψ tries to make the feature distributions indistinguishable for the discriminator. (b) In CDAN, the discriminator D is conditioned on the classifier prediction $\widehat{\mathbf{y}}$ via a multilinear map $\mathbf{z} \otimes \widehat{\mathbf{y}}$.

it is risky that AdaBN excludes the statistics on the target domain from training. Thus, Transferable Normalization (TransNorm) (Wang et al., 2019c) applies domain-specific mean and variance at both training and inference to capture sufficient statistics of both domains.

Finally, both MMD and JMMD may misalign samples from different classes due to a suboptimal modeling of the discriminative structure. To alleviate this problem, Contrastive Adaptation Network (CAN) (Kang et al., 2019) alternatively estimates the labels of target samples through clustering, and adapts the feature representations in a class-wise manner. Besides, CAN uses class-aware sampling for both domains to improve adaptation efficiency.

3.2.2 Domain Adversarial Learning

Domain Adversarial Neural Network. An important milestone for modeling distributions is the Generative Adversarial Net (GAN) (Goodfellow et al., 2014). Inspired by GAN, Domain Adversarial Neural Network (DANN) (Ganin and Lempitsky, 2015; Ganin et al., 2016) integrates a two-player game into domain adaptation (Figure 20). The first player is the domain discriminator D trained to distinguish the source features from the target features and the second player is the feature generator ψ trained simultaneously to confuse the domain discriminator. As mentioned in Section 3.2, the upper bound of $\mathcal{H}\Delta\mathcal{H}$-Divergence between feature distributions can be estimated by training the domain discriminator D,

$$L_{\text{DANN}}(\psi) = \max_{D} \mathbb{E}_{\mathbf{x}^s \sim \widehat{\mathcal{S}}} \log[D(\mathbf{z}^s)] + \mathbb{E}_{\mathbf{x}^t \sim \widehat{\mathcal{T}}} \log[1 - D(\mathbf{z}^t)], \tag{28}$$

where $\mathbf{z} = \psi(\mathbf{x})$ is the feature representation for $\mathbf{x}$. The objective of the feature generator ψ is to minimize the source error as well as the $\mathcal{H}\Delta\mathcal{H}$-Divergence bounded by Equation 28,

$$\min_{\psi, h} \mathbb{E}_{(\mathbf{x}^s, \mathbf{y}^s) \sim \widehat{\mathcal{S}}} L_{\text{CE}}(h(\mathbf{z}^s), \mathbf{y}^s) + \lambda L_{\text{DANN}}(\psi), \tag{29}$$

where L_{CE} is the cross-entropy loss, λ is a hyper-parameter that trades off source error and domain adversary. Minimizing the cross-entropy loss will lead to discriminative representations while decreasing the domain adversarial loss will result in transferable representations.

DANN aligns the marginal feature distributions through adversarial training. However, this may be insufficient when the feature-label joint distributions change between domains. Besides, the feature distribution is usually multimodal in multi-class classification, thus even if the discriminator is fully confused, there is no guarantee that the two feature distributions are similar (Arora et al., 2017). To tackle these two issues, Conditional Domain Adversarial Network (CDAN) (Long et al., 2018) conditions features $\mathbf{z}$ on classifier predictions $\widehat{\mathbf{y}} = h(\mathbf{z})$ and introduces multilinear map $\mathbf{z} \otimes \widehat{\mathbf{y}}$ instead of $\mathbf{z}$ as the input to domain discriminator D:

$$L_{\text{CDAN}}(\psi) = \max_{D} \mathbb{E}_{\mathbf{x}^s \sim \widehat{\mathcal{S}}} \log[D(\mathbf{z}^s \otimes \widehat{\mathbf{y}}^s)] + \mathbb{E}_{\mathbf{x}^t \sim \widehat{\mathcal{T}}} \log[1 - D(\mathbf{z}^t \otimes \widehat{\mathbf{y}}^t)]. \tag{30}$$

Conditioning on $\widehat{\mathbf{y}}$, CDAN fully captures cross-variance between the feature representation and classifier prediction, resulting in better alignment of the joint distributions.

Tzeng et al. (2015) align the class distributions explicitly by transferring the similarity structure in classes from the source domain to the target domain. Specifically, the average output probability of data from each class is computed over the source samples as soft labels. Then the model is optimized to match the distribution over classes to the soft labels.

Improvements. DANN integrates a *gradient reverse layer* (GRL) into the standard architecture to implement the minimax between the feature generator and domain classifier. However, this optimizing strategy might not work well in practice due to gradient vanishing, which is also a major problem in training GANs. For instance, when the generated target features $\mathbf{z}^t$ are very distinguishable from source features such that $D(\mathbf{z}^t) = 0$, the gradient for the feature generator is small and vice versa. This makes the optimization of the feature generator difficult. Thus, Adversarial Discriminative Domain Adaptation (ADDA) (Tzeng et al., 2017) splits the optimization of feature generator ψ and domain classifier D into two independent objectives, where the maximization of D remains unchanged, but the objective of ψ becomes

$$\min_{\psi} \mathbb{E}_{\mathbf{x}^t \sim \widehat{\mathcal{T}}} - \log[D(\mathbf{z}^t)]. \tag{31}$$

This assigns small gradients for source-like target samples and larger gradients for the other target samples. It has the same fixed-point properties as GRL while making the optimization easier for the feature generator. Although adversarial domain adaptation enhances the feature *transferability*, i.e. the ability of feature representations to bridge the discrepancy across domains, studies (Chen et al., 2019c) reveal that it is at the expense of deteriorating the *discriminability*, i.e. the easiness of separating categories over the fixed feature representations of both domains. Spectral analysis shows that only the eigenvectors corresponding to the largest singular values tend to carry transferable knowledge, while other eigenvectors may reflect domain variations and thus be overly penalized in domain adversarial training. However, these eigenvectors may convey crucial discriminative information, and thus the discriminability is weakened. To tackle this transferability-discriminability dilemma, Batch Spectral Penalization (BSP) (Chen et al., 2019c) penalizes the largest singular values so that the other eigenvectors can be relatively strengthened to boost the feature discriminability. Domain Separation Network (DSN) (Bousmalis et al., 2016) introduces a private subspace for each domain, which preserves domain-specific information, such as background and low-level image statistics. Then domain alignment is performed safely in the shared subspace, which is orthogonal to the private subspace responsible for the discriminative tasks.

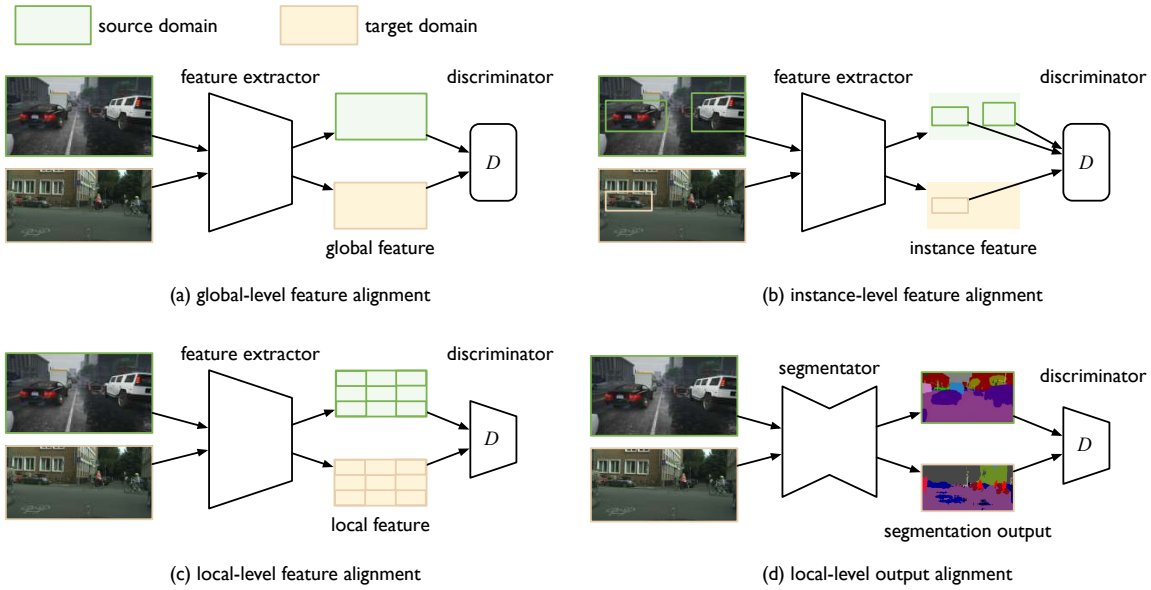


Figure 21: Where to adapt in different methods. (a) DANN performs alignment on global features. (b) DA-Faster performs alignment on both image-level and instance-level features. (c) SWDA performs alignment on local features. (d) Adapt-SegMap performs alignment on local outputs.

Domain Adversarial Leanring in Real-World Scenarios. Domain adversarial learning has largely improved the performance on unlabeled target domain and has been widely adopted in many applications (Hoffman et al., 2016; Chen et al., 2018). However, real-world scenarios are much more complex. Here we list several situations that are often encountered yet not well resolved, and review some existing solutions to them.

Which part to adapt is unknown. In image recognition, we only need to classify the input. Yet in applications such as object detection, we need to locate the *Region of Interests* (RoIs) first and then classify them. Due to the distribution shift across domains, the localization of RoIs on the target domain is not reliable, thus which part to adapt in the adversarial training is unknown. To alleviate this problem, as shown in Figure 21, DA-Faster (Chen et al., 2018) performs domain alignment at both image level and instance level, where the image-level alignment is believed to improve the localization of RoIs on the target domain. SWDA (Saito et al., 2019) argues that alignment of the local features is more effective than that of the global features for better localization. Although the above adversarial training methods have improved the transferability of object detectors, the discriminability might get lost in the adversarial adaptation process as mentioned by BSP (Chen et al., 2019c). Since discriminability is crucial for the localization of RoIs, D-adapt (Jiang et al., 2022) introduces parameter-independent category adaptor and bounding box adaptor to decouple adversarial adaptation from detector training, which yields sharp improvement.

There are structural dependencies between labels of each sample. In low-level classification tasks, such as semantic segmentation or token classification (Named Entity Recognition, Parts-of-speech tagging, etc.), adaptation on *features* (Hoffman et al., 2016) may not be a good option, because the feature of each pixel or each token is high-dimensional and there

exist many pixels or tokens in a single sample. Every coin has two sides. Compared to the high-level classification tasks, the *output space* of these low-level tasks contains much richer information of the distribution, e.g., scene layout and context, and thus adaptation on the output space can reduce the domain shift. As shown in Figure 21, Adapt-SegMap (Tsai et al., 2018) trains a discriminator to distinguish whether the segmentation output is from the source or the target, while the feature generator is encouraged to generate similar segmentation outputs across domains. It explicitly aligns the output distributions of target and source domains, and implicitly adapts the feature distributions. Further, ADVENT (Vu et al., 2019) minimizes the distribution distance on the *self-information* distributions, where the entropy of segmentation outputs is fed to the discriminator. In this way, the conditional information is neglected while more attention is paid to the structural dependencies between local semantics.

3.2.3 Hypothesis Adversarial Learning

Inevitably, there is still a gap between the *proxy distance* used in previous methods and the hypothesis-induced discrepancies in the theories. To close this gap, Maximum Classifier Discrepancy (MCD) (Saito et al., 2018) starts to estimate and optimize $\mathcal{H}\Delta\mathcal{H}$-Divergence in a fully parameterized way. As shown in Figure 22, MCD maximizes the discrepancy between two classifiers' outputs to detect target samples far from the support of source distribution, i.e. estimate $\mathcal{H}\Delta\mathcal{H}$-Divergence. A feature generator then learns to generate target features near the support to minimize the discrepancy, i.e. minimize the domain discrepancy. MCD uses the L_1 distance to calculate the discrepancy, while Sliced Wasserstein Discrepancy (SWD) (Lee et al., 2019) adopts the Wasserstein metric, which is the natural geometry for probability measures induced by the optimal transport theory. In theory MCD is closer to $\mathcal{H}\Delta\mathcal{H}$-Divergence, yet in experiments it is slow in convergence and very sensitive to hyper-parameters. The reason is that there exist two classifiers h and h' in MCD that maximize the discrepancy, which makes the minimax optimization hard to reach equilibrium.

Disparity Discrepancy (DD) (Zhang et al., 2019c) provides a tighter bound by taking supremum in hypothesis space $\mathcal{H}$ rather than $\mathcal{H}\Delta\mathcal{H}$. This will significantly ease the minimax optimization. As shown in Figure 22, DD introduces an adversarial classifier h' sharing the same hypothesis space with h. The supremum in $d_{h,\mathcal{H}}(\mathcal{S},\mathcal{T})$ is approximated by

$$\begin{aligned} L_{\mathrm{DD}}(h,\psi) = \max_{h'} \ & \mathbb{E}_{\mathbf{x}^s\sim\widehat{\mathcal{S}}} L^s\left[h'(\psi(\mathbf{x}^s)), h(\psi(\mathbf{x}^s))\right] \\ & - \mathbb{E}_{\mathbf{x}^t\sim\widehat{\mathcal{T}}} L^t\left[h'(\psi(\mathbf{x}^t)), h(\psi(\mathbf{x}^t))\right], \end{aligned} \tag{32}$$

where L^s and L^t are specific loss functions defined on the source domain and target domain respectively. Based on the theory (Zhang et al., 2019c), when the adversarial classifier h' is close to the supremum, minimizing the following terms will decrease the target error $\epsilon_{\mathcal{T}}$,

$$\min_{\psi,h} \ \mathbb{E}_{(\mathbf{x}^s,\mathbf{y}^s)\sim\widehat{\mathcal{S}}} L_{\mathrm{CE}}(h(\psi(\mathbf{x}^s)),\mathbf{y}^s) + \lambda L_{\mathrm{DD}}(h,\psi), \tag{33}$$

where λ is a tradeoff hyper-parameter. An intuitive explanation is that DD is looking for an adversarial classifier h' that predicts correctly on the source domain while making different predictions from h on the target domain. And then the feature generator ψ is encouraged to generate features near the decision boundary to avoid such situations.

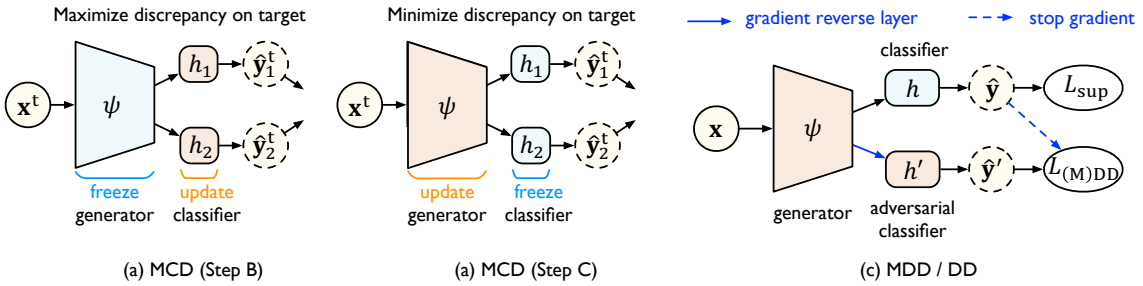


Figure 22: (a) The training of MCD has three steps. Step A: Both the classifiers and the feature generator are trained to classify the source samples correctly. Step B: The classifiers h_1 and h_2 learn to maximize the discrepancy on the target samples. (b) Step C: The feature generator ψ learns to minimize the discrepancy on the target samples. (c) DD and MDD introduce an adversarial classifier h' to maximize the discrepancy and trains the feature generator ψ to minimize the source error as well as the discrepancy.

However, DD is still limited to the 01 loss in the classification setting. Based on scoring functions and margin loss, Margin Disparity Discrepancy (MDD) (Zhang et al., 2019c) goes a crucial step forward and provides a *margin* theory for the multi-class classification setting. The margin ρ is attained by introducing parameter $\gamma \triangleq \exp \rho$ in the disparity discrepancy,

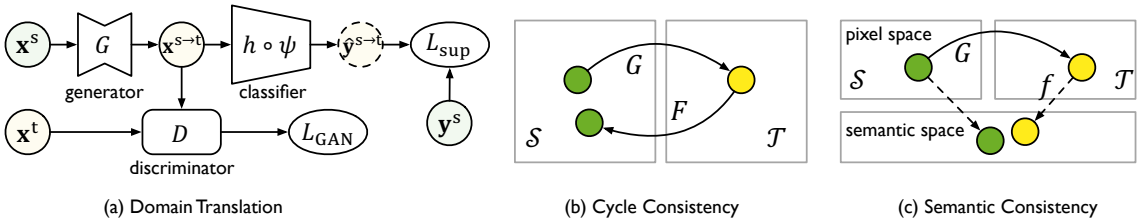
$$\begin{aligned} L_{\text{MDD}}(h, \psi) = \max_{h'} \; & \gamma \, \mathbb{E}_{\mathbf{x}^s \sim \widehat{\mathcal{S}}} \log \left[\sigma_{h(\psi(\mathbf{x}^s))}(h'(\psi(\mathbf{x}^s))) \right] \\ & + \mathbb{E}_{\mathbf{x}^t \sim \widehat{\mathcal{T}}} \log \left[1 - \sigma_{h(\psi(\mathbf{x}^t))}(h'(\psi(\mathbf{x}^t))) \right], \end{aligned} \tag{34}$$

where σ is the softmax function. A proper γ can constrain h' in a hypothesis space of proper size to avoid overestimation of the generalization bound. Note that in Equation 34, the loss on the source domain is the standard cross-entropy, while that on the target domain is a modified cross-entropy to avoid gradient vanishing and ease the optimization of h'.

In principle, DD can be easily extended to regression problems by replacing the classifiers in Figure 22 with regressors and choosing L^s and L^t as the L_1 or L_2 loss commonly used in regression. It has been extended to both keypoint detection (Jiang et al., 2021) and bounding box localization task (Jiang et al., 2022). To tackle the challenge caused by the high-dimensional output space in the keypoint detection, Regressive Domain Adaptation (RegDA) (Jiang et al., 2021) introduces a spatial probability distribution to describe the sparse density of the output space and uses it to guide the optimization of the adversarial regressor h'. In an expectation sense, this reduces the size of the hypothesis space of h' and avoids overestimation of the generalization bound in Zhang et al. (2019c).

3.2.4 Domain Translation

Domain translation is the task of mapping *raw data* of text, image, audio, and other data modality from the source distribution $\mathcal{S}$ to the target distribution $\mathcal{T}$. In domain adaptation problems, we can use translation models, usually based on Generate Adversarial Networks (GAN) (Goodfellow et al., 2014), to obtain labeled source domain in the target style, i.e. translated into the target distribution. Training on such stylized source domain can yield better transferability than models trained on the original source domain.



(a) Domain Translation (b) Cycle Consistency (c) Semantic Consistency

Figure 23: (a) The architecture for PixelDA includes a generator network G, an adversarial discriminator D and a task-specific classifier h on feature extractor ψ. (b) Cycle-consistency loss: after we translate from source domain to target domain, we should recover the source data if translating back again. (c) Semantic-consistency loss: translating between domains should not change the semantic labels of the samples, where f is the labeling function.

GANs reason about the marginal distribution, i.e., from a random vector, the generator network should synthesize data that resembles one that is drawn from the true distribution. However, marginal distribution is not enough for domain adaptation, thus Coupled Generative Adversarial Networks (CoGAN) (Liu and Tuzel, 2016) learns a joint distribution of multi-domain images from data, i.e., from a random vector, multiple generators should generate paired data that are from different distributions and share the same labels. By enforcing a weight-sharing constraint between different generators, CoGAN learns a joint distribution without the existence of corresponding images in different domains. Then the shared labels of the target samples are used to train the target model.

A more common objective of domain translation is to learn a mapping $G : \mathcal{S} \to \mathcal{T}$ such that the generated sample $G(\mathbf{x})$ is indistinguishable from the training samples of the target domain. As shown in Figure 23, PixelDA (Bousmalis et al., 2017) introduces an adversarial discriminator D to distinguish between translated samples and target samples,

$$L_{\text{GAN}}(G) = \max_D \mathbb{E}_{\mathbf{x}\sim\widehat{\mathcal{S}}} \log\left[1 - D(G(\mathbf{x}))\right] + \mathbb{E}_{\mathbf{x}\sim\widehat{\mathcal{T}}} \log\left[D(\mathbf{x})\right]. \tag{35}$$

The generator G tries to synthesize samples $G(\mathbf{x})$ that look similar to images from the target domain by $\min_G L_{\text{GAN}}(G)$. The task-specific classifier h and feature extractor ψ are trained supervisedly on the target-style generated data by $\min_{\psi,h} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\widehat{\mathcal{S}}} L_{\text{sup}}(h \circ \psi(G(\mathbf{x})), \mathbf{y})$.

Cycle Consistency. While GAN can learn a mapping between two datasets, the desired mapping may not be obtained. The source sample may be projected to an irrelevant target sample, destroying the structure or content of the original samples. Besides, multiple source samples may be mapped to the same target sample, leading to the well-known problem of *mode collapse* (Goodfellow et al., 2014). Therefore, CycleGAN (Zhu et al., 2017) introduces an additional mapping $F : \mathcal{T} \to \mathcal{S}$ from target to source and adds a constraint of *cycle consistency* to reduce the space of possible mapping functions (Figure 23). Mathematically, cycle consistency requires F and G to be bijections and inverse of each other. In practice, CycleGAN constrains $F(G(\mathbf{x})) \approx \mathbf{x}$ and $G(F(\mathbf{x})) \approx \mathbf{x}$, which preserves the structure or content of samples to obtain more meaningful mappings. CycleGAN has been widely used in domain adaptation problems, such as image classification (Hoffman et al., 2018), semantic segmentation (Hoffman et al., 2018), person re-identification (Wei et al., 2018), robotic

grasping (Bousmalis et al., 2018), object detection (Kim et al., 2019), etc. The idea goes beyond the field of image translation and is widely used in other fields, such as unsupervised machine translation (Lample et al., 2017), where the cycle consistency is also called back-translation. Unsupervised machine translation can further be used for cross-lingual domain adaptation tasks (Conneau et al., 2018), where a language is a domain.

Semantic Consistency. CycleGAN is a general-purpose translation model for vision tasks and is apt at style transfer between datasets. However, it is difficult for CycleGAN to maintain the semantic information. It has been experimentally shown that when mapping from source to target, the problem of label flipping will easily occur (Bousmalis et al., 2017; Hoffman et al., 2018). As a result, there will exist a lot of noisy labels in the translated dataset, hurting the performance of the target model. Thus, ensuring semantic consistency is important for translation-based domain adaptation. Formally, given the labeling function f, the labels assigned to sample $\mathbf{x}$ should be consistent with that of the translated sample, i.e., $f(\mathbf{x}) = f(G(\mathbf{x}))$ (Figure 23). Since function f is not accessible, several proxy functions have been proposed to approximate the semantic consistency (Taigman et al., 2017; Hoffman et al., 2018; Bousmalis et al., 2018). Given a proxy function h_{p} and a distance measure d, the objective is to reduce the semantic inconsistency,

$$\min_{G} L_{\mathrm{sc}}(G, h_{\mathrm{p}}) = d(h_{\mathrm{p}}(\mathbf{x}), h_{\mathrm{p}}(G(\mathbf{x}))). \tag{36}$$

DTN (Taigman et al., 2017) and SimGAN (Shrivastava et al., 2017) use the feature extractor as a proxy function and the goal is to translate the low-level style while keeping the high-level features invariant. PersonGAN (Wei et al., 2018) uses the foreground crop of the person image as the proxy function of the person's identity, which ensures that a person's identity remains the same before and after translation. However, the constraint on feature or pixel space might be too strong, making it difficult to change the low-level style. Therefore, Cycle-consistent Adversarial Adaptation (CyCADA) (Hoffman et al., 2018) utilizes a pre-trained source model as a proxy function to encourage generating samples that have consistent predictions under the function. This proxy function effectively avoids label flipping during the translation of handwritten digit images, yet it is still not perfect. When faced with the dataset shift of real-world problems, the predictions on the generated samples are not reliable and may provide incorrect guidance to G.

When the domain difference is primarily low-level, such as textures, illumination, and color, translation can effectively close the domain gap. But when the domain is different at the high level, such as from different camera angles, translation may fail to adapt domains. Therefore, translation methods at the low-level and adaptation methods at the high-level mentioned in Section 3.2.1-3.2.3 are complementary and can be combined in practical applications (Hoffman et al., 2018). For example, Generate to Adapt (Sankaranarayanan et al., 2018) directly uses the generative task as an auxiliary task to align features across domains.

3.2.5 Semi-Supervised Learning

Unsupervised Domain Adaptation (UDA) is closely related to Semi-Supervised Learning (SSL) since both of them aim at generalizing from the labeled samples to the unlabeled samples. The difference is that in SSL, both the labeled and unlabeled samples come from the same distribution while in UDA, the source and target distributions differ. Thus, SSL

tasks can be considered as a special case of UDA tasks and some SSL methods can be applied in UDA tasks. Since there is still no theoretical guarantee for SSL methods in the UDA scenario, the first question to answer is, under what assumptions can we use SSL in UDA? There are mainly three assumptions in SSL (Chapelle et al., 2006). (1) *Smoothness Assumption*: if two samples $\mathbf{x}_1$, $\mathbf{x}_2$ residing in a high-density region are close, then so should be their corresponding outputs $\mathbf{y}_1$, $\mathbf{y}_2$. (2) *Cluster Assumption*: if points are in the same cluster, they are likely to be of the same class. It can also be interpreted as the *Low-density Separation Assumption*, where the decision boundary should lie in the low-density regions. (3) *Manifold Assumption*: the high-dimensional data shall lie roughly on a low-dimensional manifold. Both smoothness assumption and cluster assumption are helpful for classification, but not for regression problems. Thus SSL is used more commonly in classifier adaptation. Here we review several SSL methods applied to the UDA problems.

Consistency Regularization encourages consistent predictions for similar data points. Similar data points are generated by performing different data augmentations on the same data point. While many augmentation techniques are proposed for images, few are available for other data formats, such as texts and time series. Thus this type of method is limited to certain data modalities. Self-Ensemble (French et al., 2018) applies mean-teacher, a typical consistency regularization method, to image domain adaptation. The teacher model, which is an Exponential Moving Average (EMA) of the student model, will generate predictions to train the student model on the target domain. Due to the domain shift, the predictions are noisy, thus Mutual Mean-Teaching (MMT) (Ge et al., 2020) uses two collaborative networks jointly optimized under the supervision of mutual teacher models.

Entropy Minimization encourages the model to make confident (i.e., low-entropy) predictions on unlabeled data. It serves as an auxiliary term in many domain adaptation methods (Long et al., 2016, 2018; Saito et al., 2018; Shu et al., 2018; Vu et al., 2019). The risk is that the predictions on the target domain are not reliable, and entropy minimization may hurt the performance of the model. Thus, Minimum Class Confusion (MCC) (Jin et al., 2020) introduces a weight for each instance, where uncertain samples have smaller weights to avoid minimizing entropy on the incorrectly classified samples. MCC further minimizes the instance-weighted confusion between different classes, which is simple yet frustratingly effective. Source Hypothesis Transfer (Liang et al., 2020) adopts an information maximization loss with a fair diversity-promoting objective, which circumvents the trivial solutions in entropy minimization that all unlabeled data have the same one-hot encoding.

Pseudo-Labeling produces proxy labels on unlabeled data and uses these noisy labels together with the labeled data to train the model. In self-training, a confidence threshold is used to filter out unreliable proxy labels, which may fail in UDA since the model is likely to be biased towards well-transferred classes while ignoring other hard classes. Thus, Class-Balanced Self-Training (CBST) (Zou et al., 2018) uses a class-wise confidence threshold. Still, large noise exists in the generated pseudo labels on the target domain, and the standard Cross-Entropy (CE) loss has been shown to be sensitive to label noise (Zhang et al., 2017). Towards this problem, Zhang and Sabuncu (2018) propose the Generalized Cross-Entropy (GCE) loss as an effective solution (Rusak et al., 2021; Liu et al., 2021a),

$$L_{\mathrm{GCE}}(\mathbf{x}, \tilde{y}) = 1/q \cdot (1 - h_{\tilde{y}}(\mathbf{x})^q), \tag{37}$$

where $q \in (0, 1]$ is a hyper-parameter to trade-off between the CE loss and the MAE loss.

3.2.6 Remarks

Different domain adaptation methods are compared from several perspectives in Table 5. First, statistics matching, domain adversarial learning, and hypothesis adversarial learning methods are derived from theory, enjoying theoretical guarantees while domain translation and semi-supervised learning methods are still in the empirical regime. Second, the former three categories of methods work in the feature space or the output space and are highly related to specific tasks, and some are tightly integrated to specific architectures. In contrast, translation methods work in the input space and are relatively independent of specific tasks. However, translation models and semi-supervised learning are dependent on specific data format, and are hard to scale to different modalities. Finally, statistics matching methods are based on nonparametric distances, which are data-efficient but weak in expressiveness, thereby more suitable for low-data regimes. In contrast, domain adversarial learning and hypothesis adversarial learning methods are based on parametric distances, which can only be measured throughout learning, but are more performant when scaling up data.

Table 5: Comparison between different domain adaptation methods.

	Adaptation Performance[1]	Data Efficiency[2]	Modality Scalability[3]	Task Scalability[4]	Theory Guarantee[5]
Statistics Matching	★	★★★	★★★	★★	★★★
Domain Adversarial Learning	★★	★★	★★★	★★	★★★
Hypothesis Adversarial Learning	★★★	★★	★★★	★★	★★★
Domain Translation	★★	★	★	★★★	★
Semi-Supervised Learning	★★	★★	★★	★	★

[1] Performance: performance when there are large-scale data in source and target domains.
[2] Data Efficiency: performance when there are only small-scale data in source and target domains.
[3] Modality Scalability: whether can adapt the model to various modalities, such as text, time series.
[4] Task Scalability: whether can adapt the model to different tasks, such as regression, detection.
[5] Theory Guarantee: whether the generalization error of target domain can be bounded in adaptation.

Domain adaptation is closely related to pre-training and task adaptation. First, pre-training can boost the *transferability* in domain adaptation, since pre-training will reduce the allowed hypothesis space and decrease the generalization bound on the target domain, as mentioned in Section 3.2. Thus pre-training on the source domain also serves as the first step in many domain adaptation methods, such as RegDA (Jiang et al., 2021). Pre-training also provides some new solutions for domain adaptation. When there exists a large unlabeled target domain, a feasible solution is to first perform unsupervised pre-training on the target domain, and then fine-tune with the labeled data on the source domain. This is widely adopted in cross-lingual adaptation (Lample and Conneau, 2019).

When using pre-trained models for domain adaptation, we will also encounter the problems in task adaptation, such as the *catastrophic forgetting* mentioned in 3.1.1. Thus, many domain adaptation methods babysit the learning rates to avoid catastrophic forgetting (Long et al., 2015; Ganin and Lempitsky, 2015). Compared with task adaptation, domain adaptation increases the restriction on the task space, where the task of the source domain and that of the target domain must be the same. Due to this restriction, domain adaptation has a strict theoretical guarantee. But this restriction is sometimes hard to satisfy in prac-

tice since we cannot ensure whether the category on the unlabeled target domain is exactly the same as the source domain (Busto and Gall, 2017). Therefore, real-world adaptation is often a mix of task adaptation and domain adaptation. How to explore the transferability in such a practical *open-domain* scenario is a problem to be solved.

4. Evaluation

Evaluation serves as a means for (1) measuring the performance of different architectures, different pre-training and adaptation methods, and (2) understanding the strengths and limitations of different methods. This section will elaborate on the evaluation of *transferability*, which is defined by the performance on the target task or domain. We believe that the evaluation of different methods should be performed on large-scale datasets for a practical and meaningful comparison. Thus, in Section 4.1 we list some large-scale datasets that are suitable for evaluating transferability in deep learning. Since different methods are often based on different codebases, a fair comparison between them is rather difficult. To fill this blank, in Section 4.2 we propose an open-source library, `TLlib`, to better evaluate transferability of different methods in a unified framework. Finally, Section 4.3 provides several benchmarks for evaluating both the cross-task transferability and cross-domain transferability.

4.1 Datasets

To evaluate the transferability in deep learning, we list several datasets that are large-scale in the number of samples and categories, the richness of tasks, and the diversity of domains.

The *General Language Understanding Evaluation* (GLUE) (Wang et al., 2019a) is one of the most famous benchmarks in NLP. As shown in Table 6, it consists of nine sentence or sentence-pair language understanding tasks, covering a diverse range of dataset sizes, text genres, and degrees of difficulty. It is widely used to evaluate the *cross-task* transferability of different pre-training and task adaptation methods.

Table 6: Descriptions and statistics of the GLUE datasets.

Corpus	#Train	#Test	Metrics	Task	Domain
CoLA	8.5k	1k	Matthews corr	acceptability	misc.
SST-2	67k	1.8k	acc.	sentiment	movie reviews
MRPC	3.7k	1.7k	acc./F1	paraphrase	news
STS-B	7k	1.4k	Pearson/Spearman corr	sentence similarity	misc.
QQP	364k	391k	acc./F1	paraphrase	social QA questions
MNLI	393k	20k	matched acc./mismatched acc.	NLI	misc.
QNLI	105k	5.4k	acc	QA/NLI	Wikipedia
RTE	2.5k	3k	acc	NLI	news, Wikipedia
WNLI	634	146	acc	coreference/NLI	fiction books

In contrast, there is no common benchmark to evaluate the transferability of different methods in computer vision. Table 7 lists some of the widely used vision datasets. *Food-101, CIFAR-10, CIFAR-100, SUN397, Stanford Cars, FGVC Aircraft, DTD, Oxford-III Pets, Caltech-101, Oxford 102 Flowers* are used to evaluate the transferability of different architectures under task discrepancy (Kornblith et al., 2019). *ImageNet-R(endition)* (Hendrycks et al., 2021) and *ImageNet-Sketch* (Wang et al., 2019b) are two variants of the ImageNet,

mainly used to evaluate the *cross-domain* transferablity of different architectures and pre-training methods. *DomainNet* (Peng et al., 2019) has multiple domains sharing the same category space, and is used to evaluate the *cross-domain* transferablity of different domain adaptation methods under large domain shift.

Table 7: Descriptions and statistics of typical vision datasets.

Dataset	#Train	#Test	#Classes	Metric	Domain
Food-101	75,750	25,250	101	top-1	photos and real world images
CIFAR-10	50,000	10,000	10	top-1	photos and real world images
CIFAR-100	50,000	10,000	100	top-1	photos and real world images
SUN397	19,850	19,850	397	top-1	photos and real world images
Stanford Cars	8,144	8,041	196	top-1	photos and real world images
FGVC Aircraft	6,667	3,333	100	mean per-class	photos and real world images
Describable Textures (DTD)	3,760	1,880	47	top-1	photos and real world images
Oxford-III Pets	3,680	3,369	37	mean per-class	photos and real world images
Caltech-101	3,060	6,084	102	mean per-class	photos and real world images
Oxford 102 Flowers	2,040	6,149	102	mean per-class	photos and real world images
ImageNet-R	-	30k	200	top-1	art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, and video games
ImageNet-Sketch	-	50k	1000	top-1	sketch
DomainNet-c	33,525	14,604	365	top-1	clipart images
DomainNet-p	50,416	21,850	365	top-1	artistic paintings
DomainNet-r	120,906	52,041	365	top-1	photos and real world images
DomainNet-s	48,212	20,916	365	top-1	sketch

4.2 Library

To make up for the lack of a unified codebase in some areas, we propose an open and ongoing library, `TLlib`. This library implements many representative adaptation algorithms in a unified way, allowing quantitative, fair, reproducible comparisons between different algorithms and promoting seamless integration of different pre-training or adaptation methods.

Library Usage. First, we give a short description of how to use `TLlib` using DANN as an instance. In the original implementation of DANN, the domain adversarial loss, domain discriminator, feature generator, and classifier are tightly coupled together in one `nn.Module`, which causes the difficulty of reuse, e.g., the entire algorithm needs re-implementation when the input data is changed from image to text. Yet in this case, the domain adversarial loss and the domain discriminator remain unchanged and shall be reused. Therefore, in `TLlib`, models and loss functions are decoupled. When using DANN for any case, users need only to initialize a domain discriminator and pass it to the domain adversarial loss module, and then use this module in the same way as the cross-entropy loss module defined in PyTorch (example code below). `TLlib` provides friendly and coherent APIs for supported algorithms. Detailed usages of these algorithms can be found at the documentation.

```
>>> # define the domain discriminator
>>> from dalib.modules.domain_discriminator import DomainDiscriminator
>>> discriminator = DomainDiscriminator(in_feature=1024, hidden_size=1024)
>>> # define the domain adversarial loss module
>>> from dalib.adptation.dann import DomainAdversarialLoss
>>> dann = DomainAdversarialLoss(discriminator, reduction='mean')
>>> # features from the source and target domain
>>> f_s, f_t = torch.randn(20, 1024), torch.randn(20, 1024)
>>> # calculate the final loss
>>> loss = dann(f_s, f_t)
```

Design Philosophy. `TLlib` is designed to be *extendible* by researchers and *simple* for practitioners. Currently, there are mainly two types of algorithm implementations. One is to encapsulate each algorithm in a Trainer, whose typical representative is `PyTorch-Lighting`. Users only need to feed the training data to it and do not need to care about the specific training process. Another strategy is to encapsulate the core loss function in each algorithm, and users need to implement the complete training process by themselves. A typical representative is `PyTorch` (Paszke et al., 2019). Although the former method is easier to use, it is less extendible. Since it is often necessary to adjust the training process in different transfer learning scenarios, `TLlib` adopts the latter method for better extendibility. We try our best to make `TLlib` easy to start with, e.g., we support the automatic download of most common transfer learning datasets so that users do not need to spend time on data preparation. Our code is in *PyTorch-style* and we provide training examples of different transfer algorithms in different scenarios, which allows users to quickly adapt to `TLlib` as long as they have learned PyTorch before. For more convenient algorithm selection, we provide a comprehensive benchmark among all those libraries. For faster algorithm reproduction, we provide training scripts for all the results in the benchmark.

`TLlib` is released under the MIT License and available at `https://github.com/thuml/Transfer-Learning-Library`. Documentation and tutorials are available on its website.

4.3 Benchmark

This section will present a benchmark of typical pre-training and adaptation methods on the large-scale datasets described in Section 4.1. Since such a benchmark is missing in the literature, we produce the results using the open library `TLlib` implemented in Section 4.2.

4.3.1 Pre-Training

Protocols. The transferability of pre-training methods is evaluated on the target task, where the adaptation process and data augmentations are kept the same for fair comparison. Hyper-parameters in adaptation are selected by the performance of target validation data.

Results. For the pre-training methods, the transferability cross different tasks and across different domains should be evaluated. Tables 8 and 9 list the performance on various downstream tasks with different architectures and pre-training tasks. It can be concluded that architectures and pre-training methods have a great impact on the *cross-task* transferability of deep networks. Table 10 lists the performance on ImageNet-Sketch and ImageNet-R with

different architectures and pre-training tasks. Architectures and pre-training strategies also greatly influence the *cross-domain* transferability in deep learning.

Table 8: Cross-task transferability benchmark. Results of different architectures and pre-training methods are reported from the GLUE leaderboard. BiLSTM+ELMo (Peters et al., 2018) serves as the baseline. GPT (Radford et al., 2018), $\text{BERT}_{\text{Large}}$ (Devlin et al., 2019), T5 (Raffel et al., 2020), and ERNIE (Sun et al., 2019b) have different architectures. RoBERTa (Liu et al., 2019c), XLM (Lample and Conneau, 2019), and SpanBERT (Joshi et al., 2020) share the same architecture as $\text{BERT}_{\text{Large}}$ but employ different pre-training methods.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI_m	MNLI_{mm}	QNLI	RTE	Avg
Human Baselines	66.4	97.8	86.3	92.7	80.4	92	92.8	91.2	93.6	88.1
BiLSTM +ELMo	36.0	90.4	84.9	73.3	64.8	76.4	76.1	79.9	56.8	71.0
GPT	45.4	91.3	82.3	80.0	70.3	82.1	81.4	88.1	56.0	75.2
$\text{BERT}_{\text{Large}}$	60.5	94.9	89.3	86.5	72.1	86.7	85.9	92.7	70.1	82.1
T5	71.6	97.5	92.8	93.1	90.6	92.2	91.9	96.9	92.8	91.0
ERNIE	75.5	97.8	93.9	93.0	90.9	92.3	91.7	97.3	92.6	91.7
RoBERTa	67.8	96.7	92.3	92.2	90.2	90.8	90.2	95.4	88.2	89.3
XLM	62.9	95.6	90.7	88.8	89.8	89.1	88.5	94.0	76.0	86.2
SpanBERT	64.3	94.8	90.9	89.9	89.5	88.1	87.7	94.3	79.0	86.5

Table 9: Cross-task transferability benchmark. Results on image recognition using different pre-training methods, including SimCLR (Chen et al., 2020) and BYOL (Grill et al., 2020).

Model	Pre-Training	Food	CIFAR10	CIFAR100	SUN397	Cars	Aircraft	DTD	Pets	Caltech101	Flowers	Avg
ResNet50	Random Init	86.9	95.9	80.2	53.6	91.4	85.9	64.8	81.5	72.6	92.0	80.5
	SimCLR	88.2	97.7	85.9	63.5	91.3	88.1	73.2	89.2	92.1	97.0	86.6
	BYOL	88.5	97.8	86.1	63.7	91.6	88.1	76.2	91.7	93.8	97.0	87.5
ResNet50	Supervised	87.8	96.8	84.5	64.7	91.7	86.6	75.2	92.5	91.8	97.5	86.9
ResNet101	Pre-Trained	87.6	97.7	87.0	64.8	91.7	85.6	75.4	94.0	93.1	97.9	87.5
ResNet152	on ImageNet	87.6	97.9	87.6	66.0	92.0	85.3	74.9	94.5	93.2	97.4	87.6

Table 10: Cross-domain transferability benchmark. Results are reported from the PyTorch-Image-Models (Wightman, 2019) on ImageNet using different architectures and pre-training methods. SSP refers to semi-supervised pre-training on YFCC100M (Yalniz et al., 2019). WSP refers to weakly supervised pre-training on IG-1B-Targeted (Mahajan et al., 2018).

Model	Pre-Training	Param Count	ImageNet-Sketch		ImageNetR	
			top-1	top-5	top-1	top-5
ResNet50	Standard	25.6	29.6	46.8	40.4	54.7
ResNet152d	Pre-Trained	60.2	37.9	58.4	49.3	64.4
$\text{ViT}_{\text{large, patch16}}$	on ImageNet	304.3	51.8	73.7	64.3	76.2
ResNext101_32x8d	Standard	88.8	29.4	48.5	42.6	58.3
	SSP		34.1	55.6	49.2	65.5
	WSP		54.9	77.5	75.9	86.2
	SSP+WSP		56.4	78.9	75.6	87.1

4.3.2 Task Adaptation

Protocols. We follow the common practice in the community as described in Kornblith et al. (2019). Training iterations and data augmentations are kept the same for different task adaptation methods for a fair comparison. Hyper-parameters, such as learning rate and weight decay, of each method are selected by the performance on target validation data.

Results. We mainly investigate the *cross-task* transferability between different task adaptation methods. Tables 11 and 12 compare the performance of downstream tasks with different task adaptation methods. Note that previous methods usually only report results on individual datasets, such as Aircraft and Stanford Cars, where regularization tuning performs better than vanilla fine-tuning by a large margin. But the average improvements brought by different task adaptation methods on a large number of datasets are still limited. Thus, we can conclude that the effectiveness of different task adaptation algorithms largely depends on the relatedness between the target task and the pre-training task.

Table 11: Cross-task transferability benchmark. GLUE performance with different task adaptation methods, including SMART (Jiang et al., 2020), Adapter-Tuning (Houlsby et al., 2019) and Diff Pruning (Guo et al., 2021). Results are reported from their original papers.

Model	Task Adaptation	New Params Per Task	CoLA	SST-2	MRPC	STS-B	QQP	MNLI_m	MNLI_{mm}	QNLI	RTE	Avg
RoBERTa	vanilla	100%	67.8	96.7	92.3	92.2	90.2	90.8	90.2	95.4	88.2	89.3
	SMART	100%	65.1	97.5	93.7	92.9	90.1	91.0	90.8	95.4	87.9	89.4
$\text{BERT}_{\text{Large}}$	vanilla	100%	60.5	94.9	89.3	86.5	72.1	86.7	85.9	92.7	70.1	82.1
	Adapter	2.10%	59.2	94.3	88.7	87.3	89.4	85.4	85.0	92.4	71.6	83.7
	Diff Pruning	0.50%	61.1	94.1	89.7	86.0	-	86.4	86.0	93.3	70.6	-

Table 12: Cross-task transferability benchmark. Accuracy (%) on image classification with different task adaptation methods: LWF (Li and Hoiem, 2018), DELTA (Li et al., 2019), BSS (Chen et al., 2019b), Bi-Tuning (Zhong et al., 2020). Results are reproduced by `TLlib`.

Task Adaptation	Food	CIFAR10	CIFAR100	SUN397	Cars	Aircraft	DTD	Pets	Caltech101	Flowers	Avg
ResNet50	85.1	96.9	84.1	80.7	87.8	80.1	74.4	93.2	92.9	96.5	87.2
LWF	83.9	96.5	83.6	79.5	87.4	82.2	76.3	94.0	91.7	97.1	87.2
DELTA	83.8	95.9	83.7	73.6	88.1	82.3	75.6	94.2	92.5	97.0	86.7
BSS	85.0	96.6	84.2	80.4	88.4	81.8	74.3	93.3	93.0	96.6	87.4
Bi-Tuning	85.7	97.1	84.3	80.7	90.3	84.8	74.6	93.5	93.4	97.5	88.2

4.3.3 Domain Adaptation

Protocols. We follow the standard protocols for unsupervised domain adaptation (Long et al., 2015; Ganin and Lempitsky, 2015). Training iterations and data augmentations are kept the same for different methods for a fair comparison. For each method, hyper-parameters are selected on one task and then kept the same for all other tasks, requiring the hyper-parameters of each method to transfer across tasks. This selection strategy is more

executable than the importance-weighted cross-validation (Sugiyama et al., 2007) and can be applied to various practical applications, thus it is widely adopted by many competitions.

Results. Tables 13 and 14 give the classification performance of different domain adaptation methods on DomainNet and ImageNet. We find that many state-of-the-art methods on small datasets do not perform well on large-scale datasets. This field shall pay more attention to improving the *cross-domain* transferability of deep models on large-scale datasets.

Table 13: Cross-domain transferability benchmark. Accuracy (%) for unsupervised domain adaptation on DomainNet. Results are reproduced from `TLlib`.

DomainNet	c→p	c→r	c→s	p→c	p→r	p→s	r→c	r→p	r→s	s→c	s→p	s→r	Avg
ResNet101	32.7	50.6	39.4	41.1	56.8	35.0	48.6	48.8	36.1	49.0	34.8	46.1	43.3
DAN (2015)	38.8	55.2	43.9	45.9	59.0	40.8	50.8	49.8	38.9	56.1	45.9	55.5	48.4
DANN (2016)	37.9	54.3	44.4	41.7	55.6	36.8	50.7	50.8	40.1	55.0	45.0	54.5	47.2
ADDA (2017)	38.4	54.1	44.1	43.5	56.7	39.2	52.8	51.3	40.9	55.0	45.4	54.5	48.0
JAN (2017)	40.5	56.7	45.1	47.2	59.9	43.0	54.2	52.6	41.9	56.6	46.2	55.5	50.0
CDAN (2018)	40.4	56.8	46.1	45.1	58.4	40.5	55.6	53.6	43.0	57.2	46.4	55.7	49.9
MCD (2018)	37.5	52.9	44.0	44.6	54.5	41.6	52.0	51.5	39.7	55.5	44.6	52.0	47.5
MDD (2019c)	42.9	59.5	47.5	48.6	59.4	42.6	58.3	53.7	46.2	58.7	46.5	57.7	51.8

Table 14: Cross-domain transferability benchmark. Accuracy (%) for unsupervised domain adaptation on ImageNet-scale datasets. Results are reproduced from `TLlib`.

Task Model	ImageNet→ImageNet-R ResNet50	ImageNet→ImageNet-Sketch ig_resnext101_32x8d
Source Only	35.6	54.9
DAN (Long et al., 2015)	39.8	55.7
DANN (Ganin et al., 2016)	52.7	56.5
JAN (Long et al., 2017)	41.7	55.7
CDAN (Long et al., 2018)	53.9	58.2
MCD (Saito et al., 2018)	46.7	55.0
MDD (Zhang et al., 2019c)	56.2	62.4

5. Conclusion

In this paper, we investigate how to acquire and apply transferability in the whole lifecycle of deep learning. In the pre-training section, we focus on how to improve the transferability of the pre-trained models by designing architecture, pre-training task, and training strategy. In the task adaptation section, we discuss how to better preserve and utilize the transferable knowledge to improve the performance of target tasks. In the domain adaptation section, we illustrate how to bridge the domain gap to increase the transferability for real applications. This survey connects many isolated areas with their relation to transferability and provides a unified perspective to explore transferability in deep learning. We expect this study will attract the community's attention to the fundamental role of transferability in deep learning.

Acknowledgments

This work was supported by the NSFC Grants (62022050 and 62021002), the Beijing Nova Program (Z201100006820041), and the BNRist Innovation Fund (BNR2021RC01002).

References

Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training. In *ICLR*, 2022.

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL*, 2021.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, 2016.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In *ICML*, 2017.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. In *JMLR*, 2002.

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: Pretrained language model for scientific text. In *EMNLP*, 2019.

S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning, 79*, page 151–175, 2010a.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *NeurIPS*, 2006.

Shai Ben-David, Tyler Lu, Teresa Luu, and David Pal. Impossibility theorems for domain adaptation. In *AISTATS*, pages 129–136, 2010b.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *ICML workshop*, 2012.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NeurIPS*, 2007.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *TPAMI*, 35(8):1798–1828, 2013.

Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. Deep learning for ai. *Communications of the ACM*, 64(7):58–65, 2021.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *NeurIPS*, 2016.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.

Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *ICRA*, 2018.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *ICCV*, 2017.

Rich Caruana. Multitask learning. Technical report, 1997.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006. ISBN 0262033585.

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, 2012.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019a.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021a.

Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. In *NeurIPS*, 2019b.

Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *ICML*, 2019c.

Xinyang Chen, Sinan Wang, Jianmin Wang, and Mingsheng Long. Representation subspace distance for domain adaptation regression. In *ICML*, 2021b.

Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster R-CNN for object detection in the wild. In *CVPR*, 2018.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, 2014.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. An embarrassingly simple approach for transfer learning from pretrained language models. In *NAACL*, 2019.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: evaluating cross-lingual sentence representations. In *EMNLP*, 2018.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *NeurIPS*, 2017.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, pages 4109–4118, 2018.

Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *ECCV*, 2018.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *TPAMI*, page 1–20, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Geoffrey French, Michal Mackiewicz, and Mark H. Fisher. Self-ensembling for domain adaptation. In *ICLR*, 2018.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(59):1–35, 2016.

Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In *ICLR*, 2018.

Yixiao Ge, Dapeng Chen, and Hongsheng Li. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. In *ICLR*, 2020.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

Boqing Gong, Kristen Grauman, and Fei Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML*, 2013.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. 2015.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *ICLR*, 2021.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 13(25):723–773, 2012a.

Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *NeurIPS*, 2012b.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, 2020.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *ICLR*, 2021.

Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *ACL*, 2021.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *CVPR*, 2019.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. 2016.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *ACL*, 2018.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Pre-training graph neural networks. In *ICLR*, 2020.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. In *NeurIPS*, 2007.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Yunhun Jang, Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Learning what and where to transfer. In *ICML*, 2019.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *ACL*, 2020.

Junguang Jiang, Yifei Ji, Ximei Wang, Yufeng Liu, Jianmin Wang, and Mingsheng Long. Regressive domain adaptation for unsupervised keypoint detection. In *CVPR*, 2021.

Junguang Jiang, Baixu Chen, Jianmin Wang, and Mingsheng Long. Decoupled adaptation for cross-domain object detection. In *ICLR*, 2022.

Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. In *ECCV*, 2020.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. In *TACL*, 2020.

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *CVPR*, 2019.

Taekyung Kim, Minki Jeong, Seunghyeon Kim, Seokeon Choi, and Changick Kim. Diversify and match: A domain adaptive representation learning paradigm for object detection. In *CVPR*, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114 (13):3521–3526, 2017.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019.

Zhi Kou, Kaichao You, Mingsheng Long, and Jianmin Wang. Stochastic normalization. In *NeurIPS*, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. In *NeurIPS*, 2019.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. In *ICLR*, 2017.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*, 2020.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *CVPR*, 2019.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. Mixout: Effective regularization to finetune large-scale pretrained language models. In *ICLR*, 2020a.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020b.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*, 2020.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *ICLR*, 2018.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, Zeyu Chen, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. In *ICLR*, 2019.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. In *ICLR Workshop*, 2017.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2018.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020.

Hong Liu, Jianmin Wang, and Mingsheng Long. Cycle self-training for domain adaptation. In *NeurIPS*, 2021a.

Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NeurIPS*, 2016.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021b.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *ACL*, 2019a.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. 2019c.

Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S. Yu. Transfer feature learning with joint distribution adaptation. In *ICCV*, 2013.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.

Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Transferable representation learning with deep adaptation networks. *TPAMI*, 41(12):3071–3085, 2019.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *ICLR*, 2018.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

Arun Mallya and Svetlana Lazebnik. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. In *ECCV*, 2018.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, 2017.

Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018.

Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *ICML*, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *NeurIPS*, 2019.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *TKDE*, pages 1345–1359, 2010.

Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *TNNLS*, pages 199–210, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pages 947–1012, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual bert? In *ACL*, 2019.

J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Technical report, OpenAI*, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67, 2020.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *ICLR*, 2020.

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. In *NeurIPS*, 2019.

S-A Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017.

Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. A survey on domain adaptation theory: learning bounds and theoretical guarantees, 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

Michael T. Rosenstein. To transfer or not to transfer. In *NeurIPS*, 2005.

Evgenia. Rusak, Steffen Schneider, Peter Gehler, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Adapting imagenet-scale models to complex distribution shifts with self-learning. *arXiv preprint arXiv:2104.12928*, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018.

Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection. In *CVPR*, 2019.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? In *NeurIPS*, 2020.

Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*, 2018.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

Timo Schick and Hinrich Schütze. Exploiting cloze questions for few-shot text classification and natural language inference. In *EACL*, 2020.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning.* PhD thesis, Technische Universität München, 1987.

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. In *ICML*, 2012.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577 (7792):706–710, 2020.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

Rui Shu, Hung H. Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *ICLR*, 2018.

Yang Shu, Zhangjie Cao, Jinghan Gao, Jianmin Wang, and Mingsheng Long. Omni-training for data-efficient deep learning. *arXiv preprint arXiv:2110.07510*, 2021a.

Yang Shu, Zhi Kou, Zhangjie Cao, Jianmin Wang, and Mingsheng Long. Zoo-tuning: Adaptive transfer from a zoo of models. In *ICML*, 2021b.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet. Hilbert space embeddings and metrics on probability measures. *JMLR*, 2010.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *JMLR*, 8(35):985–1005, 2007.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *NeurIPS*, 2008.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, 2019a.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019b.

Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 1998.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020.

Lisa Torrey and Jude Shavlik. Transfer learning. 2010.

Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *CVPR*, 2018.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. 2014.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, pages 4068–4076, 2015.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Perez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *CVPR*, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019a.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *NeurIPS*, 2019b.

Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *NeurIPS*, 2019c.

Ximei Wang, Jinghan Gao, Mingsheng Long, and Jianmin Wang. Self-tuning for data-efficient deep learning. In *ICML*, 2021.

Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime G. Carbonell. Characterizing and avoiding negative transfer. In *CVPR*, 2019d.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *CVPR*, 2018.

Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

Zhirong Wu, Yuanjun Xiong, X Yu Stella, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *EMNLP*, 2021.

I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *ICML*, 2019.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NeurIPS*, 2014.

Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, 2021.

Amir Roshan Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. In *ICLR*, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019a.

Jeffrey O. Zhang, Alexander Sax, Amir Zamir, Leonidas J. Guibas, and Jitendra Malik. Side-tuning: Network adaptation via additive side networks. 2019b.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *ICML*, 2019c.

Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.

Nanxuan Zhao, Zhirong Wu, Rynson W. H. Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? In *ICLR*, 2021.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? assessing self-supervised learning for law and the casehold dataset. In *ICAIL*, 2021.

Jincheng Zhong, Ximei Wang, Zhi Kou, Jianmin Wang, and Mingsheng Long. Bi-tuning of pre-trained representations. *arXiv preprint arXiv:2011.06182*, 2020.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 2015.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.

Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018.