Distributional Robustness: Towards Challenging and Real-World Settings of Distribution Shift



A. Tuan Nguyen

St Cross College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

February 2024

With love, I dedicate this thesis to
my family, especially my late grandmother,
for always being by my side.

Acknowledgements

First of all, I am indebted to my supervisor, Prof. Philip Torr, for his constant and invaluable support during my time at Oxford. Phil gave me the space to freely explore research ideas and the time I needed to dive deeply into each one of them. More importantly, he taught me how to approach each problem, and how to convey my research clearly to the audience. I would also like to thank my co-supervisors, Dr. Atılım Güneş Baydin and Prof. Yarin Gal. I worked with Güneş and Yarin mostly in the early stage of my DPhil, when I was very new to research, and learned a lot from them. I also want to express my gratitude to Ser-Nam Lim, who acted as my "industry" mentor. Ser-Nam provided me with an external and additional source of advice, from which I benefited significantly. I am also deeply thankful to all my professors, teachers, and peers who have motivated me and imparted invaluable knowledge, not only during my PhD study but also throughout my life.

Secondly, I would like to thank Prof. Andrea Vedaldi and Prof. Stephen Roberts who assessed my transfer and confirmation of status. Andrea and Steve gave me many constructive feedbacks and criticisms and helped shape this thesis. I am also grateful to Prof. Aandrea Vedaldi and Dr. Carl Henrik Ek who are my DPhil examiners, despite their busy schedules.

Last but importantly, I would like to express my love and gratitude to my father Nguyễn Xuân Trường, my mother Phạm Thị Luận, my grandfather Phạm Xuân Ngoạn, my sister Nguyễn Minh Ngọc, and especially my late grandmother Vũ Thị Nọt: *Bà ơi, cháu thành công rồi! Con cảm ơn bố, mẹ, ông, bà đã nuôi dạy con thành người hôm nay.* Without them, this thesis would not have been possible, nor would I have become who I am today. I also want to thank all my friends, spanning from the United Kingdom to the United States and Vietnam. While it would be exhaustive to list each one individually – because I am lucky enough to be surrounded by

so many great friends, their presence has undoubtedly made my journey much more enjoyable.

Statement of Originality

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. Except where otherwise stated, this thesis only contains my original works at the University of Oxford. The contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

A. Tuan Nguyen
University of Oxford

Abstract

The ever-evolving world we live in causes continuous data distribution shifts for any machine learning application. As machine learning models are notoriously not robust against such distribution shifts, it is of our interest to develop techniques to deal with this problem. **Hence, this thesis investigates the issue of distribution shift in machine learning, with a focus on real-world and challenging settings of this problem.**

We first revisit the typical settings of Domain Generalization (centralized multi-source domain generalization – Chapter 2) and Domain Adaptation (single-source unsupervised domain adaptation – Chapter 3). For both settings, we present our new theoretical insights into the invariant representation learning framework and develop new representation alignment techniques based on those insights. Our techniques lead to better representation alignment across domains, and thus an improved generalization ability to target distributions. We then switch our attention to more challenging settings of distribution shift. In particular, in Chapter 4, we study and incorporate the problem of Domain Generalization into the Federated Learning setting, which is more difficult than a centralized setting since Federated Learning prohibits the domains from sharing their data and information. Next, in Chapter 5, we consider the test time adaptation setting, where the unlabeled target domain data is only revealed at deployment, in the form of a stream of test data batches. This is more challenging than a typical unsupervised domain adaptation setting because the target data is only available in a small quantity at a time, and we typically do not have the target data during the source domain training nor do we have the source data during adaptation. Finally, in Chapter 6, we look at foundation vision-language models for the zero-shot classification problem, and the challenges of adapting such models from the pre-trained data distribution to a downstream distribution. While vision-language models have gained

popularity as a solution for the classification problem, typical unsupervised domain adaptation methods have proved to be ineffective for adapting such models against downstream distribution shifts. This motivates us to develop an unsupervised prompt learning method to facilitate zero-shot classification for vision-language models.

Contents

1 Introduction **1**

1.1 A brief overview of Distribution Shift and the Challenges 1

1.1.1 Existing Methods: 2

1.1.2 Challenges: 5

1.2 Contributions 6

1.3 Background 8

1.3.1 Representation Learning framework 8

1.3.2 Distribution Shift and Representation Alignment 11

1.3.3 Contrastive Vision-Language Models 14

1.4 Publications 16

1.5 Thesis Structure 17

1.6 Conclusion 18

2 Domain Invariant Representation Learning with Domain Density Transformations **19**

2.1 Introduction 19

2.2 Related Work 22

2.3 Theoretical Approach 24

2.3.1 Problem Statement 24

2.3.2 Learning a Domain-Invariant Representation with Domain Density Transformation Functions 26

2.4 A Practical Implementation using Generative Adversarial Networks . 28

2.5 Experiments 30

2.5.1 Datasets 30

2.5.2 Experimental Setting 30

2.5.3 Results 31

2.6 Conclusion 33

3 KL Guided Domain Adaptation **35**

3.1 Introduction . . . 36
3.2 Related Work . . . 38
3.3 Approach . . . 40
3.3.1 Problem Statement . . . 40
3.3.2 KL Guided Domain Adaptation . . . 43
3.3.3 Optimization . . . 47
3.4 Experiments . . . 48
3.4.1 Datasets . . . 48
3.4.2 Baselines . . . 49
3.4.3 Experimental Setting . . . 49
3.4.4 Results . . . 49
3.5 Conclusion . . . 50

4 FedSR: A Simple and Effective Domain Generalization Method for Federated Learning 52
4.1 Introduction . . . 53
4.2 Related Work . . . 54
4.2.1 Federated Learning . . . 54
4.2.2 Domain Generalization . . . 55
4.3 Approach . . . 56
4.3.1 Preliminaries . . . 56
4.3.2 Problem Setting: Representation Learning for Domain Generalization . . . 58
4.3.3 FedSR . . . 60
4.3.3.1 L2-norm Regularizer . . . 60
4.3.3.2 Conditional Mutual Information regularizer . . . 61
4.3.3.3 Implementation Choices and Optimization . . . 63
4.4 Experiments . . . 64
4.4.1 Datasets . . . 64
4.4.2 Experimental Setting . . . 64
4.4.3 Baselines . . . 66
4.4.4 Results . . . 67
4.5 Conclusion . . . 68

5 TIPI: Test Time Adaptation with Transformation Invariance 69
5.1 Introduction . . . 70
5.2 Related Works . . . 72

5.3 Approach . . . 73
5.3.1 Problem setting . . . 73
5.3.2 Test Time Domain Adaptation with Transformation Invariance 74
5.3.2.1 Invariance Regularization Technique . . . 75
5.3.2.2 Practical choice of the transformation set . . . 78
5.3.2.3 Additional Discussions . . . 78
5.4 Experiments . . . 80
5.4.1 Evaluation Protocol and Baselines: . . . 80
5.4.2 Robustness to Image Corruptions . . . 81
5.4.2.1 Experimental Settings . . . 81
5.4.2.2 Results . . . 83
5.4.3 Domain Adaptation . . . 84
5.4.3.1 Experimental Settings . . . 84
5.4.3.2 Results . . . 85
5.4.4 Runtime and memory consumption . . . 85
5.5 Conclusion . . . 86

6 uCAP: An Unsupervised Prompting Method for Vision-Language Models 87
6.1 Introduction . . . 88
6.2 Related Work . . . 90
6.3 Approach: uCAP . . . 91
6.4 Experiments . . . 95
6.4.1 Experiment Settings . . . 96
6.4.2 Main Results . . . 98
6.4.3 Additional Results and Discussion . . . 98
6.5 Conclusion . . . 102

7 Conclusions 103
7.1 Concluding Remarks . . . 103
7.2 Future Vision . . . 103

References 117

Appendix A Domain Invariant Representation Learning with Domain Density Transformations 118

Appendix B KL Guided Domain Adaptation 122

Appendix C FedSR: A Simple and Effective Domain Generalization Method for Federated Learning **137**

Appendix D TIPI: Test Time Adaptation with Transformation Invariance **142**

Appendix E uCAP: An Unsupervised Prompting Method for Vision-Language Models **146**

Appendix F Publications and Contributions List **148**

List of Figures

1.1 Graphical Model of the Representation Learning framework 9
1.2 Graphical model for Unsupervised Domain Adaptation with Representation Learning 12
1.3 An example of two domains where marginal alignment is not sufficient 13

2.1 An example of two domains where marginal alignment is not sufficient 21
2.2 Graphical Model of the Domain Generalization problem 25
2.3 Domain density transformation 27
2.4 Visualization of the representation space 32

3.1 Graphical model for Unsupervised Domain Adaptation with Representation Learning 41
3.2 Benefits of Reverse KL for representation alignment 47

6.1 Class Description Alignment for Contrastive Vision-Language Models 88
6.2 Graphical Model of uCAP 91
6.3 uCAP performance with a varying number of prompts 99

B.1 Visualization of the representation of KL, MMD, DANN, and ERM . 129

C.1 Visualization of the representation space of FedSR and FedAVG . . . 141

Chapter 1

Introduction

1.1 A brief overview of Distribution Shift and the Challenges

Can a car detection network trained solely in urban areas of developed countries perform well in rural areas of other countries? What if a classification model trained on black-and-white images is later tested on colored images? The ability to generalize to out-of-distribution data, while seeming natural and innate to humans, is a concept that machine learning and deep learning models struggle with [Wang and Deng, 2018]. Indeed, considering the above example, a human after being taught to differentiate between two types of car in black-and-white images can generalize well to colored images, while a machine learning model usually struggles to do so. This is because, intuitively, without any prior knowledge about the colored image distribution and its relation to black-and-white images, the machine learning model is not guaranteed to (and often does not) extrapolate to the unseen target domain of colored images.

Distribution shift refers to the problem where the testing data distribution is shifted from the training distribution. Specifically, if we have a training data distribution $p_S(x, y)$ and a target (testing) distribution $p_T(x, y)$, where $x \in \mathcal{X}$ is the input and $y \in \mathcal{Y}$ is the label, the distribution shift problem occurs when $p_S(x, y) \neq p_T(x, y)$. This happens in virtually any real-world applications of machine learning. For example, it can be attributed to degradation in the input sensor system (a camera getting old), malicious attacks from users (adding noises to input images), or simply just the inherent continuous shift in the data distribution due to our ever-changing world. However, most machine learning algorithms assume an over-simplified assumption of independent and identically distributed (i.i.d.) data points for the training and test datasets (presumably because this simplified setting offers formulation simplicity

and a theoretical learning guarantee), which are generally not true for any real-world applications. Therefore, they typically struggle to generalize to a shifted testing distribution, like in the above examples. Empirically, it has been shown that machine learning models are notoriously not robust under even a slight distribution shift, e.g., with common image corruptions (JPEG compression loss, Gaussian blur, etc.) [Hendrycks and Dietterich, 2019], with small adversarial noises [Szegedy et al., 2014], or from simulation to real-world images [Peng et al., 2017]. This renders machine learning models unreliable and unsafe when deployed into real-world environments, especially for safety-critical applications such as autonomous vehicles and medical diagnosis systems.

Therefore, distributional robustness is an important research topic and has been extensively studied.

1.1.1 Existing Methods:

In this subsection, we very briefly introduce the related works to give a better context of how this thesis is positioned.

There are two schools of thought to tackle the distribution shift problem, namely the proactive and reactive approaches. The former aims to learn a robust machine learning model that we expect to generalize better to unseen target distributions; while the latter reactively adapts the model to a target distribution when it is revealed.

Proactive approaches:

The idea of proactively learning a robust and generalizable model can be traced back to the very early days of machine learning. For example, regularizers such as LASSO (l_1 norm of the parameters) [Tibshirani, 1996] are often used for classical machine learning models to limit the model's capacity, avoid overfitting, and hope for better generalization at test time. In the context of distributional robustness, there is a wide spectrum of proactive approaches, targeting network architecture, optimization, and learning methodologies. These proactive approaches can be thought of as to introduce some inductive biases into the model and/or its training process. As a result, the final model would depend on not only the training data, but also this sets of inductive biases. By tailoring these inductive biases for the generalization problem (often using some prior domain knowledge), we are more likely to achieve a robust model under distribution shift. For example, Zhang [2019] introduces anti-aliasing filters

into deep neural networks and shows that they can help improve the performance and generalization ability of such networks. Later, Vasconcelos et al. [2021] also confirm that anti-aliasing filters improve out-of-distribution robustness, e.g., for image corruptions or natural distribution shift. There are also many methods to improve the optimization process of the training objective, aiming for better generalization. For instance, Foret et al. [2021b] propose Sharpness-Aware Minimization (SAM) to find flat minima of the loss landscape, which leads to better generalization. Möllenhoff and Khan [2023] later interpret SAM as an optimal relaxation of the Bayes objective, which opens a Bayesian view for robust model training. However, despite efforts to improve the model architecture and training recipe for better generalization performance, the most effective approach is still to collect data from different distributions and develop methods to learn a generalizable machine learning model using these source datasets – the so-called Domain Generalization training setting [Blanchard et al., 2011, Muandet et al., 2013].

Domain Generalization: The idea is quite simple: we need to collect more training data distributions in order to generalize well to unseen target distributions, just like, in a classical machine learning setting, we need to collect more data points to generalize better to unseen data points. Therefore, we have seen a great amount of attention and research into the domain generalization problem. For instance, Muandet et al. [2013], Li et al. [2018c], and Li et al. [2018d] propose to learn a domain-invariant representation for better generalization across domains; Du et al. [2020], Balaji et al. [2018], and Li et al. [2018a] leverage the idea of meta-learning, hoping a network that can adapt well within the source domains will adapt better to the target domains; while Ding and Fu [2017], Chattopadhyay et al. [2020], and Seo et al. [2019] propose to learn domain-specific information to aid the domain-invariant representation in the prediction task of each domain.

Federated Domain Generalization: is a specific scenario of Domain Generalization where we tackle this problem in a federated learning setting. Federated Learning (FL) [McMahan et al., 2017, Stich, 2019, Wang and Joshi, 2021] refers to the decentralized and privacy-preserving machine learning framework in which multiple clients collaborate to train a global model without sharing their data. Addressing the domain generalization problem within this framework has proven to be more challenging than in the typical centralized setting, as data sharing among domains is not permitted. There have been some early attempts to incorporate domain generalization into the FL framework. For example, Liu et al. [2021a] propose a technique tailored for medical image segmentation which allows the clients to share their data in the frequency space

with each other. This can be viewed as a form of data leakage and is undesirable in the FL setting we consider. Meanwhile, Zhang et al. [2021] align the representation distribution across domains via a reference distribution (from a generative model). While this does fully preserve data privacy, it can be overly complicated to implement in practice, and the performance is not yet on par with SOTA centralized methods. Chapter 4 will dive into and tackle this challenging and relatively underexplored scenario of domain generalization.

Reactive approaches:

On the other hand, reactive approaches leverage information from the target domain to adapt the source model so that it performs well in the target environment. Hence, this setting is also called domain adaptation. This is only possible if/when the target domain is revealed.

Domain Adaptation: There are two different but related research interests, namely to develop the generalization theory and practical methodologies, for the domain adaptation problem. Theory for the generalization of machine learning models under distribution shift has been a crowded research topic for a long time [Ben-David et al., 2010, Mansour et al., 2009, Acuna et al., 2021, Cortes et al., 2010, Johansson et al., 2019, Azizzadenesheli et al., 2019]. There are several bounds of the target loss in the domain adaptation problem [Ben-David et al., 2010, Mansour et al., 2009]. These bounds are often conceptually similar: they bound the target loss by the training loss and an additional generalization gap caused by the distribution mismatch (for which different works use different distances/divergences to measure). As a result, they all consist of the training loss, a marginal misalignment term (of the input distribution) and a conditional misalignment term (of the label distribution given the input) between the source and target data distributions. Regarding methodologies for the domain adaptation problem, the most straightforward approach is perhaps to collect data and annotations from the target domain and finetune the source model with this labeled dataset. However, annotating a dataset is often time-consuming and costly. Therefore, we are more interested in unsupervised or semi-supervised domain adaptation. Motivated by the above generalization bounds, a popular method for unsupervised domain adaptation is to learn a representation of the input data whose marginal distribution is aligned across the domains [Ganin et al., 2016, Li et al., 2018c, Shen et al., 2018, Sun and Saenko, 2016]. This will reduce the theoretical generalization gap and often empirically lead to better predictive performance in

the target domain. However, this approach is only feasible in the classical setting of domain adaptation, where the labeled source data and the unlabeled target data are both available during training.

Test Time Adaptation: is a newer framework where the target data is unavailable during source model training, and the source data is inaccessible during adaptation, thereby preserving data privacy. In particular, the model only has access to the target data when deployed in the target environment, as the stream of small test data batches, making the adaptation task extremely challenging. Due to this characteristic, test time adaptation is considered more practical and has gained popularity compared to the classical setting of domain adaptation mentioned earlier. So far, there are two main lines of work dealing with this test time adaptation problem. The first approach is based on the batch normalization layer in the network, with the idea of using the test-time batch statistics for these layers instead of the source statistics [Schneider et al., 2020, Nado et al., 2020]. This can partially help with the representation misalignment between the source and target distributions but is only applicable when the network has BatchNorm layers and the batch size is large. The other line of work is to use an unsupervised surrogate loss to "refine" the model on the target data. For example, TENT [Wang et al., 2021] minimizes the prediction entropy of the model on the target data. However, as pointed out by its authors, TENT needs a large batch size to avoid collapsing to a trivial prediction (which outputs only one class), due to the characteristics of this surrogate objective. Meanwhile, TTT [Sun et al., 2020] and TTT+++ [Liu et al., 2021b] train a self-supervised task alongside the main task on the source domain, and continue to train the self-supervised task during the online adaptation phase. As such, TTT and TTT++ are not suitable for all off-the-shelf machine learning models, since they require modifications to the training phase of the source domain (to incorporate the self-supervised task).

Both proactive and reactive approaches are often used together to create a robust model that can generalize better to unseen target domains, and to actively adapt this model for a target environment when necessary. For this reason, this thesis will focus on both the domain generalization and domain adaptation settings, since they are the most effective and predominant for the above two groups of approaches.

1.1.2 Challenges:

Although distributional robustness has been extensively studied, there are still open challenges for the domain generalization and domain adaptation problems:

- **Incomplete theory for distributional robustness, especially in the context of domain generalization and domain adaptation.** For example, it was previously unknown if the conditional misalignment in the representation space is bounded in the domain adaptation problem, and it is not clear when a marginally- and conditionally-aligned representation exists and how to learn one in the domain generalization problem.
- **Limited research and methodologies for real-world settings of distribution shift.** While the classical settings of domain generalization and domain adaptation are relatively well-studied, they are not suitable for some real-world applications. For example, test time domain adaptation and federated learning for domain generalization have gained interest as solutions for the distribution shift problem in real-world scenarios, as they emphasize and respect data privacy. These settings are relatively understudied in the literature.
- **The difficulties of applying existing works for foundation models.** Foundation models such as large vision-language models have emerged as powerful tools for many machine problems. These large models, which are typically trained on a huge amount of data, exhibit remarkable robustness and generalization ability. However, when needing to adapt such models to a downstream task, existing domain adaptation methods are not suitable, since they often require modifying and refining the models, which is expensive and often destroys the vast amount of information the models have learned.

1.2 Contributions

This thesis investigates different settings to tackle the distribution shift problem, aiming to improve machine learning models' performance on target environments at deployment. We emphasize the importance of distributional robustness in model development and deployment, and that both proactive and reactive approaches should be taken into account to develop and maintain a robust model. However, in this thesis, we isolate these settings and study them individually. In particular, the contributions of each technical chapter are as follows:

Chapter 2 – Domain Invariant Representation Learning with Domain Density Transformations: We consider the typical setting of centralized multi-source domain generalization, with the representation learning framework. We first provide some insight into domain-invariant representation learning with an observation about the necessary and sufficient condition of the existence of an invariant representation. We then propose a theoretically grounded method to learn a domain invariant representation, based on the density transformation functions across domains. Remarkably, our method can align both the representation's marginal and conditional distributions simultaneously. We then provide a practical implementation of our method, termed DIRT-GAN, that uses Generative Adversarial Networks to learn the density transformation functions. DIRT-GAN achieves preferable predictive performance and representation alignment effect when compared to existing domain invariant representation learning methods.

Chapter 2 – KL Guided Domain Adaptation: We revisit the classical setting of single-source unsupervised domain adaptation. We first provide a new bound on the source-to-target domain generalization gap. This bound consists of a marginal misalignment term and a conditional misalignment term, which are both measured by the reverse KL divergence. We further show that the conditional misalignment in the representation space is bounded by the conditional misalignment in the input space, which is an improvement over existing generalization analyses. Based on this bound, we propose a new marginal representation alignment technique, using the reverse KL divergence. Finally, we illustrate that the reverse KL is more efficient and stable to compute (when compared to existing techniques), while having preferable properties for the representation alignment task (zero-forcing and mode-seeking). As a result, our method shows superior performance compared to existing marginal alignment techniques for the domain adaptation task.

Chapter 4 – FedSR: A Simple and Effective Domain Generalization Method for Federated Learning: In this chapter, we explore the challenges of domain generalization in the federated learning setting. Motivated by such challenges, we propose a simple method, termed FedSR, that learns a simple representation of the data for a better generalization ability. In particular, FedSR employs two regularizers on the representation, namely an L2-norm regularizer and a conditional mutual information (between the input and the representation given the label) regularizer. We further show that the L2-norm regularizer acts as an implicit form of marginal representation

alignment, while the conditional mutual information regularizer acts as an implicit form of conditional representation alignment. FedSR fully respects the data privacy nature of federated learning, while outperforming existing federated learning methods in the domain generalization problem and performing on par with centralized domain generalization methods.

Chapter 5 – TIPI: Test Time Adaptation with Transformation Invariance: We introduce TIPI, a fully test time adaptation method for the unsupervised domain adaptation problem. Motivated by a generalization bound under transformation functions, we propose to find transformations that can simulate real-world source-target distribution shifts, and enforce the prediction network to be invariant under such transformations, in the form of an invariance regularizer. We opt to use l_∞ adversarial noises as the transformations since they can simulate well most distribution shifts in practice. Finally, we show in our experiments that TIPI achieves competitive performance with existing works while being extremely robust with small batch sizes.

Chapter 6 – uCAP: An Unsupervised Prompting Method for Vision-Language Models: In this chapter, we look at the domain adaptation challenges of foundation vision-language models (VLMs) for the zero-shot classification task. To address VLMs' limitation of needing domain-specific prompts to adapt to a downstream domain, we propose uCAP, a method to learn the domain prompts automatically without labeled data. Towards this end, we model the generation of images given the class names and a domain-specific prompt with an unsupervised likelihood distribution, and then perform Bayesian inference of the prompts. Our unsupervised prompt learning method achieves state-of-the-art performance across a wide range of models and datasets.

1.3 Background

In this section, we provide some backgrounds to make following the thesis easier.

1.3.1 Representation Learning framework

Representation learning is a popular framework that empowers many domain generalization and domain adaptation works [Ganin et al., 2016, Li et al., 2018b, Sun and Saenko, 2016]. The idea behind these works is intuitive and straightforward:

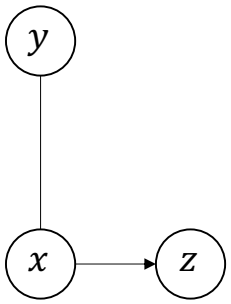


Figure 1.1: **Graphical Model of the Representation Learning framework**.

given that the data distribution changes across domains, it is of our interest to find a representation of the data whose distribution is more aligned across all domains. We first cover the general representation framework here, and subsequently cover representation alignment in the next subsection.

First, assume that we have a data distribution $p(x, y)$ where $x \in \mathcal{X}$ is the input data (covariate) and $y \in \mathcal{Y}$ is the label. For the ease of notation, we assume x and y are continuous in this section. However, note that the discussion here is unchanged if x or y or both are discrete variables (simply replace the corresponding probability density functions and integral signs with the probability mass functions and summation signs). Note also that the discussion in this section hold for any the causal relationship between x and y.

In a representation learning framework, we aim to learn a representation $z \in \mathcal{Z}$ of x with a mapping $p(z|x)$. This mapping can be either deterministic (i.e., $p(z|x) = \delta_{g(x)}(z)$ where $g : \mathcal{X} \to \mathcal{Z}$ is a deterministic function of x and δ is the Dirac delta function) or stochastic (e.g., it can be a Gaussian distribution $p(z|x) = \mathcal{N}(z; \mu(x), \Sigma(x))$).

The joint distribution of x, y, and z is:

$$p(x, y, z) = p(x, y)p(z|x).$$

The graphical model describing this representation learning framework is depicted in Figure 1.1.

From z, we often learn a predictive distribution $\hat{p}(y|z)$ to predict y. The predictive distribution of y given x is thus:

$$\hat{p}(y|x) = \mathbb{E}_{p(z|x)} \left[\hat{p}(y|z) \right].$$

In the context of deep learning, both $p(z|x)$ and $\hat{p}(y|z)$ are parameterized by neural networks, and they are optimized jointly so that the predictive distribution $\hat{p}(y|x)$

minimize some loss function of choice. For example, if we use the typical cross-entropy loss, the loss function would be:

$$\ell = \mathbb{E}_{p(x)} \left[-\log \mathbb{E}_{p(z|x)} \left[\hat{p}(y|z) \right] \right] \tag{1.1}$$

Mutual Information: A quantity of interest in the representation learning framework is the mutual information between z and x, denoted by $I(z, x)$. In particular:

$$I(z, x) = \mathbb{E}_{p(z,x)} \left[\log \frac{p(z, x)}{p(z)p(x)} \right] = \mathrm{KL} \left[p(z, x) | (p(z)p(x)) \right], \tag{1.2}$$

This is a non-negative quantity indicating the mutual information between z and x. When $I(z, x) = 0$, we have $p(z, x) = p(z)p(x)$, meaning that z and x are independent, thus sharing no mutual information (such representation is not good for the prediction task).

The mutual information among variables is often used in the representation learning framework (e.g., [Alemi et al., 2017, Tishby et al., 2000]).

Data Processing Inequality: states that, for any prediction task of label y, no processing of x can increase the information that x contains about y. Simply put, z always has less information about y than x does. Formally, the theorem is stated as:

Theorem 1.1 (Data processing inequality)**.**

$$I(x, y) \geq I(z, y).$$

While there are many proofs of the Data processing inequality, we present one that is based only on the definition in Eq. 1.2.

Proof.

$$
\begin{aligned}
& I(x,y) - I(z,y) \\
&= \mathbb{E}_{p(x,y)}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] - \mathbb{E}_{p(z,y)}\left[\log \frac{p(z,y)}{p(z)p(y)}\right] \\
&= \mathbb{E}_{p(x,z,y)}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] - \mathbb{E}_{p(x,z,y)}\left[\log \frac{p(z,y)}{p(z)p(y)}\right] \\
&= \mathbb{E}_{p(x,z,y)}\left[\log \frac{p(x,y)}{p(x)p(y)} - \log \frac{p(z,y)}{p(z)p(y)}\right] \\
&= \mathbb{E}_{p(x,z,y)}\left[\log \frac{p(x,y)}{p(x)} - \log \frac{p(z,y)}{p(z)}\right] \\
&= \mathbb{E}_{p(x,z,y)}\left[\log p(y|x) - \log p(y|z)\right] \\
&= \mathbb{E}_{p(x,z)}\left[\mathbb{E}_{p(y|x,z)}\left[\log p(y|x) - \log p(y|z)\right]\right] \\
&= \mathbb{E}_{p(x,z)}\left[\mathbb{E}_{p(y|x)}\left[\log p(y|x) - \log p(y|z)\right]\right] \\
&\qquad \text{(since } p(y|x,z) = p(y|x) \text{ – } y \text{ and } z \text{ are conditionally independent given } x\text{)} \\
&= \mathbb{E}_{p(x,z)}\left[\mathrm{KL}\left[p(y|x)|p(y|z)\right]\right] \\
&\geq 0
\end{aligned}
$$

□

When $I(z,y) = I(x,y)$, meaning z contains the maximum amount of information about y, the representation z is sufficient for this prediction task. This is thus often called the sufficiency condition.

1.3.2 Distribution Shift and Representation Alignment

In this subsection, we introduce briefly the domain generalization and domain adaptation settings, and a common approach, namely representation alignment, for both problems.

Assuming we have a source data distribution $p_S(x,y)$ and a target data distribution $p_T(x,y)$. In cases where we have multiple source domains $S_1, S_2, ..., S_K$, p_S represents a combination of the individual source domain data distributions, e.g., $p_S(x,y) = \frac{1}{K}\sum_{i=1}^{K} p_{S_i}(x,y)$. The distribution shift problem arises when $p_T(x,y) \neq p_S(x,y)$, and in such a scenario, a model trained solely with the source data is not guaranteed to perform well in the target environment.

In the domain generalization setting, the target distribution p_T is often unknown, and the goal is to develop a robust model capable of performing well in different target

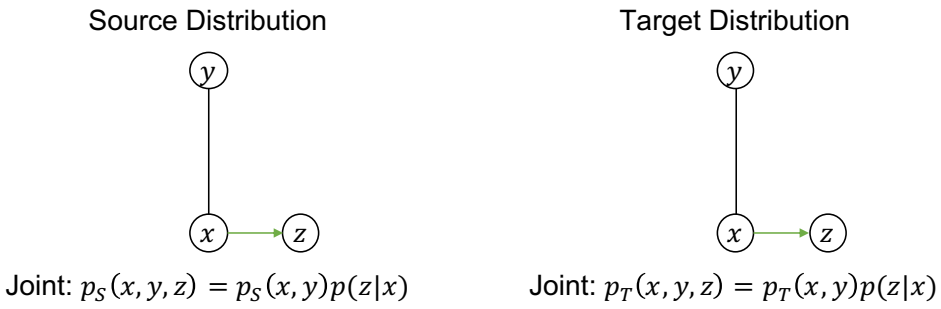


Figure 1.2: **Graphical model**. Representation learning for the distribution shift problem. We aim to learn a representation z of x such that $p_S(z, y) \approx p_T(z, y)$. Note that the representation mapping $p(z|x)$ (green edge) is shared between the source and target domains.

environments. This necessitates collecting multiple source domains for the network to learn to generalize to unseen target domains, a scenario known as multi-source domain generalization. Meanwhile, in domain adaptation, we possess some information about the target distribution, typically in the form of unlabeled data (unsupervised domain adaptation) or a limited amount of labeled data (few-shot or semi-supervised domain adaptation).

Since $p_S(x, y) \neq p_T(x, y)$, it is natural to learn a representation z of x via the representation learning framework such that $p_S(z, y) \approx p_T(z, y)$. Figure 1.2 depicts this process. Note that the representation mapping $p(z|x)$ is shared between the source and target distributions in this case. According to the loss function in Eq. 1.1, the training loss and test loss are:

$$\ell_{train} = \mathbb{E}_{p_S(x)} \left[-\log \mathbb{E}_{p(z|x)} \left[\hat{p}(y|z) \right] \right], \qquad \ell_{test} = \mathbb{E}_{p_T(x)} \left[-\log \mathbb{E}_{p(z|x)} \left[\hat{p}(y|z) \right] \right].$$

During training, we often sample only one z per x, leading to the training objective:

$$\overline{\ell_{train}} = \mathbb{E}_{p_S(z,x)} \left[-\log \hat{p}(y|z) \right].$$

If $p_S(z, y)$ and $p_T(z, y)$ are perfectly aligned, i.e., $p_S(z, y) = p_T(z, y)\ \forall z \in \mathcal{Z}, y \in \mathcal{Y}$, we have:

$$\begin{aligned} \ell_{test} &\leq \mathbb{E}_{p_T(z,x)} \left[-\log \hat{p}(y|z) \right] \quad \text{(due to Jensen Inequality)} \\ &= \overline{\ell_{train}}, \end{aligned}$$

meaning that we have a perfect generalization from the training objective to the target distribution. We will also see later in Chapter 3 that, even if $p_S(z, y)$ and $p_T(z, y)$

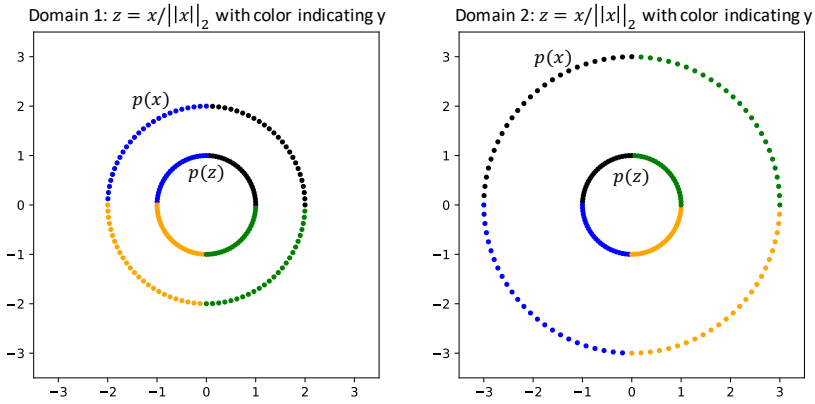


Figure 1.3: **An example of two domains**. For each domain, x is uniformly distributed on the outer circle (radius 2 for domain 1 and radius 3 for domain 2), with the color indicating class label y. After the transformation $z = x/||x||_2$, the marginal of z is aligned (uniformly distributed on the unit circle for both domains), but the conditional $p(y|z)$ is not aligned. Thus, using this representation for predicting y would not generalize well across domains.

are only close (and not strictly equal), we still have a generalization guarantee of the model to the target distribution.

To align the joint distribution of z and y means to align the marginal distribution of z and the conditional distribution of y given z across the domains. This can be viewed as to "explain away" the domain, resulting in a domain-invariant representation. These two types of alignment are termed marginal alignment and conditional alignment [Tanwani, 2020], and are both important for the domain generalization and domain adaptation problems. For instance, In Figure 1.3, we illustrate an example where the representation z satisfies the marginal alignment but not the conditional alignment. Specifically, x is distributed uniformly on the circle with radius 2 (and centered at the origin) for domain 1 and distributed uniformly on the circle with radius 3 (centered at the origin) for domain 2. The representation z defined by the mapping $z = g(x) = x/||x||_2$ will align the marginal distribution $p(z)$, i.e., z is now distributed uniformly on the unit circle for both domains. However, the conditional distribution $p(y|z)$ is not aligned between the two domains (y is represented by color), which means using this representation for classification is suboptimal, and in this extreme case would lead to 0% accuracy in the target domain 2. This is an extreme case of misalignment but it does illustrate the importance of the two types of alignment.

In the multi-source domain generalization setting, since there are multiple source domains, a common approach is to align both the marginal and the conditional distributions, and hope that this alignment also generalize to unseen target distributions [Li et al., 2018c,d, Zhao et al., 2020]. On the other hand, in unsupervised domain

adaptation, one often focuses only on marginal distribution alignment since unlabeled data is available from the target distribution [Ganin et al., 2016, Shen et al., 2018]. As such, this also necessitates bounding the misalignment of the conditional distribution of the label y given the representation z for a sound learning process [Nguyen et al., 2022b].

1.3.3 Contrastive Vision-Language Models

In this subsection, we review the family of contrastive vision-language models. These models are the main focus of Chapter 6, where we explore strategies to adapt a foundation vision-language model to a downstream distribution. Unlike the previously mentioned (and common) discriminative models, contrastive vision-language models are trained to identify similarities between the vision and natural language modalities using a contrastive loss function. These models are typically pre-trained with large datasets covering various image sub-domains, making them robust to a wider range of downstream data distributions. Therefore, such models serve as excellent candidates for alleviating distribution shift in classification settings.

[Radford et al., 2021] is one of the first works in contrastive multi-modal pre-training, which popularizes this family of models. Contrastive multi-modal pre-training can be applied to any pair or set of modalities such as natural language, image, audio, and video. More specifically for vision-language pre-training, we need to prepare a dataset of paired input, usually in the forms of images and their corresponding captions.

The architecture of vision-language models consists of two main parts: a vision encoder f_θ and a text encoder f_ϕ. The vision encoder is often a ResNet [He et al., 2016] or a VisionTransformer [Dosovitskiy et al., 2021], while the text encoder usually has the Transformer architecture [Vaswani et al., 2017]. Since f_θ and f_ϕ produce modality-specific representations, we also have the projection layers to transform such representations into a single multi-modal embedding space: g_θ for the vision modality and g_ϕ for the language modality. These so-called projectors are often just a linear layer that transforms the representations into fixed-dimension embedding.

Given an image x and a caption t, their similarity score is measured by the cosine similarity S_C in the embedding space:

$$s_{\theta,\phi}(x,t) = S_C\left(g_\theta(f_\theta(x)), g_\phi(f_\phi(t))\right) = \frac{g_\theta(f_\theta(x)).g_\phi(f_\phi(t))}{||g_\theta(f_\theta(x))||_2.||g_\phi(f_\phi(t))||_2},$$

where . is the dot product and $||v||_2$ denote the l_2 norm of a vector v.

Pre-training: Given the paired image-caption dataset, at each pre-training iteration, we select a minibatch $\{(x_1, t_1), (x_2, t_2), ..., (x_B, t_B)\}$ for optimization. The goal is to maximize the similarity between the paired image and caption (i.e., x_i and t_i) and minimize the similarity between unpaired ones (i.e., x_i and t_j for $i \neq j$). We often use the temperature-scaled cross-entropy loss. Specifically, with a temperature τ, the objective is:

$$\min_{\theta,\phi} \frac{1}{2B} \sum_{i=1}^{B} \left[-\log \frac{e^{s_{\theta,\phi}(x_i,t_i)/\tau}}{\sum_{j=1}^{B} e^{s_{\theta,\phi}(x_i,t_j)/\tau}} - \log \frac{e^{s_{\theta,\phi}(x_i,t_i)/\tau}}{\sum_{j=1}^{B} e^{s_{\theta,\phi}(x_j,t_i)/\tau}} \right]$$

As a result of this pre-training objective, an image and a piece of text that describe the same concept will have a high similarity score in the embedding space, and vice versa, the ones that describe different concepts will have a lower similarity score.

Inference – Zero-shot classification: Vision-Language models excel at the zero-shot classification task [Radford et al., 2021, Li et al., 2023b, Cherti et al., 2023]. This is because such models have already learned to identify similarities between the vision and natural language modalities during the pre-training phase. Consequently, when provided with only class names or descriptions, they can classify images into their corresponding classes based on this notion of similarity. Zero-shot classification has gained popularity as it does not require domain-specific supervised training, and is robust to the label shift problem (e.g., continually adding and removing certain classes). Here, we provide a simplified overview of performing zero-shot classification with contrastive vision language models, while in Chapter 6, we discuss how to additionally adapt such models to downstream data distributions when necessary.

Given a classification task, we first need to form a class description for each class. These descriptions are often based on domain knowledge and/or the trial-and-error method. For example, the descriptions can be "a photo of a cat" and "a photo of a dog". After collecting the class descriptions $c_1, c_2, ..., c_K$, we compute the class embeddings with $e_i = g_\phi(f_\phi(c_i)), \forall 1 \leq i \leq K$. At inference time, when receiving an image x for classification, the predicted class would be:

$$\arg \max_{1 \leq i \leq K} S_C \left(g_\theta(f_\theta(x)), e_i \right),$$

where S_C is, again, the cosine similarity.

Note that, in practice, we often have multiple descriptions per class. In that case, common practice is to average the text embeddings of the class descriptions to form the class embedding.

1.4 Publications

Each chapter of this integrated thesis is based on a published research paper. These papers are listed below, and the detailed author contributions can be found in Appendix F.

Chapter 2:

> **A. Tuan Nguyen**, Toan Tran, Yarin Gal, and Atılım Güneş Baydin. Domain Invariant Representation Learning with Domain Density Transformations. *NeurIPS*, 2021.

Chapter 3:

> **A. Tuan Nguyen**, Toan Tran, Yarin Gal, Philip H. S. Torr, and Atılım Güneş Baydin. KL Guided Domain Adaptation. *ICLR*, 2022.

Chapter 4:

> **A. Tuan Nguyen**, Philip Torr, and Ser-Nam Lim. FedSR: A Simple and Effective Domain Generalization Method for Federated Learning. *NeurIPS*, 2022.

Chapter 5:

> **A. Tuan Nguyen**, Thanh Nguyen-Tang, Ser-Nam Lim, and Philip Torr. TIPI: Test Time Adaptation with Transformation Invariance. *CVPR*, 2023.

Chapter 6:

> **A. Tuan Nguyen**, Kai Sheng Tai, Bor-Chun Chen, Satya Narayan Shukla, Hanchao Yu, Philip Torr, Taipeng Tian, and Ser-Nam Lim. uCAP: An Unsupervised Prompting Method for Vision-Language Models. *ECCV*, 2024.

1.5 Thesis Structure

The remainder of this thesis is organized as follows: Chapters 2-6 present our technical papers, and the supplementary materials of the papers are included in the Appendices. In particular:

- **Chapter 2 (with Appendix A) – Domain Invariant Representation Learning with Domain Density Transformations:** explores the centralized multi-source domain generalization problem with several theoretical insights into the invariant representation learning framework and introduces DIRT – a method to simultaneously align the marginal and conditional distributions of a representation, leading to better generalization performance.
- **Chapter 3 (with Appendix B) – KL Guided Domain Adaptation:** revisits the single-source unsupervised domain adaptation with several novel theoretical analyses and proposes a new representation alignment technique based on the reverse KL divergence.
- **Chapter 4 (with Appendix C) – FedSR: A Simple and Effective Domain Generalization Method for Federated Learning:** explores the challenges of domain generalization in the federated learning setting and proposes FedSR, a simple method for domain generalization that fully respect the data privacy requirement of federated learning.
- **Chapter 5 (with Appendix D) – TIPI: Test Time Adaptation with Transformation Invariance:** tackles the problem of test time domain adaptation with TIPI, a robust and fully test time adaptation method for off-the-shelf machine learning models.
- **Chapter 6 (with Appendix E) – uCAP: An Unsupervised Prompting Method for Vision-Language Models:** introduces uCAP, an automatic and unsupervised prompt learning method for the zero-shot classification setting of vision-language models.

Finally, Chapter 7 concludes the thesis with some discussions and concluding remarks.

1.6 Conclusion

Distribution shift poses a significant challenge in machine learning, often rendering models inaccurate and unreliable in practical applications. Therefore, it is crucial to prioritize the robustness of models across all stages of development, deployment, and maintenance, ensuring their safe integration into real-world scenarios. A harmonious combination of proactive and reactive approaches is essential: proactive measures aim to learn robust models, facilitating easier adaptation to target environments, while reactive strategies are employed to actively adjust models to the target distribution when proactive measures prove insufficient. Hence, exploring different settings and strategies for tackling the distribution shift problem is important, and this thesis delves into each individual setting. Ultimately, we aim for this thesis to provide the necessary tools and motivation to effectively address the challenge of distribution shift.

Chapter 2

Domain Invariant Representation Learning with Domain Density Transformations

Abstract

Domain generalization refers to the problem where we aim to train a model on data from a set of source domains so that the model can generalize to unseen target domains. Naively training a model on the aggregate set of data (pooled from all source domains) has been shown to perform suboptimally, since the information learned by that model might be domain-specific and generalize imperfectly to target domains. To tackle this problem, a predominant domain generalization approach is to learn some domain-invariant information for the prediction task, aiming at a good generalization across domains. In this work, we propose a theoretically grounded method to learn a domain-invariant representation by enforcing the representation network to be invariant under all transformation functions among domains. We next introduce the use of generative adversarial networks to learn such domain transformations in a possible implementation of our method in practice. We demonstrate the effectiveness of our method on several widely used datasets for the domain generalization problem, on all of which we achieve competitive results with state-of-the-art models.

2.1 Introduction

Domain generalization refers to the machine learning scenario where a model is trained on multiple source domains so that it is expected to generalize well to unseen

target domains. The key difference between domain generalization [Khosla et al., 2012, Muandet et al., 2013, Ghifary et al., 2015] and domain adaptation [Zhao et al., 2019, Zhang et al., 2019, Combes et al., 2020, Tanwani, 2020] is that, in domain generalization, the learner does not have access to data of the target domain, making the problem much more challenging. One of the most common domain generalization approaches is to learn an invariant representation across domains, aiming at a good generalization performance on target domains. For instance, in the representation learning framework, the prediction function $f(x) \approx y$, where x is input data and y is a label, is obtained as a composition $h \circ g(x)$ of a deep representation network $z = g(x)$, where z is a learned representation of data x, and a smaller classifier $h(z)$, predicting label y given representation z, both of which are shared across domains. With this framework, we can aim to learn an "invariant" representation z across the source domains with the "hope" of a better generalization to the target domain.

Most existing "domain-invariance"-based methods in domain generalization focus on the marginal distribution alignment [Muandet et al., 2013, Ajakan et al., 2014, Sun and Saenko, 2016, Shen et al., 2018, Li et al., 2018b], which are still prone to distribution shifts when the conditional data distribution is not stable. In particular, the marginal alignment refers to making the representation distribution $p(z)$ to be the same across domains. This is essential since if $p(z)$ for the target domain is different from that of source domains, the classification network $h(z)$ would face out-of-distribution data at test time. Conditional alignment refers to aligning the conditional distribution of the label given the representation $p(y|z)$ to expect that the classification network (trained on the source domains) would give accurate predictions at test time. The formal definitions of these two types of alignment are discussed in Section 2.3.

In Figure 2.1, we illustrate an example where the representation z satisfies the marginal alignment but not the conditional alignment. Specifically, x is distributed uniformly on the circle with radius 2 (and centered at the origin) for domain 1 and distributed uniformly on the circle with radius 3 (centered at the origin) for domain 2. The representation z defined by the mapping $z = g(x) = x/||x||_2$ will align the marginal distribution $p(z)$, i.e., z is now distributed uniformly on the unit circle for both domains. However, the conditional distribution $p(y|z)$ is not aligned between the two domains (y is represented by color), which means using this representation for classification is suboptimal, and in this extreme case would lead to 0% accuracy in the target domain 2. This is an extreme case of misalignment but it does illustrate the importance of the conditional alignment. Therefore, we need to align both the marginal and the conditional distributions for a domain-invariant representation.

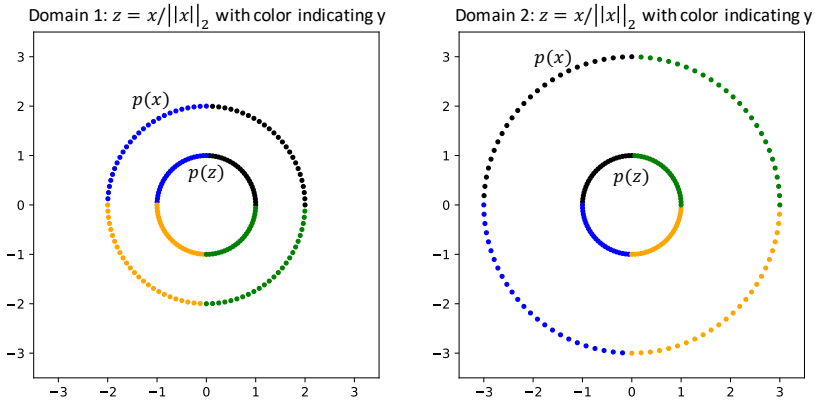


Figure 2.1: **An example of two domains**. For each domain, x is uniformly distributed on the outer circle (radius 2 for domain 1 and radius 3 for domain 2), with the color indicating class label y. After the transformation $z = x/||x||_2$, the marginal of z is aligned (uniformly distributed on the unit circle for both domains), but the conditional $p(y|z)$ is not aligned. Thus, using this representation for predicting y would not generalize well across domains.

Recently, there have been several attempts [Li et al., 2018c,d, Zhao et al., 2020] to align the joint distribution of the representation and the label $p(y, z)$ in a domain generalization problem by aligning the distribution of z across domains for each class, i.e., $p(z|y)$ (given that the label distribution $p(y)$ is unchanged across domains). However, the key drawbacks of these methods are that they either do not scale well with the number of classes or have limited performance in real-world computer vision datasets (see Section 2.5).

In this chapter, we focus on learning a domain-invariant representation that aligns both the marginal and the conditional distributions in domain generalization problems. We present theoretical results regarding the necessary and sufficient conditions for the existence of a domain-invariant representation, and subsequently propose a method to learn such representations by enforcing the invariance of the representation network under domain density transformation functions. A simple intuition for our approach is that if we enforce the representation to be invariant under the transformations among the source domains, the representation will become more robust under other domain transformations.

Furthermore, we introduce an implementation of our method in practice, in which the domain transformation functions are learned through the training process of generative adversarial networks (GANs) [Goodfellow et al., 2014, Choi et al., 2018]. We conduct extensive experiments on several widely used datasets and observe a significant improvement over relevant baselines. We also compare our methods against other state-of-the-art models and show that our method achieves competitive results.

Our contribution in this work is threefold:

- We revisit the domain invariant representation learning problem and shed some light by providing several observations: a necessary and sufficient condition for the existence of a domain-invariant representation and a connection between domain-independent representation and a marginally-aligned representation.
- We propose a theoretically grounded method for learning a domain-invariant representation based on domain density transformation functions. We also demonstrate that we can learn the domain transformation functions by GANs in order to implement our approach in practice.
- We empirically show the effectiveness of our method by performing experiments on widely used domain generalization datasets (e.g., Rotated MNIST, VLCS, and PACS) and comparing our method with relevant baselines (especially CIDG [Li et al., 2018c], CIDDG [Li et al., 2018d] and DGER [Zhao et al., 2020]).

2.2 Related Work

Domain generalization: Domain generalization is a seminal task in real-world machine learning problems where the data distribution of a target domain might vary from that of the training source domains. Therefore, extensive research has been developed to handle that domain-shift problem, aiming at a better generalization performance in the unseen target domain. A predominant approach for domain generalization is domain invariance [Muandet et al., 2013, Li et al., 2018c,d, Arjovsky et al., 2019, Wang et al., 2020, Akuzawa et al., 2019, Ilse et al., 2020, Zhao et al., 2020, Ajakan et al., 2014, Li et al., 2018b, Sun and Saenko, 2016, Shen et al., 2018] that learns a domain-invariant representation (which we define as to align the marginal distribution of the representation or the conditional distribution of the output given the representation or both). We are particularly interested in CIDG [Li et al., 2018c], CIDDG [Li et al., 2018d], and DGER [Zhao et al., 2020], which also learn a representation that aligns the joint distribution of the representation and the label given that the class distribution is unchanged across domains. It should be noted that Zhao et al. [2020] assume the label is distributed uniformly on all domains, while our proposed method only requires an assumption that the distribution of the label is unchanged across domains (and not necessarily uniform). We also show later in this chapter that the invariance of the distribution of class label across domains turns out

to be the necessary and sufficient condition for the existence of a domain-invariant representation. Moreover, we provide a unified theoretical discussion about the two types of alignment, and then propose a method to learn a representation that aligns both the marginal and conditional distributions via domain density transformation functions for the domain generalization problem. Note that there exist several related works, such as [Ajakan et al., 2014, Ganin et al., 2016], that use adversarial loss with a domain discriminator to align the marginal distribution of the representation among domains, but they are different from our approach. In particular, our method only uses GANs or normalizing flows to learn the transformations among domains, and learn a representation that is invariant under these functions, without using an adversarial loss on the representation (which can lead to very unstable training [Goodfellow, 2016, Kodali et al., 2017]). There also exist works [Liu and Tuzel, 2016, Hoffman et al., 2018, Bousmalis et al., 2017, Russo et al., 2018] in the domain adaptation literature that use generative modeling to learn a domain transformation function from source to target images, and use the transformed images to train a classifier. Our method differs from these by enforcing the representation to be invariant under the domain transformation, and we show theoretically that the representation learned that way would be domain-invariant marginally and conditionally. Meanwhile, the above works use the domain transformation to transform the images and train the classifier directly on the transformed data, and are not effective or applicable for domain generalization.

Another line of methods that received a recent surge in interest is applying the idea of meta-learning for domain generalization problems [Du et al., 2020, Balaji et al., 2018, Li et al., 2018a, Behl et al., 2019]. The core idea behind these works is that if we train a model that can adapt among the source domains well, it would be more likely to adapt to unseen target domains. Recently, there are approaches [Ding and Fu, 2017, Chattopadhyay et al., 2020, Seo et al., 2019] that make use of the domain specificity, together with domain invariance, for the prediction problem. The argument here is that domain invariance, while being generalized well between domains, might be insufficient for the prediction of each specific domain and thus domain specificity is necessary. We would like to emphasize that our method is not a direct competitor of meta-learning based and domain-specificity based methods. In fact, we expect that our method can be used in conjunction with these methods to get the best of both worlds for better performance.

Density transformation between domains: Since our method is based on domain density transformations, we will review briefly some related works here. To transform

the data density between domains, one can use several types of generative models. Two common methods are based on GANs [Zhu et al., 2017, Choi et al., 2018, 2020] and normalizing flows [Grover et al., 2020]. Although our method is not limited to the choice of the generative model used for learning the domain transformation functions, we opt to use GAN, specifically StarGAN [Choi et al., 2018], for scalability. This is just an implementation choice to demonstrate the use and effectiveness of our method in practice, and it is unrelated to our theoretical results.

Connection to contrastive learning: Our method can be interpreted intuitively as a way to learn a representation network that is invariant (robust) under domain transformation functions. On the other hand, contrastive learning [Chen et al., 2020a,b, Misra and Maaten, 2020] is a representation learning paradigm where the model learns images' similarity. In particular, contrastive learning encourages the representation of the input to be similar under different transformations (usually image augmentations). However, the transformations in contrastive learning are not learned and do not serve the purpose of making the representation robust under domain transformations. Our method first learns the transformations between domains and then uses them to learn a representation that is invariant under domain shifts.

2.3 Theoretical Approach

2.3.1 Problem Statement

Let us denote the data distribution for a domain $d \in \mathcal{D}$ by $p(x, y|d)$, where the variable $x \in \mathcal{X}$ represents the data and $y \in \mathcal{Y}$ is its corresponding label. The graphical model for our domain generalization framework is depicted in Figure 2.2, in which the joint distribution is presented as follows:

$$p(d, x, y, z) = p(d)p(y)p(x|y, d)p(z|x).$$

In the domain generalization problem, since the data distribution $p(x, y|d)$ varies across domains, we expect changes in the marginal data distribution $p(x|d)$ or the conditional data distribution $p(y|x, d)$ or both. In this work, we assume that $p(y|d)$ is invariant across domains, i.e., the marginal distribution of the label y is not dependent on the domain d – this assumption is shown to be the key condition for the existence of a domain-invariant representation (see Remark 2.1). This is practically reasonable since in many classification datasets, the class distribution can be assumed to be

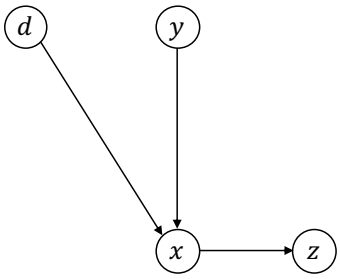


Figure 2.2: **Graphical model**. The data distribution is $p(x, y|d)$ for each domain d. Our goal is to learn a representation z with a mapping $p(z|x)$ from x so that z can be generalized across domains for the prediction task.

unchanged across domains (usually uniform distribution among the classes, e.g., balanced datasets).

Our aim is to find a domain-invariant representation z represented by the mapping $p(z|x)$ that can be used for the classification of label y and be generalized among domains. In practice, this mapping can be deterministic (in that case, $p(z|x) = \delta_{g_\theta(x)}(z)$ with some function g_θ, where δ is the Dirac delta function) or probabilistic (e.g., a normal distribution with the mean and standard deviation outputted by a network parameterized by θ). For all of our experiments, we use a deterministic mapping for an efficient inference at test time, while in this section, we present our theoretical results with the general case of a distribution $p(z|x)$.

In most existing domain generalization approaches, the domain-invariant representation z is defined using one of the two following definitions:

Definition 2.1 (Marginal Distribution Alignment)**.** *The representation z is said to satisfy the marginal distribution alignment condition if $p(z|d)$ is invariant w.r.t. d.*

Definition 2.2 (Conditional Distribution Alignment)**.** *The representation z is said to satisfy the conditional distribution alignment condition if $p(y|z, d)$ is invariant w.r.t. d.*

However, when the joint data distribution varies between domains, it is crucial to align both the marginal and the conditional distribution of the representation z. To this end, this work aims to learn a representation z that satisfies both the marginal and conditional alignment conditions. We justify our assumption of independence between y and d (thus $p(y|d) = p(y)$) by the following remark, which shows that this assumption turns out to be the necessary and sufficient condition for the existence

of a domain-invariant representation. Note that this condition is also used in several existing works [Zhao et al., 2020, Li et al., 2018c,d].

Remark 2.1. *The invariance of $p(y|d)$ across domains d is the necessary and sufficient condition for the existence of a domain-invariant representation (that aligns both the marginal and conditional distributions).*

Proof. provided in Appendix A.1.1. □

It is also worth noting that methods that learn a domain-independent representation, for example, [Ilse et al., 2020], only align the marginal distribution. This comes directly from the following remark:

Remark 2.2. *A representation z satisfies the marginal distribution alignment condition if and only if $I(z, d) = 0$, where $I(z, d)$ is the mutual information between z and d.*

Proof. provided in Appendix A.1.2. □

The question still remains how we can learn a non-trivial domain invariant representation that satisfies both of the distribution alignment conditions. This will be discussed in the following subsection.

2.3.2 Learning a Domain-Invariant Representation with Domain Density Transformation Functions

To present our method, we will make some assumptions about the data distribution. Specifically, for any two domains d, d', we assume that there exists an invertible and differentiable function denoted by $f_{d,d'}$ that transforms the density $p(x|y, d)$ to $p(x'|y, d'), \forall y$. Let $f_{d',d}$ be the inverse of $f_{d,d'}$, i.e., $f_{d',d} := (f_{d,d'})^{-1}$.

Due to the invertibility and differentiability of f's, we can apply the change of variables theorem [Rudin, 2006, Bogachev, 2007] for the distributions above. In particular, with $x' = f_{d,d'}(x)$ (and thus $x = f_{d',d}(x')$), we have:

$$p(x|y, d) = p(x'|y, d') \left| \det J_{f_{d',d}}(x') \right|^{-1}, \tag{2.1}$$

where $J_{f_{d',d}}(x')$ is the Jacobian matrix of the function $f_{d',d}$ evaluated at x'.

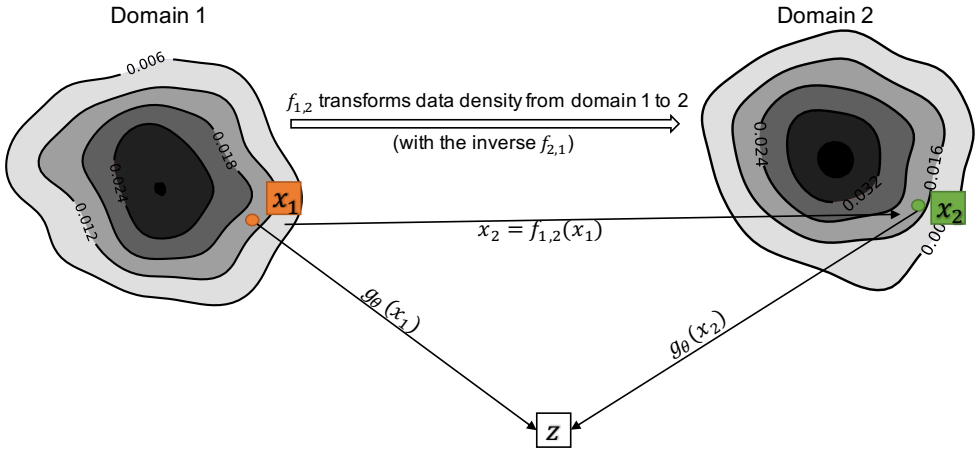


Figure 2.3: **Domain density transformation**. If we know the function $f_{1,2}$ that transforms the data density from domain 1 to domain 2, we can learn a domain invariant representation network $g_\theta(x)$ by enforcing it to be invariant under $f_{1,2}$, i.e., $g_\theta(x_1) = g_\theta(x_2)$ for any $x_2 = f_{1,2}(x_1)$.

Multiplying both sides of Eq. 2.1 with $p(y|d) = p(y|d')$, we get

$$p(x, y|d) = p(x', y|d') \left|\det\, J_{f_{d',d}}(x')\right|^{-1};$$

and marginalizing both sides of the above equation over y gives us

$$p(x|d) = p(x'|d') \left|\det\, J_{f_{d',d}}(x')\right|^{-1}. \tag{2.2}$$

By using Eq. 2.1 and Eq. 2.2, we can prove the following theorem, which offers an efficient way to learn a domain-invariant representation, given the transformation functions f's between domains.

Theorem 2.1. *Given an invertible and differentiable function $f_{d,d'}$ (with the inverse $f_{d',d}$) that transforms the data density from domain d to d' (as described above). Assuming that the representation z satisfies:*

$$p(z|x) = p(z|f_{d,d'}(x)),\ \forall x, \tag{2.3}$$

Then it aligns both the marginal and the conditional of the data distribution for domain d and d'.

Proof. provided in Appendix A.1.3. □

This theorem indicates that, if we can find the functions f's that transform the data densities among the domains, we can learn a domain-invariant representation z by

encouraging the representation to be invariant under all the transformations f's. This idea is illustrated in Figure 2.3. We therefore can use the following learning objective to learn a domain-invariant representation $z = g_\theta(x)$:

$$\mathbb{E}_d \left[\mathbb{E}_{p(x,y|d)} \left[l(y, g_\theta(x)) + \mathbb{E}_{d'}[dis(g_\theta(x), g_\theta(f_{d,d'}(x)))] \right] \right]. \tag{2.4}$$

Assume that we have a set of K sources domain $D_s = \{d_1, d_2, ..., d_K\}$, the objective function in Eq. 2.4 becomes:

$$\mathbb{E}_{d,d' \in D_s, p(x,y|d)} \left[l(y, g_\theta(x)) + dis(g_\theta(x), g_\theta(f_{d,d'}(x))) \right], \tag{2.5}$$

where $l(y, g_\theta(x))$ is the prediction loss of a network that predicts y given $z = g_\theta(x)$, and dis is a distance metric to enforce the invariant condition in Eq. 2.3. In our implementation, we use a squared error distance, e.g., $dis(g_\theta(x), g_\theta(f_{d,d'}(x))) = ||g_\theta(x) - g_\theta(f_{d,d'}(x))||_2^2$, since it performs the best in practice. However, we also considered other distances such as the constrastive distance, which we discuss in more detail in Appendix A.2.

This theorem motivates us to learn such domain transformation functions for our domain-invariant representation learning framework. In the next section, we show how one can incorporate this idea into real-world domain generalization problems by learning the transformations with generative adversarial networks.

2.4 A Practical Implementation using Generative Adversarial Networks

In practice, we can learn the functions f's that transform the data distributions among domains by using several generative modeling frameworks, e.g., normalizing flows [Grover et al., 2020] or GANs [Zhu et al., 2017, Choi et al., 2018, 2020]. One advantage of normalizing flows is that this transformation is naturally invertible by design of the neural network. However, existing frameworks (e.g., [Grover et al., 2020]) require two flows to transform between each pair of domains, making it not scalable (scales linearly with the number of domains). Moreover, an initial implementation of our method using AlignFlow shows similar performance to the version using GAN. Therefore, we opt to use GANs for better scalability. In particular, we use the StarGAN [Choi et al., 2018] model, which is a unified network (only requiring a single network to transform across all domains) designed for image domain transformations. It should be noted

that the transformations learned by StarGAN are differentiable everywhere or almost everywhere with typical choices of the activation function (e.g., tanh or ReLU), and the cycle consistency loss enforces each pair of transformations to approximate a pair of inverse functions.

The goal of StarGAN is to learn a unified network G that transforms the data density among multiple domains. In particular, the network $G(x, d, d')$ (i.e., G is conditioned on the image x and the two different domains d, d') transforms an image x from domain d to domain d'. Different from the original StarGAN model that only takes the image x and the desired destination domain d' as its input, in our implementation, we feed both the original domain d and desired destination domain d' together with the original image x to the generator G.

The generator's goal is to fool a discriminator D into thinking that the transformed image belongs to the destination domain d'. In other words, the equilibrium state of StarGAN, in which G completely fools D, is when G successfully transforms the data density of the original domain to that of the destination domain. After training, we use $G(., d, d')$ as the function $f_{d,d'}(.)$ described in the previous section and perform representation learning via the objective function in Eq. 2.5.

Three important loss functions of the StarGAN architecture are:

- Domain classification loss $\mathcal{L}_{cls}$ that encourages the generator G to generate images that closely belong to the desired destination domain d'.
- The adversarial loss $\mathcal{L}_{adv}$ that is the classification loss of a discriminator D that tries to distinguish between real images and the synthetic images generated by G. The equilibrium state of StarGAN is when G completely fools D, which means the distribution of the generated images (via $G(x, d, d'), x \sim p(x|d)$) becomes the distribution of the real images of the destination domain $p(x'|d')$. This is our objective, i.e., to learn a function that transforms domains' densities.
- Reconstruction loss $\mathcal{L}_{rec} = \mathbb{E}_{x,d,d'}[||x - G(x', d', d)||_1]$ where $x' = G(x, d, d')$ to ensure that the transformations preserve the image's content. Note that this also aligns with our interest since we want $G(., d', d)$ to be the inverse of $G(., d, d')$, which minimizes $\mathcal{L}_{rec}$.

We can enforce the generator G to transform the data distribution within the class y (e.g., $p(x|y, d)$ to $p(x'|y, d')$ $\forall y$) by sampling each minibatch with data from the same class y, so that the discriminator will distinguish the transformed images with the real

images from class y and domain d'. However, we found that this constraint can be relaxed in practice, and the generator almost always transforms the image within the original class y.

As mentioned earlier, after training the StarGAN model, we can use the generator $G(., d, d')$ as our $f_{d,d'}(.)$ function and learn a domain-invariant representation via the learning objective in Eq. 2.5. We name this implementation of our method DIRT-GAN (Domain Invariant Representation learning with domain Transformations via Generative Adversarial Networks).

2.5 Experiments

2.5.1 Datasets

To evaluate our method, we perform experiments in three datasets that are commonly used in the domain generalization literature.

Rotated MNIST [Ghifary et al., 2015]: In this dataset, 1,000 MNIST [LeCun et al., 2010] images (100 per class) are chosen to form the first domain (denoted $\mathcal{M}_0$), then counter-clockwise rotations of $15°, 30°, 45°, 60°$, and $75°$ are applied to create five additional domains, denoted $\mathcal{M}_{15}, \mathcal{M}_{30}, \mathcal{M}_{45}, \mathcal{M}_{60}$, and $\mathcal{M}_{75}$. The task is classification with ten classes (digits 0 to 9).

VLCS [Ghifary et al., 2015]: contains 10,729 images from four domains, each of which is a subdataset. The four datasets are VOC2007 (V), LabelMe (L), Caltech-101 (C), and SUN09 (S). The task is classification with five classes.

PACS [Li et al., 2017]: contains 9,991 images from four different domains: art painting, cartoon, photo, and sketch. The task is classification with seven classes.

2.5.2 Experimental Setting

For all datasets, we perform "leave-one-domain-out" experiments, where we choose one domain as the target domain, train the model on all remaining domains, and evaluate it on the chosen domain. Following standard practice, we use 90% of available data as training data and 10% as validation data, except for the Rotated MNIST experiment where we do not use a validation set and just report the performance of the last epoch.

For the **Rotated MNIST** dataset, we use a network of two 3x3 convolutional layers and a fully connected layer as the representation network g_θ to get a representation z of 64 dimensions. A single linear layer is then used to map the representation z to the ten output classes. This architecture is the deterministic version of the network used by Ilse et al. [2020]. We train our network for 500 epochs with the Adam optimizer [Kingma and Ba, 2015], using learning rate 0.001 and minibatch size 64, and report performance on the test domain after the last epoch.

For the **VLCS** and **PACS** datasets, for a fair comparison against our main baselines, we use the most common choices of backbone networks for those datasets in existing works as the representation networks g_θ, i.e., Alexnet [Krizhevsky et al., 2012] for VLCS and Resnet18 [He et al., 2016] for PACS. We replace the last fully connected layer of the backbone with a linear layer of dimension 256 so that our representation has 256 dimensions. As with the Rotated MNIST experiment, we use a single linear layer to map from the representation z to the output. We train the network for 100 epochs with stochastic gradient descent (SGD) using learning rate 0.001, momentum 0.9, minibatch size 64, and weight decay 0.001. Data augmentation is also standard practice for real-world computer vision datasets like VLCS and PACS, and during training, we augment our data as follows: crops of random size and aspect ratio, resizing to 224 $\times$ 224 pixels, random horizontal flips, random color jitter, randomly converting the image tile to grayscale with 10% probability, and normalization using the ImageNet channel means and standard deviations.

The StarGAN [Choi et al., 2018] model implementation is taken from the authors' source code with no significant modifications. For each set of source domains, we train the StarGAN model for 100,000 iterations with a minibatch of 16 images per iteration.

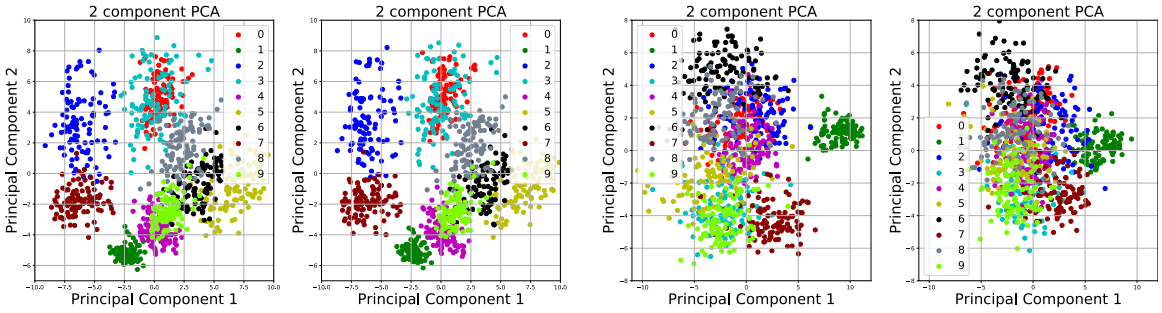
Our code is available at https://github.com/atuannguyen/DIRT. We train our model on a NVIDIA Quadro RTX 6000.

2.5.3 Results

Rotated MNIST Experiment. Table 2.1 shows the performance of our model on the Rotated MNIST dataset. The main baselines we consider in this experiment are HIR [Wang et al., 2020], DIVA [Ilse et al., 2020], and DGER [Zhao et al., 2020], which are domain-invariance based methods. Our method recognizably outperforms those, illustrating the effectiveness of the method in learning a domain-invariant representation over the existing works. We also include other best-performing models

Table 2.1: **Rotated Mnist**. Reported numbers are mean accuracy and standard deviation among 5 runs

Model	Domains						Average
	$\mathcal{M}_0$	$\mathcal{M}_{15}$	$\mathcal{M}_{30}$	$\mathcal{M}_{45}$	$\mathcal{M}_{60}$	$\mathcal{M}_{75}$	
HIR [Wang et al., 2020]	90.34	99.75	99.40	96.17	99.25	91.26	96.03
DIVA [Ilse et al., 2020]	93.5	99.3	99.1	99.2	99.3	93.0	97.2
DGER [Zhao et al., 2020]	90.09	99.24	99.27	99.31	99.45	90.81	96.36
DA [Ganin et al., 2016]	86.7	98.0	97.8	97.4	96.9	89.1	94.3
LG [Shankar et al., 2018]	89.7	97.8	98.0	97.1	96.6	92.1	95.3
HEX [Wang et al., 2019]	90.1	98.9	98.9	98.8	98.3	90.0	95.8
ADV [Wang et al., 2019]	89.9	98.6	98.8	98.7	98.6	90.4	95.2
DIRT-GAN	97.2(±0.3)	99.4(±0.1)	99.3(±0.1)	99.3(±0.1)	99.2(±0.1)	97.1(±0.3)	**98.6**



(a) Domain $\mathcal{M}_{30}$ DIRT-GAN (b) Domain $\mathcal{M}_{60}$ DIRT-GAN (c) Domain $\mathcal{M}_{30}$ DeepAll (d) Domain $\mathcal{M}_{60}$ DeepAll

Figure 2.4: **Visualization of the representation space**. Each point indicates a representation z of an image x in the two-dimensional space and its color indicates the label y. Two left figures are for our method DIRT-GAN and two right figures are for the naive model DeepAll.

for this dataset in the second half of the table. To the best of our knowledge, we set a new state-of-the-art performance on this Rotated MNIST dataset.

We further analyze the distribution of the representation z by performing principal component analysis to reduce the dimension of z from 64 to two principal components. We visualize the representation space for two domains $\mathcal{M}_{30}$ and $\mathcal{M}_{60}$, with each point indicating the representation z of an image x in the two-dimensional space and its color indicating the label y. Figures 2.4a and 2.4b show the representation space of our method (in domains $\mathcal{M}_{30}$ and $\mathcal{M}_{60}$ respectively). It is clear that both the marginal (judged by the general distribution of the points) and the conditional (judged by the positions of colors) are relatively aligned. Meanwhile, Figures 2.4c and 2.4d show the representation space with naive training (in domains $\mathcal{M}_{30}$ and $\mathcal{M}_{60}$ respectively), showing the misalignment in the marginal distribution (judged by the general distribution of the points) and the conditional distribution (for example, the

Table 2.2: **VLCS**. Reported numbers are mean accuracy and standard deviation among 5 runs

Model	VLCS				
	V	L	C	S	Average
CIDG [Li et al., 2018c]	65.65	60.43	91.12	60.85	69.51
CIDDG [Li et al., 2018d]	64.38	63.06	88.83	62.10	69.59
DGER [Zhao et al., 2020]	73.24	58.26	96.92	69.10	74.38
HIR [Wang et al., 2020]	69.10	62.22	95.39	65.71	73.10
JiGen [Carlucci et al., 2019]	70.62	60.90	96.93	64.30	73.19
DIRT-GAN (ours)	72.1(±1.0)	64.0(±0.9)	97.3(±0.2)	72.2(±1.1)	**76.4**

Table 2.3: **PACS**. Reported numbers are mean accuracy and standard deviation among 5 runs

Model	PACS				
	Art Painting	Cartoon	Photo	Sketch	Average
DGER [Zhao et al., 2020]	80.70	76.40	96.65	71.77	81.38
JiGen [Carlucci et al., 2019]	79.42	75.25	96.03	71.35	79.14
MLDG [Li et al., 2018a]	79.50	77.30	94.30	71.50	80.70
MetaReg [Balaji et al., 2018]	83.70	77.20	95.50	70.40	81.70
CSD [Piratla et al., 2020]	78.90	75.80	94.10	76.70	81.40
DMG [Chattopadhyay et al., 2020]	76.90	80.38	93.35	75.21	81.46
DIRT-GAN (ours)	82.56(± 0.4)	76.37(± 0.3)	95.65(± 0.5)	79.89(± 0.2)	**83.62**

distributions of blue points and green points).

VLCS and PACS. Tables 2.2 and 2.3 show the results for the VLCS and PACS datasets. In these real-world computer vision datasets, we consider HIR [Wang et al., 2020], CIDG [Li et al., 2018c], CIDDG [Li et al., 2018d], and DGER [Zhao et al., 2020] as our main domain-invariance baselines. We also include other approaches (meta-learning based or domain-specificity based) in the second half of the tables for reference. Our method significantly outperforms other invariant-representation baselines, namely CIDG, CIDDG, and DGER, with the same backbone architectures, showing the effectiveness of our representation alignment method.

2.6 Conclusion

To conclude, in this work we propose a theoretically grounded approach to learn a domain-invariant representation for the domain generalization problem by using

domain transformation functions. We also provide insights into domain-invariant representation learning with several theoretical observations. We then introduce an implementation for our method in practice with the domain transformations learned by a StarGAN architecture and empirically show that our approach outperforms other domain-invariance based methods. Our method also achieves competitive results on several datasets when compared to other state-of-the-art models. A potential limitation of our method is that we need to train an additional network (StarGAN) to learn to transform data density among domains. However, this network is only used during training, and the required computation at test time is still the same as other models. In the future, we plan to incorporate our method into meta-learning based and domain-specificity based approaches for improved performance. We also plan to extend the domain-invariant representation learning framework to the more challenging scenarios, for example, where domain information is not available (i.e., we have a dataset pooled from multiple source domains but do not know the domain identification of each data instance).

Chapter 3

KL Guided Domain Adaptation

Abstract

Domain adaptation is an important problem and often needed for real-world applications. In this problem, instead of i.i.d. training and testing data points, we assume that the source (training) data and the target (testing) data have different distributions. With that setting, the empirical risk minimization training procedure often does not perform well, since it does not account for the change in the distribution. A common approach in the domain adaptation literature is to learn a representation of the input that has the same (marginal) distribution over the source and the target domains. However, these approaches often require additional networks and/or optimizing an adversarial (minimax) objective, which can be very expensive or unstable in practice. To improve upon these marginal alignment techniques, in this work, we first derive a generalization bound for the target loss based on the training loss and the reverse Kullback-Leibler (KL) divergence between the source and the target representation distributions. Based on this bound, we derive an algorithm that minimizes the KL term to obtain a better generalization to the target domain. We show that with a probabilistic representation network, the KL term can be estimated efficiently via minibatch samples without any additional network or a minimax objective. This leads to a theoretically sound alignment method which is also very efficient and stable in practice. Experimental results also suggest that our method outperforms other representation-alignment approaches.

3.1 Introduction

With advances in neural network architectures [He et al., 2016, Vaswani et al., 2017], machine learning algorithms have achieved state-of-the-art performance in many tasks such as object classification, object detection, and natural language processing. However, machine learning models have been focusing mostly on the case of independent and identically distributed (i.i.d.) data points; and such an assumption often does not hold in practice. When the i.i.d. assumption is violated and the target domain has a different distribution compared to the source domain, a typical learner trained on the source data via empirical risk minimization would not perform well at test time, since it does not account for the distribution shift. To tackle this problem, many methods have been proposed for domain adaptation [Zhao et al., 2019, Zhang et al., 2019, Combes et al., 2020, Tanwani, 2020] and domain generalization [Khosla et al., 2012, Muandet et al., 2013, Ghifary et al., 2015], the goal of which is to train a machine learning algorithm that can generalize well to the target domain.

A common approach to tackle these problems is to learn a representation such that its distribution does not change across domains. There are two types of distribution alignment: marginal alignment (aligning the marginal distribution of the representation) and conditional alignment (aligning the conditional distribution of the label given the representation) [Nguyen et al., 2021, Tanwani, 2020]. For domain adaptation and domain generalization problems with multiple source domains, we can use the data and labels to align both the marginal and the conditional distributions across the source domains, aiming to generalize to the target domain. However, in a single-source domain adaptation problem, with only unlabeled data from the target domain, it is often only possible to align the marginal distribution of the representation. This marginal alignment should help the classifier avoid out-of-distribution data at test time.

This chapter focuses on such a single-source domain adaption problem, which is also one of the most common settings in practice. Current marginal alignment techniques usually require additional computation (e.g., of an additional network) [Ganin et al., 2016, Li et al., 2018b] and/or a minimax objective [Ganin et al., 2016, Shen et al., 2018], leading to an expensive and/or unstable training procedure [Goodfellow, 2016, Kodali et al., 2017]. For example, DANN [Ganin et al., 2016] employs an adversarial training procedure, with a domain discriminator that classifies the domain of the representation, and maximizes the adversarial loss of the discriminator. When the discriminator is completely fooled, the marginal distribution of the representation is

aligned across domains. MMD [Gretton et al., 2012, Li et al., 2018b] utilizes maximum mean discrepancy to align the representation distribution. This does not use a minimax objective, thus leading to a more stable training process; however, it does require additional computation of several Gaussian kernels. While more sophisticated (non-marginal-alignment) methods have been proposed recently to achieve better results in the domain adaptation problem, we argue that studying the family of plain marginal-alignment techniques is still an important task since they are the backbones upon which most other domain adaptation (and domain generalization) methods are built.

To address the above issues of existing marginal alignment techniques, we first derive a generalization bound on the loss of the target domain using the training loss and a reverse Kullback-Leibler (KL) divergence between the source and target distributions. There are existing bounds of the target loss in the literature [Ben-David et al., 2010], however, these analyses focus mostly on the case of binary classification and the bounds use a total variation distance or a $\mathcal{H}$-divergence between the distributions, which are not easy to estimate in practice (for example, Ajakan et al. [2014] require an adversarial network to estimate the $\mathcal{H}$-divergence). In this work, we show that with a probabilistic representation network, we can estimate the KL divergence easily using samples, leading to an alignment method that requires virtually no additional computation nor a minimax objective. Therefore, our training procedure is simple and stable in practice. Moreover, the reverse KL has the zero-forcing effect [Minka, 2005], which is very effective in alleviating the out-of-distribution problem in practice. This can be explained as follows: the out-of-distribution problem arises when the classifier faces a new representation at test time that is in a (near) zero mass region of the source representation distribution (and thus it never faced before). The reverse KL tends to force the target representation distribution to have (near) zero mass wherever the source distribution has (near) zero mass (this is the zero-forcing property), which helps the classifier avoid out-of-distribution data. The reverse KL also has the mode-seeking effect [Minka, 2005] which allows for a more flexible alignment of the representation (to one or some of the modes of the source domain). For example, consider the classification problem of buildings (houses, hotels, etc.) where source images are collected from urban and remote areas of a country (two modes); while the target images are collected from urban areas but from a different country. Ideally, we want to match the representation distribution of the target domain to that of the first mode of the source domain since they are both from urban areas. The reverse KL allows this flexible alignment (as it results in a relatively small value of the reverse

KL) due to its mode-seeking property. Meanwhile, other distance metrics/divergences aim to match the whole source and target representation distribution, which might collapse the two modes of the source domains.

Our contributions in this work are:

- We construct a generalization bound of the target loss in the domain adaptation problem using the reverse KL divergence.
- We propose to reduce the generalization bound by minimizing the above KL term. Furthermore, we show that with a probabilistic representation, the KL term can be estimated easily using minibatches, without any additional computation or a minimax objective as opposed to most existing works.
- We conduct extensive experiments and show that our method significantly outperforms relevant baselines, namely ERM [Bousquet et al., 2003], DANN [Ganin et al., 2016], MMD [Gretton et al., 2012, Li et al., 2018b], CORAL [Sun and Saenko, 2016], and WD [Shen et al., 2018]. We empirically show that the reverse KL divergence is very effective for representation alignment since it is stable and efficient to compute in practice.

3.2 Related Work

Generalization bound for the distribution shift problem: There exist works studying bounds for the distribution shift problem in the literature [Ben-David et al., 2010, Mansour et al., 2009]. However, their analyses of the classification problem are limited to the case of binary labels. Moreover, these bounds are only applicable or practical for deterministic labeling functions, which is not the case for most datasets in practice. Therefore, their analyses cannot be generalized to the general case of supervised learning. The differences between our bound and theirs are as follows. First of all, our bound works for the general case of supervised learning: it works for both the classification (including multiclass classification) and regression problems, it makes no assumptions about the labeling mechanism (can be probabilistic or deterministic), and it works for virtually all predictive distributions commonly used in practice. Secondly, our bound uses a different divergence, namely the KL divergence, which is easier to estimate in practice compared to total variation or $\mathcal{H}$-divergence. We provide a brief review of the above bounds and discuss their differences from ours in more detail in Appendix B.2. Recently, Acuna et al. [2021] revise the previous domain adaptation

bounds and generalize them to a multi-class classification setting, as well as to the class of f-divergence (including KL divergence). However, their setting is still very restricted: the loss function needs to satisfy the triangle inequality (which does not hold for many loss functions in practice), and they need to know the true labeling function (optimal Bayes classifier) for a probabilistic labeling mechanism (which is often not available). We also provide further analysis, that under some reasonable assumptions, the conditional misalignment in the representation space is bounded by the conditional misalignment in the input space, which allows for a sound marginal alignment method. This can be seen as an improvement over prior works. Some specific cases of distribution shift have also been studied. For example, Cortes et al. [2010] and Johansson et al. [2019] study the generalization bound for the covariate shift problem, i.e., $p_T(x) \neq p_S(x)$ but $p_T(y|x) = p_S(y|x)$, where p_S is the source distribution and p_T is the target distribution. In contrast, Azizzadenesheli et al. [2019] provide a generalization bound for the label shift problem, i.e., $p_T(y) \neq p_S(y)$ but $p_T(x|y) = p_S(x|y)$.

Domain adaptation: While the literature on the domain adaptation problem is vast, we cover the most closely related works to ours here. A common method for the domain adaptation problem is to align the marginal distribution of the representation between the source and target domains. DANN [Ganin et al., 2016] employs a domain discriminator to classify the domain of a representation and maximizes its adversarial loss (a minimax game). WD [Shen et al., 2018] uses a neural network function f (which is 1-Lipschitz continuous) to calculate the Wasserstein distance between two distributions and minimizes it. This is also a minimax game since the Wasserstein distance is the supremum over the search space of f. MMD [Gretton et al., 2012, Li et al., 2018c] uses the maximum mean discrepancy to align the representation distributions. This method does not need a minimax objective; however, it requires the additional computation of several Gaussian kernels. Finally, CORAL [Sun and Saenko, 2016] matches the first two moments of the distributions; and while being a simple method, it fails to align more complex distributions. We consider these marginal alignment techniques our main baselines since our method falls into this category, and investigate the effectiveness of the reverse KL divergence in aligning the distributions of representation. Recently, more sophisticated alignment methods [Kang et al., 2019, Xu et al., 2019, Zhu et al., 2020] have been proposed for the domain adaptation problem, which achieve state-of-the-art performance. Instead of simply aligning the marginal distribution of the representation, these methods minimize the

intra-class distance of the representation across domains, and possibly maximize the inter-class distance between them, using the MMD or L2 distance. However, they require pseudo labels for the target domain (often obtained via clustering). Moreover, they are complementary to our method, as we conjecture that our method can also be used in conjunction with these, leading to the same algorithms but with the KL distance instead of MMD or L2.

3.3 Approach

3.3.1 Problem Statement

In this work, we consider one of the most common domain adaptation settings, which consists of a single source domain S with the joint data distribution $p_S(x, y)$ and a target domain T with the data distribution $p_T(x, y)$, where x denotes the input sample and y is the label. We assume that these two domains have the same support sets $\mathcal{X}, \mathcal{Y}$. Regarding the training process of the domain adaptation problem, we further denote a labeled dataset of size N_S sampled from the source domain $(x_S^{(i)}, y_S^{(i)})_{i=1}^{N_S}$, where $(x_S^{(i)}, y_S^{(i)}) \sim p_S(x, y)$, and an unlabeled dataset of size N_T from the target domain $(x_T^{(i)})_{i=1}^{N_T}$, where $x_T^{(i)} \sim p_T(x)$.

The goal of a typical domain adaptation framework is to train a model using the labeled dataset of the source domain together with the unlabeled dataset from the target domain, so that the model will perform decently in the target domain. Note that this is only effective if the labeling mechanism is not too different between the source and the target domains [Ben-David et al., 2010].

In the domain adaption problem, we expect the changes in the marginal distribution so that $p_S(x) \neq p_T(x)$, or the conditional distribution so that $p_S(y|x) \neq p_T(y|x)$, or both, which often render the typical empirical risk minimization training procedure ineffective. This motivates a line of approaches that learn a representation z of x whose marginal and conditional distributions are more aligned across the domains and use it for the prediction task, aiming for a better generalization performance to the target domain.

The general representation learning framework aims to learn a representation z from x with the mapping $p(z|x)$, which can be deterministic or probabilistic. That latent representation z is expected to contain the label-related information; and is then used to predict the label y (by a classifier). Note that since the source

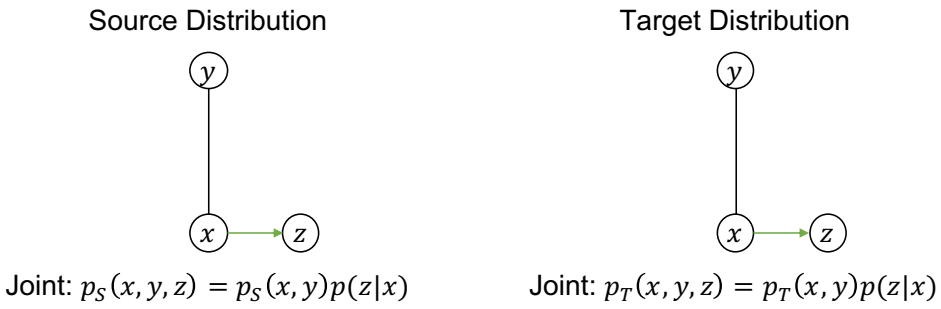


Figure 3.1: **Graphical model**. Note that the distribution $p(z|x)$ (green edge), corresponding to our representation network, is shared between the source and target domains.

and target domains have the same support set for x and share the representation mapping $p(z|x)$, they also have the same support set for z, denoted by $\mathcal{Z}$. Given the representation z, we learn a classifier to predict y through the predictive distribution $\hat{p}(y|z)$ that is an approximation of the ground truth conditional distribution $p_S(y|z)$. During training, the representation network $p(z|x)$ and the classifier $\hat{p}(y|z)$ are trained jointly on the source domain, and we "hope" that they can generalize to the target domain, meaning that both $p(z|x)$ and $\hat{p}(y|z)$ are kept unchanged between training and testing. The graphical model of that representation learning process is represented in Figure 3.1. In this chapter, we consider a probabilistic representation mapping; specifically, the representation network will output vectors $\mu(x)$ and $\sigma^2(x)$, and $p(z|x) = \mathcal{N}(z; \mu(x), \text{diag}(\sigma^2(x)))$, where $\mathcal{N}$ denotes a Gaussian distribution. This can also be thought of as a generalization of a deterministic representation, as we recover the deterministic case if $\sigma^2(x) \to 0$. Also, note that our method is not limited to the choice of this representation distribution, i.e., the discussion in this section holds for virtually any other distribution.

The joint distributions of x, y, z for the source and target domains can be represented as follows:

$$p_S(x, y, z) = p_S(x, y)p(z|x) \ , \qquad p_T(x, y, z) = p_T(x, y)p(z|x) \ .$$

And we define the predictive distribution of y given x as

$$\hat{p}(y|x) = \mathbb{E}_{p(z|x)}[\hat{p}(y|z)] \ . \tag{3.1}$$

Remark 3.1 (Inference complexity of a probabilistic representation)**.** *Using a probabilistic representation, we need to sample multiple z from $p(z|x)$ to estimate Eq. 3.1 during test time. However, this is not a big issue for the representation learning*

framework, since we only need to run the representation network $p(z|x)$ (which is usually deep) once to get a distribution of z. After sampling multiple z from that distribution, we only need to rerun the classifier $\hat{p}(y|z)$, which is usually a small network (e.g., often contains one layer). Furthermore, we can also run $\hat{p}(y|z)$ (a small network) in parallel for multiple z to reduce inference time if necessary.

During training, we usually sample a single z from $p(z|x)$ for each x. The training objective is

$$\begin{aligned} l_{train} &= \mathbb{E}_{x,y\sim p_S(x,y),z\sim p(z|x)}[-\log \hat{p}(y|z)] \\ &\quad \text{(this is also the upper bound of } \mathbb{E}_{p_S(x,y)}[-\log \hat{p}(y|x)] \text{ via Jensen Inequality)} \\ &= \mathbb{E}_{p_S(z,y)}[-\log \hat{p}(y|z)] \end{aligned}$$

where $-\log \hat{p}(y|z)$ is the loss of a "data point" (z, y). For common choices of the predictive distribution in the classification and regression problems, this is a non-negative quantity. For example, for a classification problem with a categorical predictive distribution, this becomes the cross-entropy loss, while for a regression problem with a Gaussian predictive distribution (with a fixed variance), it becomes the squared error (with an additive constant).

Minimizing l_{train} will enforce $\hat{p}(y|z) \approx p_S(y|z)$.

We consider the below two assumptions of the representation z on the source domain:

Assumption 3.1. *$I_S(z, y) = I_S(x, y)$, where $I_S(\cdot, \cdot)$ is the mutual information term, calculated on the source domain. In particular:*

$$I_S(z,y) = \mathbb{E}_{p_S(z,y)}\left[\log \frac{p_S(z,y)}{p_S(z)p_S(y)}\right]; \quad I_S(x,y) = \mathbb{E}_{p_S(x,y)}\left[\log \frac{p_S(x,y)}{p_S(x)p_S(y)}\right]$$

This is often referred to as the "sufficiency assumption" since it indicates that the representation z has the same information about the label y as the original input x, and is sufficient for this prediction task (in the source domain). Note that the data processing inequality indicates that $I_S(z, y) \leq I_S(x, y)$, so here we assume that z contains maximum information about y.

Remark 3.2. *Assumption 3.1 is an optimization goal of the training process on the source domain.*

In particular, l_{train} (with an additive constant) is an upper bound of $-I_S(z, y)$, which is an upper bound of $-I_S(x, y)$. Thus, minimizing l_{train} will enforce $I_S(z, y)$ to be equal to $I_S(x, y)$. For a more detailed discussion of this, please refer to, for example, [Alemi et al., 2017].

Assumption 3.2. $p_S(y|x) = \mathbb{E}_{p(z|x)}[p_S(y|z)] \quad \forall x, y \in \mathcal{X}, \mathcal{Y}$

When this assumption holds, the predictive distribution in Eq. 3.1 will approximate $p_S(y|x)$, as long as $\hat{p}(y|z)$ approximates $p_S(y|z)$.

Remark 3.3. *Assumption 3.2 is also an optimization goal of the training process on the source domain.*

This is because l_{train} is an upper bound of $\mathbb{E}_{p_S(x,y)}[-\log \hat{p}(y|x)]$, which is an upper bound of $\mathbb{E}_{p_S(x,y)}[-\log p_S(y|x)]$. Thus, minimizing l_{train} will enforce $\hat{p}(y|x)$ to be equal to $p_S(y|x)$. Therefore, $p_S(y|x) \approx \hat{p}(y|x) = \mathbb{E}_{p(z|x)}[\hat{p}(y|z)] \approx \mathbb{E}_{p(z|x)}[p_S(y|z)]$.

These two assumptions ensure that our network has good performance on the **source** domain. Note also that we only make the above two assumptions about the source domain, where we can enforce them through the training process. We do not make these assumptions about the target domain, since we have no access to the full target distribution. These two assumptions will also be used to prove our later theoretical result (Proposition 3.2).

3.3.2 KL Guided Domain Adaptation

Now we will consider the test loss in the domain adaptation problem, and how we can reduce it. The test loss (of the target domain) is:

$$\begin{aligned} l_{test} &= \mathbb{E}_{p_T(x,y)}[-\log \hat{p}(y|x)] = \mathbb{E}_{p_T(x,y)}[-\log \mathbb{E}_{p(z|x)}[\hat{p}(y|z)]] \\ &\leq \mathbb{E}_{p_T(x,y)}[\mathbb{E}_{p(z|x)}[-\log \hat{p}(y|z)]] \quad \text{(Jensen Inequality)} \\ &= \mathbb{E}_{p_T(z,y)}[-\log \hat{p}(y|z)] \end{aligned} \tag{3.2}$$

Note that if the representation z is invariant (both marginally and conditionally), then $p_T(z, y) = p_S(z, y)$ and Eq. 3.2 becomes l_{train}, and we have a perfect generalization between the source domain and the target domain. However, there is no way to guarantee the invariance, since we do not know the target domain and the target data distribution. In that case, we introduce the following proposition that ensures a generalization bound of the test loss based on the training loss and the KL divergence:

Proposition 3.1. *If the loss* $-\log \hat{p}(y|z)$ *is bounded by* M [1] $\forall z \in \mathcal{Z}, y \in \mathcal{Y}$, *we have:*

$$\begin{aligned} l_{test} &\leq l_{train} + \frac{M}{\sqrt{2}}\sqrt{\mathrm{KL}[p_T(y,z)|p_S(y,z)]} \\ &= l_{train} + \frac{M}{\sqrt{2}}\sqrt{\mathrm{KL}[p_T(z)|p_S(z)] + \mathbb{E}_{p_T(z)}\left[\mathrm{KL}[p_T(y|z)|p_S(y|z)]\right]} \end{aligned}$$

Proof. provided in Appendix B.1.1. □

This bound is similar to other bounds in the literature (e.g., [Ben-David et al., 2010]) in the sense that it also contains the training loss, a marginal misalignment term and a conditional misalignment term ($\mathrm{KL}[p_T(z)|p_S(z)]$ and $\mathbb{E}_{p_T(z)}\left[\mathrm{KL}[p_T(y|z)|p_S(y|z)]\right]$ respectively in our case). However, Ben-David et al. [2010] consider a binary classification problem, and their bounds are only practical for a deterministic labeling function (requires knowing the true labeling function, which is unknown for a probabilistic labeling mechanism, to compute the bound); while our bound works for the general case of supervised learning with any labeling mechanism. For a brief review of these bounds and a detailed discussion about their differences from ours, please refer to Appendix B.2. Note that the bound in Proposition 3.1 is also true when applying to the input space directly (e.g., replacing z with x). However, we are more interested in the bound in the representation space, since we can reduce it by regularizing the KL term.

To reduce the generalization gap, we want $p_T(z, y)$ to be close to $p_S(z, y)$. Aligning the marginal distribution (i.e., $p_S(z) \approx p_T(z)$) helps the classifier network $\hat{p}(y|z)$ avoid out-of-distribution data since the target representations it faces at test time belong to the source representation distribution which it was trained on; while aligning the conditional distribution ($p_S(y|z) \approx p_T(y|z)$) makes sure the classifier gives more accurate predictions on the target domain since $\hat{p}(y|z)$ was trained to approximate $p_S(y|z)$. In the domain adaptation problem, since we only have the unlabeled data from the target domain, we often align the marginal distribution of z only. However, one problem is that the conditional misalignment also depends on the representation z, and when learning a representation z that aligns the marginal, we might accidentally increase $\mathbb{E}_{p_T(z)}\left[\mathrm{KL}[p_T(y|z)|p_S(y|z)]\right]$ at the same time, leading to a net increase in

[1] In the classification problem, we can enforce this quite easily by augmenting the output softmax of the classifier so that each class probability is always at least $\exp(-M)$. For example, if we choose $M = 3 \Rightarrow \exp(-M) \approx 0.05$, and if the output softmax is $(p_1, p_2, ..., p_C)$, we can augment it into $(p_1 \cdot K + 0.05, p_2 \cdot K + 0.05, ..., p_C \cdot K + 0.05)$, where $K = 1 - 0.05 \cdot C$ and C is the number of classes. This ensures the bound for the loss of a data point, while remaining the output prediction class.

the above generalization bound. For example, what if (and is it possible that) the conditional misalignment increases to infinity while we learn a representation z?

Therefore, it is crucial that we can bound the above conditional misalignment. The below proposition handles this problem.

Proposition 3.2. *If Assumption 3.1 and 3.2 hold, and if* $\frac{p_T(x,y)}{p_S(x,y)} < \infty$ *(i.e., there exists N, which can be arbitrarily large, such that* $\frac{p_T(x,y)}{p_S(x,y)} < N \; \forall x \in \mathcal{X}, y \in \mathcal{Y}$*), we have:*

$$\mathbb{E}_{p_T(z)}\left[\mathrm{KL}[p_T(y|z)|p_S(y|z)]\right] \leq \mathbb{E}_{p_T(x)}\left[\mathrm{KL}[p_T(y|x)|p_S(y|x)]\right]$$

Proof. provided in Appendix B.1.2. □

This shows that the conditional misalignment in the representation space is bounded by the conditional misalignment in the input space. This can also be viewed as an improvement over the analyses of Ben-David et al. [2010], where it is not clear if the conditional misalignment in the representation space is bounded or not. It then follows that:

$$l_{test} \leq l_{train} + \frac{M}{\sqrt{2}}\sqrt{\mathrm{KL}[p_T(z)|p_S(z)] + \mathbb{E}_{p_T(x)}\left[\mathrm{KL}[p_T(y|x)|p_S(y|x)]\right]}. \tag{3.3}$$

As mentioned earlier, in order for domain adaptation to be effective, we should expect that the labeling mechanism does not change too much [Ben-David et al., 2010]. Thus, the conditional misalignment $\mathbb{E}_{p_T(x)}\left[\mathrm{KL}[p_T(y|x)|p_S(y|x)]\right]$ is often small (and fixed – not dependent on the representation z). Therefore, to reduce the generalization bound, we can focus on minimizing $\mathrm{KL}[p_T(z)|p_S(z)]$, with the objective:

$$l_{train} + \beta \mathrm{KL}[p_T(z)|p_S(z)] \tag{3.4}$$

where β is a hyper-parameter.

Discussion on the use of reverse KL: Our derivation leads to the reverse KL term $\mathrm{KL}[p_T(z)|p_S(z)]$ as a regularizer of the distance between the two domains representations. We argue that there are several reasons that make this a good choice as a divergence between the source and target representation distributions. **(1)** First of all, as mentioned earlier, the KL term can be computed easily without any additional network or a minimax objective (details in Subsection 3.3.3). This leads to

an efficient and stable training procedure, which often results in improved performance. **(2)** Secondly, the reverse KL has the zero-forcing/mode-seeking effect [Minka, 2005] that helps to alleviate the out-of-distribution problem. Specifically, the reverse KL forces the target representation distribution to have zero mass wherever the source distribution has zero mass (zero-forcing), thus preventing the out-of-distribution data at test time (Figure 3.2c). On the other hand, its mode-seeking nature allows flexible alignment of the representation. For example, consider the case where the source domain is a mixture of two components (Figure 3.2a, i.e., it has two modes), and the target distribution is close to one of the two components. Ideally, we want to learn a representation network that matches the representation of the target domain to that of the corresponding component of the source mixture (Figure 3.2b). This representation will still perform well at test time since we would not have the out-of-distribution problem (the classification network is already trained on this mode of the source distribution). This flexible alignment (to one or some of the modes) is accepted by the reverse KL since it leads to a relatively small reverse KL value. Meanwhile, other methods such as DANN, MMD, CORAL, and WD aim to match the representation distribution of the target domain and that of the whole source domain together, which could compress the representation too much, negatively affecting its expressive power. For instance, in the above example, trying to match the whole distribution of source and target domains based on other distance metrics might force the two modes of the source domain to collapse. The flexible alignment of the reverse KL (while still being very effective in preventing out-of-distribution data) might be beneficial in some practical cases.

We also empirically found that adding an auxiliary term $\mathrm{KL}[p_S(z)|p_T(z)]$ (forward KL) with a small coefficient β_{aux} to the objective can help to align the distribution faster, leading to the objective:

$$l_{train} + \beta \mathrm{KL}[p_T(z)|p_S(z)] + \beta_{aux} \mathrm{KL}[p_S(z)|p_T(z)] \tag{3.5}$$

In practice, setting β_{aux} to a (very) small value or zero often leads to the best results (using larger β_{aux} can hurt the performance). Note that this does not invalidate our earlier discussion, as with such small values of β_{aux}, the alignment behavior is still dominated by the reverse KL.

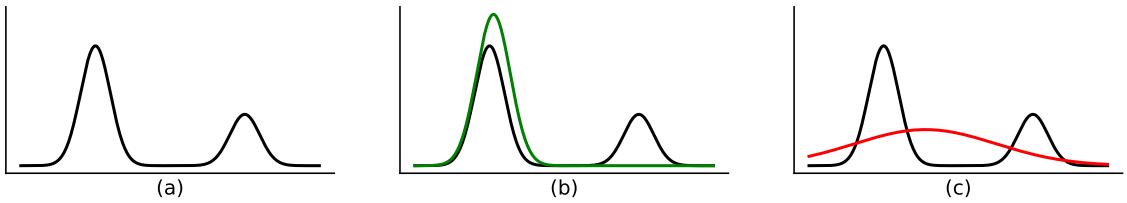


Figure 3.2: **Reverse KL** allows a flexible alignment of the representation while still effectively preventing the out-of-distribution problem. **(a)** Source representation distribution (black). Consider the case where the data distribution $p_S(x)$ of the source domain has two modes, then the representation distribution $p_S(z)$ will likely also have two modes; and consider the case where the target distribution has only one mode. **(b)** An acceptable target representation distribution (green) that helps the classifier avoid the out-of-distribution problem. Reverse KL allows for this type of flexible alignment (match to one/some of the modes) due to its mode-seeking nature. **(c)** A problematic target representation distribution (red), since the classification network will face out-of-distribution data at test time, in the area between the two modes. Reverse KL will prevent this due to its zero-forcing nature.

3.3.3 Optimization

In practice, we estimate Eq. 3.5 using minibatches. In particular, given a labeled minibatch $(\tilde{x}_S^{(i)}, \tilde{y}_S^{(i)})_{i=1}^B$ of the source domain and an unlabelled one $(\tilde{x}_T^{(i)})_{i=1}^B$ of the target domain, and a single sampled representation for each x: $(\tilde{z}_S^{(i)})_{i=1}^B$ and $(\tilde{z}_T^{(i)})_{i=1}^B$, we can get an unbiased estimator of the objective 3.5 as follows:

$$
\begin{aligned}
& l_{train} + \beta \mathrm{KL}[p_T(z)|p_S(z)] + \beta_{aux} \mathrm{KL}[p_S(z)|p_T(z)] \\
=& \mathbb{E}_{p_S(z,y)}[-\log \hat{p}(y|z)] + \beta \mathbb{E}_{p_T(z)}[\log p_T(z) - \log p_S(z)] \\
& \quad + \beta_{aux} \mathbb{E}_{p_S(z)}[\log p_S(z) - \log p_T(z)] \\
\approx& \frac{1}{B} \sum_{i=1}^{B} -\log \hat{p}(\tilde{y}_S^{(i)}|\tilde{z}_S^{(i)}) + \beta \frac{1}{B} \sum_{i=1}^{B} \left[\log p_T(\tilde{z}_T^{(i)}) - \log p_S(\tilde{z}_T^{(i)})\right] \\
& \quad + \beta_{aux} \frac{1}{B} \sum_{i=1}^{B} \left[\log p_S(\tilde{z}_S^{(i)}) - \log p_T(\tilde{z}_S^{(i)})\right]
\end{aligned} \tag{3.6}
$$

However, it still requires knowing $p_S(z)$ and $p_T(z)$ to compute Eq. 3.6. We also use the minibatch to approximate these quantities:

$$
p_S(z) = \mathbb{E}_{p_S(x)}[p(z|x)] \approx \frac{1}{B} \sum_{i=1}^{B} p(z|x_S^{(i)});\ p_T(z) = \mathbb{E}_{p_T(x)}[p(z|x)] \approx \frac{1}{B} \sum_{i=1}^{B} p(z|x_T^{(i)}). \tag{3.7}
$$

Intuitively, we use a minibatch of data to construct a distribution of the representation z (which is a mixture of B components) and match that distribution for the two domains

with the KL divergence. As mentioned earlier, we use a Gaussian distribution with a diagonal covariance matrix for $p(z|x)$ in practice and employ the reparameterization trick [Kingma and Welling, 2014] to sample z.

Although the estimator in Eq. 3.6 is unbiased, the approximations in Eq. 3.7 will introduce some bias into our estimator (however, the estimator is still consistent, i.e., it becomes exact when $B \to \infty$). Therefore, the batch size might have an effect on the performance of the model. However, via an ablation study, we found that the effect of this biased estimator is not severe in practice, and our model achieves good performance even with a batch size of 64. For detailed results and discussion of this ablation study, please refer to Appendix B.3.4.

3.4 Experiments

3.4.1 Datasets

RotatedMNIST consists of 70,000 MNIST [LeCun et al., 2010] images that are divided into six domains, each with 11,666 images. The images in each domain are rotated counter-clockwise by $0°, 15°, 30°, 45°, 60°$ and $75°$ respectively. We denote the six domains as $\mathcal{M}_0, \mathcal{M}_{15}, \mathcal{M}_{30}, \mathcal{M}_{45}, \mathcal{M}_{60}$ and $\mathcal{M}_{75}$. We use $\mathcal{M}_0$ as the source domain, and perform five experiments, each with $\mathcal{M}_{15}, \mathcal{M}_{30}, \mathcal{M}_{45}, \mathcal{M}_{60}$ or $\mathcal{M}_{75}$ as the target domain. The task is classification of the ten digit labels.

DIGITS is a common domain adaptation dataset, with 3 digit classification sub-datasets, namely MNIST, USPS [Hull, 1994] and SVHN [Netzer et al., 2011]. Three common adaptation experiments are MNIST $\to$ USPS, USPS $\to$ MNIST, and SVHN $\to$ MNIST.

VisDA17 [Peng et al., 2017] is a challenging real-world classification dataset with a simulation-to-real adaptation task. This dataset contains over 280K images from 12 classes. The source domain contains renderings of 3D models, while the target domain contains real images.

Please refer to Appendix B.3 for results of datasets with more domains such as **PACS** [**Li et al., 2017**].

3.4.2 Baselines

We consider all common marginal alignment methods for domain adaptation as our baselines, including **DANN**, **MMD**, **CORAL** and **WD**. We also consider **ERM** (empirical risk minimization) and its variant **ERM (prob)** (same as ERM but with the probabilistic representation network used in our model). For ERM, DANN, MMD, and CORAL, we follow the implementation by Gulrajani and Lopez-Paz [2021]; while for ERM (prob) and WD, we use our own implementation in Pytorch [Paszke et al., 2019]. Note that we do not include methods that are not from the marginal-alignment family since they are out of the scope of this work. For the full description of these baselines, please refer to Appendix B.4.1 and the official code at https://github.com/atuannguyen/KL.

3.4.3 Experimental Setting

For the RotatedMNIST and DIGITS experiments, we use a simple convolutional neural network with four 3×3 convolutional layers (followed by an average pooling layer) as the representation network. For VisDA17, we use a Resnet50 as the representation network. Only the last layer of the representation network differs for a deterministic representation (ERM, DANN, CORAL, MMD, WD) and a probabilistic one (ERM (prob) and KL (ours)). For a representation of size d_z, the last layer's dimension of a deterministic representation network is d_z, while that of a probabilistic network is $2 \cdot d_z$ (d_z for μ and d_z for σ^2). Please refer to Appendix B.4 for the detailed experimental setting (including data split, hyper-parameter tuning for each model, evaluation protocol, etc.)

3.4.4 Results

RotatedMNIST and DIGITS: Table 3.1 and Table 3.2 show the results for the RotatedMNIST and DIGITS experiments. It is clear that in these experiments, aligning the representation between domains does help improve the generalization performance. Among the baselines (DANN, MMD, CORAL, and WD), MMD performs the best, which we attribute to the fact that it does not use a minimax objective, leading to a more stable optimization. Meanwhile, CORAL performs the worst, since it only matches the first two moments of the distributions and might fail to align complex distributions. Our method, KL, largely outperforms the baselines, indicating its effectiveness. Visualization of the representation space also shows that our method

Table 3.1: Rotated MNIST experiments with $\mathcal{M}_0$ as the source domain.

	Target Domain					
Model	$\mathcal{M}_{15}$	$\mathcal{M}_{30}$	$\mathcal{M}_{45}$	$\mathcal{M}_{60}$	$\mathcal{M}_{75}$	Average
ERM	97.5±0.2	84.1±0.8	53.9±0.7	34.2±0.4	22.3±0.5	58.4
ERM (prob)	96.8±0.3	83.2±1.6	51.3±0.9	31.4±1.1	20.7±0.7	56.7
DANN	97.3±0.4	90.6±1.1	68.7±4.2	30.8±0.6	19.0±0.6	61.3
MMD	97.5±0.1	95.3±0.4	73.6±2.1	44.2±1.8	32.1±2.1	68.6
CORAL	97.1±0.3	82.3±0.3	56.0±2.4	30.8±0.2	27.1±1.7	58.7
WD	96.7±0.3	93.1±1.2	64.1±3.3	41.4±7.6	27.6±2.0	64.6
KL (ours)	**97.8±0.1**	**97.1±0.2**	**93.4±0.8**	**75.5±2.4**	**68.1±1.8**	**86.4**

Table 3.2: DIGITS and VisDA17 experiments.

	DIGITS				VisDA17
Model	M → U	U → M	S → M	Average	S → R
ERM	73.1±4.2	54.8±6.2	65.9±1.4	64.6	39.1±0.5
ERM (prob)	70.3±3.2	59.0±8.3	67.6±1.3	65.6	37.2±2.2
DANN	90.7±0.4	91.2±0.8	71.1±0.5	84.3	57.7±1.3
MMD	91.8±0.3	94.4±0.5	82.8±0.3	89.7	62.8±1.1
CORAL	88.0±1.9	83.3±0.1	69.3±0.6	80.2	39.5±4.5
WD	88.2±0.6	60.2±1.8	68.4±2.5	72.3	38.9±4.8
KL (ours)	**98.2±0.2**	**97.3±0.5**	**92.5±0.9**	**96.0**	**70.6±0.5**

aligns the representation better than existing methods. This visualization can be found in Appendix B.3.1.

VisDA17 (Table 3.2): In this challenging dataset, many marginal alignment techniques (CORAL, WD) fail the adaptation task (achieve similar accuracy as the ERM baselines). MMD and KL (ours) are again the best performers, confirming that a stable training objective is beneficial in practice. Our method outperforms all other marginal-alignment approaches significantly, suggesting the effectiveness of the KL divergence in representation alignment.

3.5 Conclusion

In conclusion, in this chapter, we derive a generalization bound of the target loss in the domain adaptation problem using the reverse KL divergence. We then show that with a probabilistic representation, the KL divergence can easily be estimated using Monte Carlo (minibatch) samples, without any additional computation or adversarial objective. By minimizing the KL divergence, we can reduce the generalization bound

and have a better guarantee about the test loss. We also empirically show that our method outperforms relevant baselines with large margins, which we attribute to its simple and stable training procedure and the mode-seeking/zero-forcing nature of the reverse KL. We conclude that KL divergence is very effective as a tool for representation alignment. In general, a limitation of marginal alignment methods (including ours) is that when the conditional distribution changes significantly from the source domain to the target domain, aligning the marginal would not help the target domain's performance. This is also reflected in our generalization bound. For future work, we would want to investigate the use of KL divergence in other types of alignment. For example, we can follow the algorithm in [Kang et al., 2019] to minimize the intra-class distance of the representation across domains and maximize the inter-class distance between them, but using the KL divergence instead of MMD as the distance between representation distributions. Another direction would be using the KL divergence to align the conditional distribution across domains in a multi-source setting.

Chapter 4

FedSR: A Simple and Effective Domain Generalization Method for Federated Learning

Abstract

Federated Learning (FL) refers to the decentralized and privacy-preserving machine learning framework in which multiple clients collaborate (with the help of a central server) to train a global model without sharing their data. However, most existing FL methods only focus on maximizing the model's performance on the source clients' data (e.g., mobile users) without considering its generalization ability to unknown target data (e.g., a new user). In this work, we incorporate the problem of Domain Generalization (DG) into Federated Learning to tackle the aforementioned issue. However, virtually all existing DG methods require a centralized setting where data is shared across the domains, which violates the principles of decentralized FL and hence not applicable. To this end, we propose a simple yet novel representation learning framework, namely FedSR, which enables domain generalization while still respecting the decentralized and privacy-preserving natures of this FL setting. Motivated by classical machine learning algorithms, we aim to learn a **simple representation** of the data for better generalization. In particular, we enforce an L2-norm regularizer on the representation and a conditional mutual information (between the representation and the data given the label) regularizer to encourage the model to only learn essential information (while ignoring spurious correlations such as the background). Furthermore, we provide theoretical

connections between the above two objectives and representation alignment in domain generalization. Extensive experimental results suggest that our method significantly outperforms relevant baselines in this particular problem.

4.1 Introduction

In this chapter, we are interested in the problem of decentralized Federated Learning (FL), where K clients collaborate to jointly train a machine learning model with their local (decentralized) data. For privacy reasons, we consider the setting where it is desirable/preferred that the clients do not share their data with each other. For example, McMahan et al. [2017] bring up an example scenario where mobile devices collaborate to train a model without sending their data to a central server. Another example is that K hospitals train a diagnosis model together but do not want to share their patients' data.

However, a major concern with this machine learning framework is that it often does not account for the possible distribution shift on the target data, a problem commonly referred to as Domain Generalization (DG). Furthermore, and perhaps more problematically, it is not straightforward to incorporate existing DG techniques into this FL setting due to its privacy-preserving nature. Specifically, most domain invariant representation learning methods [Ganin et al., 2016, Sun and Saenko, 2016, Nguyen et al., 2021, Li et al., 2018b, Nguyen et al., 2022b] require sharing/comparing the representation of the data across domains, while meta-learning based methods [Li et al., 2018a, Balaji et al., 2018] require access to data from a pool of multiple source domains (in a central server).

There have been some early attempts to incorporate domain generalization into the FL framework. For example, Liu et al. [2021a] propose a technique tailored for medical image segmentation which allows the clients to share their data in the frequency space with each other. This can be viewed as a form of data leakage and is undesirable in the FL setting we consider. Meanwhile, Zhang et al. [2021] align the representation distribution across domains via a reference distribution (from a generative model). While this does fully preserve data privacy, it can be overly complicated to implement in practice, and the performance is not yet on par with SOTA centralized methods. Nevertheless, this is one of our main baselines in this chapter.

To tackle the above challenges, in this work, we introduce a simple yet effective method for domain generalization in the FL setting. We propose to learn a "simple" representation of the data; specifically, we employ two regularization methods, namely the L2 regularizer on the representation, and the conditional mutual information between the data and the representation given the label. These regularizers are easy to implement and do not require data sharing among clients, thus fully respecting the principles of FL. In particular, they aim to restrict the amount of information the representation can contain, which we hope will help to ignore spurious correlations and lead to better generalization to unseen target domains. With a deeper look, we show that these regularizers act as **implicit** forms of marginal and conditional representation alignment , which also further explains the performance gain of our method. This means that our method attempts to learn a representation whose marginal distribution and conditional distribution (given the label) are invariant across the domains without explicitly comparing those distributions between clients. Our contributions in this work are as follows:

- We propose a simple method, which aims to learn a simple representation of the data for the prediction task, for DG in the FL setting. To this end, we revisit two common regularization methods, namely the conditional mutual information regularizer and the L2-norm regularizer on the representation, to restrict the amount of information it can contain.
- Furthermore, we show theoretically and empirically that these two regularizers are well connected with marginal and conditional alignment in the invariant representation learning framework.
- We validate the effectiveness of our resulting model with a wide range of datasets and model architectures and show that it significantly outperforms existing FL methods in the domain generalization task and can even compete with state-of-the-art centralized methods.

4.2 Related Work

4.2.1 Federated Learning

One of the first and most common FL algorithms is FedAVG [McMahan et al., 2017], which uses several local SGD updates within each communication round to the server

to save communication cost. In the context of DG in the FL setting, this method is equivalent to Empirical Risk Minimization in a centralized setting, and is one of our main baselines. One of the main challenges that FL needs to tackle is the statistical diversity among the clients (i.e., each client has a different local data distribution). A great amount of research [Li et al., 2020a, Zhao et al., 2018, Haddadpour and Mahdavi, 2019, Li et al., 2020b, Khaled et al., 2020, Smith et al., 2017] has been done to tackle this problem (i.e., how to train a shared model to deal with non-i.i.d. data of the source clients in the FL setting). Another line of approaches to deal with this problem is to allow each client to have a personalized model and jointly train the models in a FL way (personalized FL) [Hanzely and Richtárik, 2020, Deng et al., 2020, Arivazhagan et al., 2019, Fallah et al., 2020, Jiang et al., 2019]. Although the above methods tackle the distribution shift problem across the source domains/clients, they still have not dealt with the generalization ability to an unseen target domain – the problem we consider in this chapter.

4.2.2 Domain Generalization

Most of the existing DG methods consider a centralized setting. A predominant and effective approach is to learn a domain-invariant representation [Muandet et al., 2013, Li et al., 2018c,d, Wang et al., 2020, Akuzawa et al., 2019, Ilse et al., 2020, Zhao et al., 2020, Ajakan et al., 2014, Li et al., 2018b, Sun and Saenko, 2016, Shen et al., 2018] (meaning to learn a representation whose marginal distribution and/or conditional distribution given the label are unchanged across the domains), hoping that this representation generalizes better to an unseen target domain. Our method can be viewed as an implicit form of representation alignment, thus more suitable for the FL setting. Another line of approaches for DG is to use the idea of meta-learning [Du et al., 2020, Balaji et al., 2018, Li et al., 2018a]; with the logic being that if a model can adapt among the source domains well, it is more likely to adapt to an unseen target domain. These methods typically require access to a pool of data from source domains (in a centralized server). Zhao and Shen [2022] considers a single-domain domain generalization setting. However, it needs to generate a "fake" target distribution from a source distribution and then maximize the mutual information between the representations of the two domains (source vs fake target). This might be overly complicated when applied at each client level in a FL setting and hence less practical. Peng et al. [2020] consider federated domain adaptation, where each source domain aligns the representation distribution with the target domain by receiving the

representations from this target distribution. This cannot be applied to our problem setting to align the representation of two clients, since one client cannot have access to another source client's data (including the data representation) for privacy reasons.

4.3 Approach

4.3.1 Preliminaries

We first present the problem of Federated Learning and the importance of Domain Generalization in this setting.

Federated Learning: In the conventional FL setting, K clients collaborate with each other to train a global model. Each clients i ($1 \leq i \leq K$) has its own data distribution $p_i(x, y)$, where $x \in \mathcal{X}$ is the input and $y \in \mathcal{Y}$ is its corresponding label, and a dataset with N_i data points: $\mathcal{D}_i = \{(x_i^{(1)}, y_i^{(1)}), ..., (x_i^{(N_i)}, y_i^{(N_i)})\}$. It is often assumed that the data distribution $p_i(x, y)$ changes across the clients (which we will use interchangeably with domains in this chapter). Here, the data distributions $p_i(x, y)$ are sampled from a family $\mathcal{E}$ of distributions ($\mathcal{E}$ can be finite or infinite), i.e., $p_i(x, y) \sim \mathcal{E}$. Our aim is for the clients to jointly train a global model with parameters w (this model will be specified later), with the loss function of a data point (x, y) denoted as $\ell(w; x, y)$.

The global objective function to minimize is:

$$f(w) := \frac{1}{K} \sum_{i=1}^{K} f_i(w), \tag{4.1}$$

where $f_i(w)$ is the local objective function:

$$f_i(w) = \mathbb{E}_{p_i(x,y)}[\ell(w; x, y)] \approx \frac{1}{N_i} \sum_{n=1}^{N_i} \ell(w; x_i^{(n)}, y_i^{(n)}).$$

Note that the objective in Eq. 4.1 can also be a weighted average among the clients. However, since the analysis and discussion are unchanged, we will use this simple average objective in this work.

In most Federated Learning algorithms, the optimization process consists of a number of rounds. Within each round, each (active) client runs its own local minimization of the local objective for several iterations of stochastic gradient descent; then the central

server will collect the newly learned parameters from all (active) clients, average them to form the new global parameter, and broadcast it back to each client at the beginning of next round. By taking multiple steps of the local minimization each round, the algorithm can reduce the communication cost (which is often the bottleneck). For a more detailed description of these FL methods, we refer the readers to [McMahan et al., 2017, Wang and Joshi, 2021, Stich, 2019].

Domain Generalization: In the DG problem, we are more interested in training a model to perform well in an unseen target domain $p_T(x, y) \sim \mathcal{E}$ (which is different from $p_i(x, y)\ \forall i$). Therefore, we hope to minimize not only the expected loss on the source domains/clients $\mathbb{E}_{p_i(x,y)}[\ell(w; x, y)]$ but also the loss on unseen target domains, in either an average or worst-case sense as defined below.

In the average case, we want to minimize this loss over the distribution family $\mathcal{E}$:

$$\mathbb{E}_{p_T \sim \mathcal{E}} \left[\mathbb{E}_{p_T(x,y)}[\ell(w; x, y)] \right].$$

Meanwhile, we might also be interested in the worst-case quantity, i.e., the loss of the domain in which the model performs the worst:

$$\sup_{p_T \in \mathcal{E}} \left[\mathbb{E}_{p_T(x,y)}[\ell(w; x, y)] \right].$$

In a conventional machine learning problem, we usually need to observe more **data points** to learn to generalize better to **unseen data points**. Similarly, in this Domain Generalization problem, we additionally need to observe more **domains** in order to generalize better to **unseen target domains**. However, we typically have a limited number of source domains in these DG problems, so regularization techniques often need to be applied for better generalization. One common technique among those is to learn a domain invariant representation, which means to learn a representation z of x that has the same marginal and/or conditional distribution across the domains. These methods often help to ignore domain-specific spurious correlations and lead to better generalization. We will discuss this in more detail later in the chapter.

Generalization Challenge of FL: From the objective in Eq. 4.1, it is clear that conventional FL methods mainly focus on maximizing the performance (minimizing the loss) on K source clients' data, which is very problematic in many applications.

For example, if K clinical institutes in the US and UK collaborate to train a Covid prediction model, the goal is not only to perform well on their own data distributions but also to generalize well to data from other countries. Similarly, if we use Federated Learning to train a machine learning model with data collected from multiple mobile phone users, it is crucial that the model can generalize to a new user. Thus, domain generalization is very important for Federated Learning.

However, as mentioned earlier, applying existing domain generalization techniques to Federated Learning is **not straightforward**. This is because most existing DG methods need a centralized training setting with data (or a part of the data, such as a representation) being shared across the domains. For example, the aforementioned domain invariant representation learning approach typically requires sharing and comparing the representation distribution across domains. Towards this end, we propose a simple but effective domain invariant representation learning technique for Federated Learning that fully keeps its decentralized and privacy aspects. We first introduce the problem setting of representation learning in Subsection 4.3.2, and then present our domain invariant representation learning method in Subsection 4.3.3.

4.3.2 Problem Setting: Representation Learning for Domain Generalization

In the representation learning framework, we aim to learn a representation z of x with a distribution $p(z|x)$ parameterized by w_1, which we omit here for notation simplicity. This can be a deterministic mapping (i.e., $p(z|x) = \delta_{g_{w_1}(x)}(z)$ where δ is the Dirac delta function) or a probabilistic mapping (e.g., $p(z|x) = \mathcal{N}(z; \mu_{w_1}(x), \sigma^2_{w_1}(x))$ where $\mathcal{N}$ is the normal distribution). With the data distribution $p_i(x, y)$ of client i, its joint distribution of x, y, z is:

$$p_i(x, y, z) = p_i(x, y)p(z|x), \quad \text{since } z \text{ is conditionally independent of } y \text{ given } x$$

From the representation z, we learn a classifier/regressor that predicts y given z with the predictive distribution $\hat{p}(y|z)$ parameterized by w_2 (again, we omit w_2 here for notation simplicity). For example, with a classification problem, this is often just a linear layer followed by a softmax layer to form the predictive distribution.

The predictive distribution of y given x of our model is:

$$\hat{p}(y|x) = \mathbb{E}_{p(z|x)}[\hat{p}(y|z)] \tag{4.2}$$

Note that when $p(z|x)$ is a deterministic mapping, Eq. 4.2 simplifies into $\hat{p}(y|z = g_{w_1}(x))$.

For both regression and classification, the loss function of a data point x, y is often the negative log predictive, i.e, $\ell(w; x, y) = -\log \mathbb{E}_{p(z|x)}[\hat{p}(y|z)]$, and in this case $w = \{w_1, w_2\}$.

Hence, the local objective of client i is:

$$f_i(w) = \mathbb{E}_{p_i(x,y)} \left[-\log \mathbb{E}_{p(z|x)}[\hat{p}(y|z)] \right]$$

During training, for a probabilistic mapping, we often sample a single z per x from $p(z|x)$ (for a deterministic mapping, it is obvious that we only sample one z), leading to the training objective:

$$\begin{aligned} \overline{f_i}(w) &= \mathbb{E}_{p_i(x,y)} \left[\mathbb{E}_{p(z|x)}[-\log \hat{p}(y|z)] \right] \\ &\approx \frac{1}{N_i} \sum_{n=1}^{N_i} -\log \hat{p}(y_i^{(n)}|z_i^{(n)}), \text{ where } z_i^{(n)} \text{ is a single sample from } p(z|x_i^{(n)}) \end{aligned} \quad (4.3)$$

The quantity $-\log \hat{p}(y|z)$ is often non-negative for all common predictive distributions. For example, with a categorical predictive distribution in a classification problem, this is the cross-entropy loss; while with a Gaussian predictive distribution (with a fixed variance) in a regression problem, this becomes the squared error (with an additive constant).

Note that according to Jensen's Inequality, $\overline{f_i}(w)$ is an upper bound of $f_i(w)$, so the training process is sound.

The representation learning framework that we discussed so far still has not dealt with the problem of domain generalization. Many existing works in DG focus on learning a domain-invariant representation, which means to learn a representation z that has the same marginal ($p_i(z)$) and/or conditional ($p_i(z|y)$ or $p_i(y|z)$) distributions across the domains. However, this often requires sharing information of the data (the representation) across domains/clients, which is not allowed in Federated Learning.

Motivated by the fact that classical machine learning algorithms make predictions based on a simple representation of the data and often generalize better than modern deep learning models, in this work, we aim to learn a **S**imple **R**epresentation of the data (hence the name FedSR). We apply several well-known regularization techniques to restrict the representation's complexity and encourage the deep network $p(z|x)$ to

only learn the essential information. Furthermore, we show that these regularization techniques have strong theoretical connections to domain-invariant representation learning and that they can achieve good generalization performance in practice.

4.3.3 FedSR

As mentioned earlier, we employ an L2-norm regularizer (ℓ_i^{L2R}) on the representation and (an upper bound of) the conditional mutual information between the data and the representation given the label (ℓ_i^{CMI}) to restrict the amount of information the representation can contain. Moreover, ℓ_i^{L2R} also has the effect of aligning the marginal distribution $p_i(z)$ to be centered around 0 as a Gaussian distribution, while ℓ_i^{CMI} aligns the conditional distribution $p_i(z|y)$ to a reference distribution (also chosen to be Gaussian in the experiment section). These two terms will be explained in detail below. The final local objective function of each client i is:

$$\overline{f_i} + \alpha^{L2R}\ell_i^{L2R} + \alpha^{CMI}\ell_i^{CMI},$$

where α^{L2R} and α^{CMI} are hyper-parameters. (We will also test the two regularizers separately, namely the FedL2R and FedCMI variants.)

4.3.3.1 L2-norm Regularizer

In optimization and learning theory, the L2-norm of the weights/parameters is often regarded as a measure of the model's complexity. Here, with a similar idea, we enforce an L2-norm Regularizer (L2R) on the representation (not the network parameters) to restrict the complexity of the representation. (Note that some methods like [Mohri et al., 2019] use the L2-norm but in the parameter space to help with the generalization of Federated Learning). Specifically, the regularizer has the form:

$$\begin{aligned}\ell_i^{L2R} &= \mathbb{E}_{p_i(x)}\left[\mathbb{E}_{p(z|x)}[||z||_2^2]\right] \\ &\approx \frac{1}{N_i}\sum_{n=1}^{N_i}||z_i^{(n)}||_2^2, \text{ where } z_i^{(n)} \text{ is a single sample from } p(z|x_i^{(n)}).\end{aligned} \tag{4.4}$$

Intuitively, this regularizer will have the effect of encouraging z to be centered around 0 for all clients, thus helping with the marginal alignment of the representation. This connection can be explained in more detail below:

Connection to Domain Invariant Representation Learning: First of all, rewrite ℓ_i^{L2R} as:

$$\ell_i^{L2R} = \mathbb{E}_{p_i(x)}\left[\mathbb{E}_{p(z|x)}[||z||_2^2]\right] = \mathbb{E}_{p_i(x,z)}\left[||z||_2^2\right] = \mathbb{E}_{p_i(z)}\left[||z||_2^2\right].$$

Now, consider a "reference" distribution $q(z) = \mathcal{N}(0, \sigma^2 I)$ (with a small σ). Then we have:

$$-\log q(z) = \frac{||z||_2^2}{2\sigma^2} \text{ (with an additive constant).}$$

And hence:

$$\ell_i^{L2R} = \mathbb{E}_{p_i(z)}\left[||z||_2^2\right] = 2\sigma^2 \mathbb{E}_{p_i(z)}\left[-\log q(z)\right] = 2\sigma^2 H(p_i(z), q(z)),$$

where $H(p_i(z), q(z))$ denotes the cross entropy from $q(z)$ to $p_i(z)$.

Note further that:

$$H(p_i(z), q(z)) = H(p_i(z)) + \mathrm{KL}[p_i(z)|q(z)].$$

Hence, if the entropy $H(p_i(z))$ does not change significantly during training, minimizing ℓ_i^{L2R} will also minimize $\mathrm{KL}[p_i(z)|q(z)]$, which encourages $p_i(z)$ to be close to $q(z)$, i.e., an implicit alignment of the marginal distribution.

Nevertheless, also according to the above equation, if we really want to minimize $\mathrm{KL}[p_i(z)|q(z)]$ (for the marginal alignment), it is perhaps more rigorous to minimize the regularizer $\ell_i^{L2R} - 2\sigma^2 H(p_i(z))$ instead of ℓ_i^{L2R}. However, when σ is small enough, this term is dominated by ℓ_i^{L2R} (meaning that if σ is small, minimizing ℓ_i^{L2R} almost guarantees to minimize $\mathrm{KL}[p_i(z)|q(z)]$). Empirically, we found that the two variants achieve almost identical performance. We therefore recommend using ℓ_i^{L2R} instead of $\ell_i^{L2R} - 2\sigma^2 H(p_i(z))$ for its simplicity.

4.3.3.2 Conditional Mutual Information regularizer

The mutual information term between the input x and the representation z (denoted $I(x, z)$) is often used to regularize the amount of information z can contain [Alemi et al., 2017, Tishby et al., 2000]. However, this regularizer might be too restrictive in practice if its coefficient is not tuned properly, as it encourages the representation to contain no information about the input data. In this work, we use the **conditional mutual information** of x and z given y, as it is less restricted than the former. We can also

later see that this is well-connected with conditional distribution alignment in DG. Recall that this conditional mutual information term is calculated for domain/client i as:

$$I_i(x, z|y) = \mathbb{E}_{p_i(x,y,z)} \left[\log \frac{p_i(x, z|y)}{p_i(x|y)p_i(z|y)} \right].$$

Intuitively, the terms $\overline{f_i}$ and $I_i(x, z|y)$ work together to enforce the representation z to contain only the information needed to predict the label y, and no additional information (non-label-related) about x.

However, unfortunately, this mutual information term is not tractable because of $p_i(z|y)$ (which is hard to compute due to the integration over x). Therefore, we derive an upper bound in order to minimize it. Specifically:

Lemma 4.1. *Let $r(z|y)$ be a(ny) conditional distribution of z given y $\forall y \in \mathcal{Y}$. If $p(z|x)$ and $r(z|y)$ have the same support set $\forall x \in \mathcal{X}, y \in \mathcal{Y}$, we have:*

$$I_i(x, z|y) \leq \mathbb{E}_{p_i(x,y)} \left[\mathrm{KL}[p(z|x)|r(z|y)] \right] = \ell_i^{CMI}. \tag{4.5}$$

Proof. Provided in Appendix C.1.1. □

Here, the upper bound ℓ_i^{CMI} can be computed and used as a regularizer when training the representation network. We optimize both $p(z|x)$ and $r(z|y)$ in order to minimize ℓ_i^{CMI}. Note that $r(z|y)$ can be a network that takes y as an input and outputs the distribution $r(z|y)$; or more simply for classification (when we have a finite number of labels y), we can set $r(z|y) = \mathcal{N}(z; \mu_y, \sigma_y^2)$, where μ_y, σ_y^2 ($y = \overline{1..C}$) are the parameters to be optimized, with C being the number of classes.

Note also that if we use this (upper bound) regularizer ℓ_i^{CMI}, it requires the representation mapping $p(z|x)$ to be probabilistic (due to the KL term). Hence, in the experimental section, we use a probabilistic representation network ($p(z|x) = \mathcal{N}(z; \mu_{w_1}(x), \sigma_{w_1}^2(x))$) for any models/variants with the ℓ_i^{CMI} objective, and use a deterministic network otherwise. With the empirical dataset $\mathcal{D}_i$ of client i, we have an empirical estimator of ℓ_i^{CMI} as below:

$$\begin{aligned} \ell_i^{CMI} &= \mathbb{E}_{p_i(x,y)} \left[\mathrm{KL}[p(z|x)|r(z|y)] \right] \\ &\approx \frac{1}{N_i} \sum_{n=1}^{N_i} \mathrm{KL}[p(z|x_i^{(n)})|r(z|y_i^{(n)})] \end{aligned} \tag{4.6}$$

Connection to Domain Invariant Representation Learning: We found that this regularizer has a theoretical connection with conditional distribution alignment in domain generalization. We start with the following lemma:

Lemma 4.2.

$$\ell_i^{CMI} = \mathbb{E}_{p_i(x,y)} \left[\mathrm{KL}[p(z|x)|r(z|y)] \right] \geq \mathbb{E}_{p_i(y)} \left[\mathrm{KL}[p_i(z|y)|r(z|y)] \right].$$

Proof. Provided in Appendix C.1.2. □

Therefore, by minimizing ℓ_i^{CMI}, we also minimize the divergence $\mathrm{KL}[p_i(z|y)|r(z|y)]\ \forall y$, forcing $p_i(z|y)$ and $r(z|y)$ to be close to each other: $p_i(z|y) \approx r(z|y)$. Therefore, the model will try to enforce $p_i(z|y) \approx p_j(z|y) (\approx r(z|y))\ \forall$ clients/domains i, j, i.e., we are doing implicit alignment of the conditional distribution of the representation z given the label y, which is a common and effective technique in conventional Domain Generalization. For example, Li et al. [2018d], Li et al. [2018c], and Zhao et al. [2020] explicitly align this conditional distribution (not suitable for Federated Learning).

4.3.3.3 Implementation Choices and Optimization

For the federated optimization, we use FedAVG, which is one of the first and perhaps most common FL optimization algorithms (although we can use any federated optimization techniques).

Following common practice, we use mini-batches to approximate Eq. 4.3, Eq. 4.6, and Eq. 4.4 (instead of full-batch training).

For a probabilistic network (when using ℓ^{CMI}), to allow the gradient to backpropagate through the samples $z_i^{(n)}$ to the network parameters w_1, we use the reparameterization trick [Kingma and Welling, 2014]. With this reparameterization trick, $\overline{f_i}$ and ℓ_i^{L2R} can be implemented and back-propagated straightforwardly.

Regarding the conditional mutual information regularizer ℓ_i^{CMI}, we use a Gaussian distribution for the variational conditional distribution $r(z|y)$, hence the KL term in Eq. 4.6 can be computed analytically (recall that we also use a Gaussian distribution for $p(z|x)$). Here the astute readers might wonder that such a simple choice of the variational distribution $r(z|y)$ might not be able to approximate $p_i(z|y)$. However, note that since both $p(z|x)$ and $r(z|y)$ are optimized in order to minimize ℓ_i^{CMI}, it encourages the network to learn a simple representation so that the variational distribution $r(z|y)$ can match $p_i(z|y)$ (which arguably might be a good regularization

effect). Furthermore, depending on datasets, we can, in theory, use more complex distributions such as a Gaussian mixture as the variational distribution. In such cases, one can use methods such as MC sampling to estimate the KL term. In our experiment section, for simplicity, we only use a Gaussian distribution for $r(z|y)$.

4.4 Experiments

4.4.1 Datasets

To evaluate our method, we perform experiments in four datasets (ranging from easy to more challenging) that are commonly used in the literature for domain generalization.

RotatedMNIST [Ghifary et al., 2015]: In this dataset, the MNIST images [LeCun et al., 2010] are rotated counter-clockwise with an angle of 0°, 15°, 30°, 45°, 60° and 75° to form six domains $\mathcal{M}_0$, $\mathcal{M}_{15}$, $\mathcal{M}_{30}$, $\mathcal{M}_{45}$, $\mathcal{M}_{60}$ and $\mathcal{M}_{75}$. The task is classification with ten classes (digits 0 to 9). We use the same version of the dataset as in [Nguyen et al., 2021, Ilse et al., 2020] (where only 1000 images are rotated to form the domains) for easy comparison. Note that this is slightly different from the version in [Gulrajani and Lopez-Paz, 2021] and thus the numbers are not comparable.

PACS [Li et al., 2017]: contains 9,991 images from four different domains: art painting, cartoon, photo, sketch. The task is classification with seven classes.

OfficeHome [Venkateswara et al., 2017]: has 15,500 images of daily objects from four domains: art, clipart, product, and real. There are 65 classes in this classification dataset.

DomainNet [Peng et al., 2019] is a large-scale dataset, consisting of 586,575 images from 345 classes. These images are from 6 domains: clipart, infograph, painting, quickdraw, real, sketch.

4.4.2 Experimental Setting

For all datasets, we perform "leave-one-domain-out" experiments, where we choose one domain as the target domain, train the model on all remaining domains, and evaluate it on the chosen domain. Each source domain is treated as a client. Following standard practice, we use 90% of available data as training data and 10% as validation data. We train all our models with NVIDIA A100 GPUs from our AWS cluster.

Table 4.1: **RotatedMNIST**. Reported numbers are from 3 runs

Models	Domains						Average
	$\mathcal{M}_0$	$\mathcal{M}_{15}$	$\mathcal{M}_{30}$	$\mathcal{M}_{45}$	$\mathcal{M}_{60}$	$\mathcal{M}_{75}$	
HIR	90.34	99.75	99.40	96.17	99.25	91.26	96.03
DIVA	93.5	99.3	99.1	99.2	99.3	93.0	97.2
DGER	90.09	99.24	99.27	99.31	99.45	90.81	96.36
DIRT-GAN	97.2	99.4	99.3	99.3	99.2	97.1	**98.6**
FedAVG	85.9±0.6	98.7±0.2	98.8±0.2	98.9±0.1	98.7±0.2	86.2±0.7	94.5
FedADG	89.9±0.5	98.9±0.2	99.0±0.1	99.1±0.1	98.9±0.2	90.2±0.4	96.0
FedCMI (ours)	91.5±0.3	99.2±0.1	99.3±0.1	99.2±0.1	99.2±0.1	91.0±0.3	96.5
FedL2R (ours)	90.5±0.5	99.1±0.2	99.2±0.1	99.1±0.1	99.2±0.1	90.7±0.6	96.3
FedSR (ours)	91.6±0.3	99.3±0.1	99.3±0.1	99.2±0.1	99.3±0.1	91.5±0.3	**96.7**

Table 4.2: **PACS**. Reported numbers are from 3 runs

Models	Backbone	PACS				
		A	C	P	S	Average
DGER	Resnet18	80.70	76.40	96.65	71.77	81.38
DIRT-GAN	Resnet18	82.56	76.37	95.65	79.89	**83.62**
FedAVG	Resnet18	77.8±0.5	72.8±0.4	91.9±0.5	78.8±0.3	80.3
FedADG	Resnet18	77.8±0.5	74.7±0.4	92.9±0.3	79.5±0.4	81.2
FedCMI (ours)	Resnet18	80.8±0.4	73.7±0.2	92.8±0.5	79.5±0.2	81.7
FedL2R (ours)	Resnet18	82.2±0.4	75.8±0.3	92.8±0.4	81.6±0.1	83.1
FedSR (ours)	Resnet18	83.2±0.3	76.0±0.3	93.8±0.5	81.9±0.2	**83.7**

For the RotatedMNIST dataset, we use a network of two 3x3 convolutional layers and a fully connected layer as the representation network to get a representation z of 64 dimensions. A single linear layer is then used to map the representation z to the ten output classes. This architecture is the same as the network used by [Ilse et al., 2020, Nguyen et al., 2021]. We train our network for 500 epochs with stochastic gradient descent (SGD), using a learning rate of 0.001 and minibatch size 64, and report performance on the test domain after the last epoch. Each client performs 5 local optimization iterations within each communication round ($E = 5$).

For the PACS datasets, for easy comparison with existing centralized domain invariant representation learning methods, we use the most common choice of backbone network in existing works as the representation network, i.e., Resnet18 [He et al., 2016]. We use a ResNet50 [He et al., 2016] backbone for OfficeHome and DomainNet since they are more complex datasets. We replace the last fully connected layer of the backbone with a linear layer of dimension 512 for ResNet18 and 2048 for ResNet50

to form the representation network. As with the RotatedMNIST experiment, we use a single layer to map from the representation z to the output. Each local client uses stochastic gradient descent (SGD) for a total of 5000 iterations, with learning rate 0.01, momentum 0.9, minibatch size 64, and weight decay $5e^{-4}$. Similar to the RotatedMNIST experiments, each client performs 5 local optimization iterations within each communication round ($E = 5$). However, we also found that our method is not sensitive to this hyper-parameter (performance with $E = 100$ is almost identical). For the hyper-parameters of our method (β_{CMI} and β_{L2R}), we follow Gulrajani and Lopez-Paz [2021] and use random search to tune their values based only on the validation set (no data leakage from the target domain). This protocol is highly recommended by Gulrajani and Lopez-Paz [2021] and allows for a fair comparison among the methods. For details of the tuned values, please refer to Appendix C.3.1. Data augmentation is also standard practice for real-world computer vision datasets like PACS, OfficeHome, and Domainet, and during training, we augment our data as follows: crops of random size and aspect ratio, resizing to 224 $\times$ 224 pixels, random horizontal flips, random color jitter, randomly converting the image tile to grayscale with 10% probability, and normalization using the ImageNet channel means and standard deviations. Our code is released at https://github.com/atuannguyen/FedSR.

4.4.3 Baselines

For all experiments, we consider the FL algorithms to be our main baselines: FedAVG [McMahan et al., 2017] (a common FL algorithm) and FedADG [Zhang et al., 2021] (an existing FL algorithm with DG ability). For the RotatedMNIST and PACS experiments, we also include SOTA centralized domain-invariant methods for comparison (the numbers are taken directly from the original papers). These centralized methods include HIR [Wang et al., 2020], DIVA [Ilse et al., 2020], DGER [Zhao et al., 2020], and DIRT-GAN [Nguyen et al., 2021] for RotatedMNIST; and DGER [Zhao et al., 2020] and DIRT-GAN [Nguyen et al., 2021] for PACS. For the OfficeHome and DomainNet experiments, since our setting (e.g., backbone network) and evaluation protocol are identical to those of DomainBed [Gulrajani and Lopez-Paz, 2021], our results can be compared directly to the results of the centralized methods recorded in that paper. Therefore, we also include the best performers for each dataset reported in the DomainBed paper, namely Mixup [Xu et al., 2020], CORAL [Sun and Saenko, 2016], and MLDG [Li et al., 2018a]. Note that we include these centralized methods (in the first half of each table) for reference only, and they are not direct competitors

Table 4.3: **OfficeHome**. Reported numbers are from 3 runs

		OfficeHome				
Models	Backbone	A	C	P	R	Average
Mixup	Resnet50	64.7	54.7	77.3	79.2	**69.0**
CORAL	Resnet50	64.4	55.3	76.7	77.9	68.6
FedAVG	Resnet50	62.2±0.9	55.6±0.9	75.7±0.2	78.2±0.2	67.9
FedADG	Resnet50	63.2±0.9	57.0±0.2	76.0±0.1	77.7±0.5	68.4
FedCMI (ours)	Resnet50	61.8±0.5	55.5±0.9	76.3±0.1	77.4±0.1	67.8
FedL2R (ours)	Resnet50	64.5±0.3	56.5±0.5	76.1±0.2	77.9±0.2	68.8
FedSR (ours)	Resnet50	65.4±0.5	57.4±0.2	76.2±0.6	78.3±0.3	**69.3**

Table 4.4: **DomainNet**. Reported numbers are from 3 runs

		DomainNet						
Models	Backbone	C	I	P	Q	R	S	AVG
MLDG	Resnet50	59.5	19.8	48.3	13.0	59.5	50.4	**41.8**
CORAL	Resnet50	58.7	20.9	47.3	13.6	60.2	50.2	**41.8**
FedAVG	Resnet50	59.3±0.7	16.5±0.9	44.2±0.7	10.8±1.8	57.2±0.8	49.8±0.4	39.6
FedADG	Resnet50	60.9±0.6	17.2±0.2	44.3±0.2	12.4±0.2	57.6±0.9	50.3±0.8	40.4
FedCMI (ours)	Resnet50	59.0±0.9	18.0±0.7	44.6±0.5	12.2±0.4	56.2±0.2	50.0±0.4	40.0
FedL2R (ours)	Resnet50	60.2±0.6	18.1±0.4	44.9±0.6	11.0±0.9	57.8±0.4	51.5±0.7	40.6
FedSR (ours)	Resnet50	61.0±0.6	18.6±0.4	45.2±0.5	13.4±0.6	57.6±0.2	51.8±0.3	**41.3**

of our model(s). For this reason, we highlight (in bold) the best performant model for the centralized and FL settings separately.

4.4.4 Results

Tables 4.1, 4.2, 4.3 and 4.4 show that our method (FedSR) outperforms the baseline FedAVG significantly (by a margin of 1.5% to 3% on average) and consistently in all experiments. FedADG [Zhang et al., 2021] (an existing method for DG in the FL setting) does outperform FedAVG but is not yet on par with centralized methods. Meanwhile, FedSR achieves competitive results when compared to state-of-the-art centralized DG methods (as listed in the first half of the tables). This indicates the effectiveness of our method and the potential for Domain Generalization in the FL setting. As mentioned earlier, our method can be interpreted as an implicit form of representation alignment. Please refer to Appendix C.3.2 for the visualization of our representation space, which does show the alignment effect of our method.

4.5 Conclusion

To conclude, in this work, we present a simple and effective method for implicit alignment of the representation across domains/clients in a Federated Learning setting for a better domain generalization ability. We employ two regularization techniques, namely CMI and L2R, to learn a simple representation of the data, hoping for a better generalization ability. Furthermore, we also show that these regularizers implicitly align the marginal and conditional distribution of the representation, which are shown to be effective in the domain generalization problem. Extensive experiments show that our method outperforms relevant FL baselines (FedAVG, FedADG) and can even achieve competitive DG performance when compared to centralized approaches. A potential limitation of our method is that the invariance is enforced among the source domains and might fail to generalize to the unseen target domain (since the target domain can be arbitrary). However, this is a common comment for all domain invariant representation methods and not specific to ours. Furthermore, experiments show that our method does lead to better representation alignment and prediction performance in practice.

Chapter 5

TIPI: Test Time Adaptation with Transformation Invariance

Abstract

When deploying a machine learning model to a new environment, we often encounter the distribution shift problem – meaning the target data distribution is different from the model's training distribution. In this work, we assume that labels are not provided for this new domain, and that we do not store the source data (e.g., for privacy reasons). It has been shown that even small shifts in the data distribution can affect the model's performance severely. Test Time Adaptation offers a means to combat this problem, as it allows the model to adapt during test time to the new data distribution, using only unlabeled test data batches. To achieve this, the predominant approach is to optimize a surrogate loss on the test-time unlabeled target data. In particular, minimizing the prediction's entropy on target samples [Wang et al., 2021] has received much interest as it is task-agnostic and does not require altering the model's training phase (e.g., does not require adding a self-supervised task during training on the source domain). However, as the target data's batch size is often small in real-world scenarios (e.g., autonomous driving models process every few frames in real-time), we argue that this surrogate loss is not optimal since it often collapses with small batch sizes. To tackle this problem, in this work, we propose to use an invariance regularizer as the surrogate loss during test-time adaptation, motivated by our theoretical results regarding the model's performance under input transformations.

The resulting method (TIPI – Test tIme adaPtation with transformation Invariance) is validated with extensive experiments in various benchmarks (Cifar10-C, Cifar100-C, ImageNet-C, DIGITS, and VisDA17). Remarkably, TIPI is robust against small batch sizes (as small as 2 in our experiments), and consistently outperforms TENT [Wang et al., 2021] in all settings. Our code is available at https://github.com/atuannguyen/TIPI.

5.1 Introduction

Distribution shift is a common problem and is often faced in real-world applications. Specifically, despite taking various precautions while training a machine learning model to ensure a better generalization (e.g., collecting and training on multiple source domains [Li et al., 2018d, Nguyen et al., 2021], finding flat minima [Foret et al., 2021a], training with meta-learning objectives [Li et al., 2018a], etc.), the model often still struggles when the test data distribution shifts slightly. Note that it is also common not to have labels for the new shifted domain during test time. To tackle this problem, test time adaptation (also known as online domain adaptation) is a framework that allows the model to adapt to the target distribution using unlabeled test data batches. This is necessary since we typically do not have time to annotate the data during test time and can only make use of the unlabeled data. In this framework, the model needs to give predictions for the target data while simultaneously updating itself to improve its performance on that particular target distribution. We assume a situation in which the target data only arrive in small batches, which makes the adaptation task extremely challenging and renders traditional domain adaptation techniques such as representation alignment (via a distance metric) ineffective.

Within this test time adaptation framework, a common and effective approach is to optimize a surrogate objective function in lieu of the true loss function on the target data. The first group of surrogate objectives is the loss functions of self-supervised tasks. In particular, one would formulate a user-defined task (such as predicting the rotation angle of an image) and train it alongside the main task on the source domain; and keep training the self-supervised task on the test-time target data [Sun et al., 2020, Liu et al., 2021b]. However, these are not fully test-time adaptation methods, since they require altering the training procedure of the source domain. The second line of surrogate objectives is unsupervised loss functions. Among this group of unsupervised objectives, entropy minimization (TENT) [Wang et al., 2021] is the

most successful method and has been shown to be consistent across many benchmarks. Furthermore, different from the former group, TENT is a fully test-time adaptation method. For these reasons, TENT has received much interest and a lot of follow-up papers/discussions.

However, as pointed out by its authors, TENT is not robust when using small batch sizes, as it often collapses to a trivial solution (i.e., it always predicts the same class for all input). This is detrimental to real-world applications since test data often arrive in small batches. For example, autonomous driving systems only process a few frames in real time – they typically do not accumulate the frames (let's say within one minute) to form a bigger batch. Note that TENT is previously evaluated mainly for large batch sizes (such as 64, 128, or 200).

In this work, we aim to tackle the aforementioned problem. Specifically, we aim to develop an unsupervised surrogate objective function for the test time adaptation problem such that it is task-agnostic, does not require altering the training procedure (e.g., does not require incorporating a self-supervised task into the training process), and is more resilient against small batch sizes.

We first provide theoretical results regarding a model's performance under input transformations. Specifically, we show that a model's loss on a data distribution is bounded by the KL distance on the predictive distribution of the data before and after the transformations (which we will use as a regularizer), and its loss on the transformed data distribution. Motivated by this result, we use small shifts in the input images that can simulate real source-target shifts, and enforce the network to be invariant under such data transformations. Our model outperforms TENT (and other relevant baselines) in all problem settings considered in this chapter and is remarkably robust in the small-batch-size regime.

Our contributions in this work are threefold:

- We provide theoretical results regarding a model's performance under transformations of the input data. Specifically, a model's performance on the target domain is bounded by an invariance term (maximum KL divergence of the predictive distributions before and after the transformations) and its loss on the transformed domain.

- We propose to find input transformations that can simulate the domain shifts, and enforce the network to be invariant under such transformations, using a

regularizer based on our derived bound. The resulting method is **TIPI** (**T**est t**I**me ada**P**tation with transformation **I**nvariance).

- We perform extensive experiments on a wide range of datasets (Cifar10-C, Cifar100-C, ImageNet-C, DIGITS, and VisDA17) and settings (varying batch sizes) to validate our method. TIPI shows preferable performance compared to relevant baselines.

5.2 Related Works

Distribution Shift: is an important and commonly faced problem when deploying machine learning models in real-world applications. There are two main approaches that try to alleviate this problem. The first setting is domain generalization [Muandet et al., 2013, Li et al., 2018d, Nguyen et al., 2021, Balaji et al., 2018, Nguyen et al., 2022a], where one aims to collect multiple source domains and learn a generalizable model based on those domains. Another setting is (unsupervised) domain adaptation [Ben-David et al., 2010, Mansour et al., 2009, Ganin et al., 2016, Nguyen et al., 2022b, Liang et al., 2020], where a model is trained on a labeled source dataset and we attempt to adapt that model to a target distribution with the help of unlabeled target data. This chapter focuses on the latter setting, in particular, the test-time domain adaptation variation.

Test Time Domain Adaptation: In this setting, the unlabeled target data is not available during the training phase of the source model. Instead, the model only has access to the target data when deployed in the target environment, as the stream of test data batches. As a consequence, the unlabeled target data is not available in large quantities, but only in small batches. This makes the adaptation task extremely challenging. So far, there are two main lines of work dealing with this online adaptation problem. The first approach is based on the batch normalization layer in the network, with the idea of using the test-time batch statistics for these layers instead of the source statistics [Schneider et al., 2020, Nado et al., 2020]. This can partially help with the representation misalignment between the source and target distributions but is only applicable when the network has BatchNorm layers and the batch size is large. The other line of work is to use an unsupervised surrogate loss to "refine" the model on the target data. For example, TENT [Wang et al., 2021] minimizes the prediction entropy of the model on the target data. However, as pointed out by its authors, TENT needs

a large batch size to avoid collapsing to a trivial prediction (which outputs only one class), due to the characteristics of this surrogate objective. Meanwhile, TTT [Sun et al., 2020] and TTT+++ [Liu et al., 2021b] train a self-supervised task alongside the main task on the source domain, and continue to train the self-supervised task during the online adaptation phase. Therefore, TTT and TTT++ are not fully off-the-shelf test-time adaptation methods, since they require modifications to the training phase of the source domain (to incorporate the self-supervised task). Recently, Goyal et al. [2022] generalize TENT to other source loss functions (beyond the typical cross-entropy loss). There also have been several works [Niu et al., 2022] that study data points selection strategies for optimizing the unsupervised surrogate loss (e.g., TENT): only optimizing the loss for reliable and informative data points. Note that this line of research is orthogonal/complementary to ours and we believe that our surrogate loss will also benefit from similar data point selection strategies. We leave exploring this direction to future work. Using a completely different approach, T3A [Iwasawa and Matsuo, 2021] replaces the last linear layer (classification layer) with a nonparametric classifier and keeps updating the support set of each class with the new target data points. T3A is robust against small batch sizes (since it is not optimization-based), but is not as effective as other optimization-based methods (e.g., TENT, TTT, and ours).

Connection between Adversarial Robustness and Robustness against Distribution shift: It has been observed that adversarially trained models generally perform better under certain types of domain shifts (for example, see the leaderboard of Robustbench [Croce et al., 2021] for Cifar10-C and Cifar100-C). Indeed, one might expect this to be the case if the space of adversarial attacks matches the space of distribution shifts. While it is not previously well understood, our work sheds some light on this phenomenon. Note that different from previous works, our method enforces an unsupervised robustness loss during the online adaptation phase, not at the source training stage.

5.3 Approach

5.3.1 Problem setting

Assume that we have a source domain with the distribution $p_S(x, y)$, where $x \in \mathcal{X}$ is the input and $y \in \mathcal{Y}$ is the prediction target. Similar to prior works, we consider

the classification problem, where $\mathcal{Y} = \{1, 2, \ldots, C\}$ (however, the approach can be straightforwardly extended to regression problems). Suppose that using data from the source domain, we already successfully trained a model $f_\theta : \mathcal{X} \to \mathbb{R}^C$, where θ is the parameter of the neural network and $f_\theta(x)$ outputs the prediction probabilities for the C classes ($\|f_\theta(x)\|_1 = 1$ and $0 \leq f_\theta(x)_i \leq 1, \forall i \in \{1, 2, \ldots, C\}$). This leads to a predictive distribution $\hat{p}_\theta(y|x)$, which is the categorical distribution where $\hat{p}_\theta(y = i|x) = f_\theta(x)_i$. Note that, for our method, we will not assume or require any modifications to the training process of the model f_θ on the source domain – a setting that is common and useful (e.g., a Resnet50 model downloaded directly from `torchvision`).

After that, this model is deployed to the target environment with data distribution $p_T(x, y)$. The model receives target data as a stream of unlabeled mini-batches (probably with small and varying sizes). The task is to adapt to this environment simultaneously and continually with only the unlabeled mini-batches.

The expected loss of this model on a data distribution $q(x, y)$ (for example, $q(x, y) = p_T(x, y)$) is:

$$\ell(\theta, q(x, y)) = \mathbb{E}_{q(x)} \left[l[q(y|x), \hat{p}_\theta(y|x)] \right],$$

where l is a loss function (distance/divergence) between the true labeling distribution $q(y|x)$ and the predictive distribution $\hat{p}_\theta(y|x)$. This loss is not actually computed in practice for both the source and target data (we simply do not have labeled data for both of those), and it is used for theoretical discussion only. To make the theoretical discussion simpler, in this chapter, we will consider the simple l_1 distance between the two distributions. However, note that our theoretical discussion also holds for other divergences such as the Jensen-Shannon divergence (or symmetric KL). Both of these distances satisfy the triangle inequality, which we use in our derivation.

5.3.2 Test Time Domain Adaptation with Transformation Invariance

Recall that the task is to gradually update the network parameter θ so that the model can adapt to the target domain. In this setting, we do not have access to the source data nor a large amount of target data (only small batches), so traditional domain adaptation approaches such as representation alignment are often infeasible.

Given access only to the batches of unlabeled target data, it is natural to try to come up with a surrogate objective to optimize in lieu of the true loss function (which is

not available without the labels). In this sense, this has a close resemblance to the semi-supervised learning problem. In both settings, we aim to find a surrogate function to optimize for unlabeled data. In fact, both the entropy-minimization and pseudo-labeling techniques have been borrowed from the semi-supervised learning literature, with varying degrees of success. It should be noted that, in semi-supervised learning, the surrogate loss on the unlabeled data is minimized jointly with the supervised objective for labeled data, thus model collapsing is often not a problem. It is much more challenging, however, in this online adaptation setting, since the unsupervised surrogate loss is optimized alone, which can lead to the catastrophic collapse of the network parameter (such that it can not perform prediction anymore, even for the source domain). For example, as pointed out by its authors, TENT [Wang et al., 2021] tends to collapse to a trivial model if the batch size is small, which is often the case in practice.

In this work, we examine the invariance regularization technique in the setting of test time domain adaptation. Note that a similar regularizer [Miyato et al., 2018, Sajjadi et al., 2016, Berthelot et al., 2020] has been applied to semi-supervised learning. However, to the best of our knowledge, we are the first to investigate this type of regularization in the online domain adaptation problem. We aim to investigate both **theoretically** and **empirically** how this technique can help with adapting to the new target domain. We also find that this surrogate loss is much more robust (against small batch sizes) compared to TENT.

To summarize, we propose to enforce the network to be invariant under a predefined space of transformations on the target domain data. The idea is that this space of transformations contains the actual domain shifts, or at least could simulate them. Specifically, we minimize θ with the following objective:

$$\min_{\theta} \mathbb{E}_{p_T(x)} \left[\max_{x' \in A(x)} \mathrm{KL} \left[\hat{p}_{\theta}(y|x'), \hat{p}_{\overline{\theta}}(y|x) \right] \right],$$

where $\overline{\theta}$ is an identical copy of θ, and $A(x) = \{t(x)\}_{t \in S_t}$ (S_t is the predefined set of transformations).

We present the motivations and derivations of this objective function in the next subsection. The adapting and predicting procedure is depicted in Algorithm 1.

5.3.2.1 Invariance Regularization Technique

In this subsection, we introduce the invariance regularization technique for general transformations and provide theoretical results regarding the online domain adaptation

Algorithm 1 Adapt and predict. θ is the network parameter, $\{x_i\}_{i=1}^n$ is the batch data, α is the learning rate for online adaptation

Input θ, $\{x_1, x_2, ..., x_n\}$
Output θ_{new} (newly adapted), $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$

function ADAPT-PREDICT(θ, $\{x_1, x_2, ..., x_n\}$)
  $p_i \leftarrow f_\theta(x_i)$`.detach()` ▷ stop gradient
  $x_i' \leftarrow \arg\max_{x' \in A(x_i)} \text{KL}[p_\theta(y|x'), p_i]$
  $l(\theta) = \frac{1}{n} \sum_{i=1}^n \text{KL}[p_\theta(y|x_i'), p_i]$
  $\theta_{\text{new}} = \theta - \alpha \nabla_\theta l(\theta)$
  $\hat{y}_i = \arg\max f_{\theta_{\text{new}}}(x_i)$
end function

problem, which will later motivate our method.

Let's consider a set of transformations S_t, where each $t \in S_t$ is a transformation of the data, i.e., $t : \mathcal{X} \to \mathcal{X}$. Ideally, S_t should contain the transformations responsible for the domain shift. This set can contain transformations such as data augmentations, GAN-like cross-domain transformations, random noise perturbation, etc. For now, we shall not care much about the type of transformations in S_t. The only important detail is that these transformations should preserve the class information (this is formalized in Assumption 5.1). Let $A(x) = \{t(x)\}_{t \in S_t}$ and let $p_t(t)$ be a distribution over S_t that assigns probability mass (or density) to each transformation $t \in S_t$. Each distribution $p_t(t)$ defines a conditional distribution $T(x^t|x)$ (with the support set $A(x)$), where $x^t = t(x), t \sim p_t(t)$. This in turn defines a joint distribution over x, x^t and y as:

$$p_T(x, x^t, y) = p_T(x, y)T(x^t|x).$$

It is also worth noting that we have a transformed data distribution $p_T(x^t, y)$ of x^t and y (just marginalizing out x from the above).

Assumption 5.1.

$$I_T(x, y) = I_T(x^t, y),$$

where I_T is the mutual information, calculated on the target domain (please refer to Appendix D.1.1 for their exact formula).

Intuitively, this assumption means that the transformed image x^t contains the same amount of information about the label y as the original image x does. This assumption is equivalent to $\mathbb{E}_{p_T(x,x^t)}[\text{KL}[p_T(y|x), p_T(y|x^t)]] = 0$ (a derivation can be found in Appendix D.1.1), meaning that $p_T(y|x) = p_T(y|x^t)$ almost surely.

The transformation invariance regularizer on the target domain can be defined simply as:

$$h(\theta, x) = \max_{x' \in A(x)} D\left[\hat{p}_{\overline{\theta}}(y|x), \hat{p}_{\theta}(y|x')\right], \tag{5.1}$$

where $D[\cdot, \cdot]$ is a divergence/distance of choice and $\overline{\theta}$ is an identical copy of θ (so that the gradient is not back-propagated through it). We will see later (through an analysis of the gradient) that this is extremely important. Intuitively, it also helps to alleviate the network collapse problem, since, without this gradient blocking, the network would easily collapse to a constant output to minimize the regularizer term.

Intuitively, we can view this objective as using $\hat{p}_{\overline{\theta}}(y|x)$ as the target to optimize $\hat{p}_{\theta}(y|t(x))$ for all transformation t. This makes sense because, as discussed earlier, t does not change the semantic meaning of x significantly, so the predictive distribution of x and $t(x)$ should be similar.

Based on this regularizer, we can bound the loss on the target domain as follows:

Proposition 5.1. *Under Assumption 5.1, if D in Eq. 5.1 is either the forward or reverse KL divergence, we have:*

$$\ell(\theta, p_T(x, y)) \leq \mathbb{E}_{p_T(x)}\left[\sqrt{2h(\theta, x)}\right] + \ell(\theta, p_T(x^t, y)).$$

Proof. Provided in Appendix D.1.2. □

This means that the loss on the target distribution is bounded by the invariance regularizer and the loss on the transformed distribution $p_T(x^t, y)$.

Now, assume for a moment (even if it is not realistic) that the set S_t contains all the transformations that can successfully transform the target data distribution $p_T(x, y)$ back to the source distribution $p_S(x, y)$ (i.e., there exists a distribution $p_t(t)$ over S_t such that the resulting transformed distribution $p_T(x^t, y)$ is the same with $p_S(x, y)$). Then, we have that the loss on the target domain is bounded by the loss on the source domain and our invariance regularizer. Assume also that while adapting to the new target domain, our model does not suffer from catastrophic forgetting of the source domain (which is often the case if the network does not collapse since we are adapting within the same task). Then we have that the loss on the source domain would be small, and the invariance regularizer would also be small (since we explicitly minimize it). As a consequence, the loss on the target domain would be small and we can guarantee a good adaptation performance.

However, knowing the exact transformations from the target domain back to the source domains is often infeasible (only possible if we have some prior domain knowledge). This motivates us to find other transformations that are task-agnostic and can simulate the distribution shift between the target and source domains.

5.3.2.2 Practical choice of the transformation set

We want to choose task-agnostic transformations such that they do not alter the semantic meaning of the image significantly, while they can simulate the distribution shift. The first idea that comes to mind is to use common image augmentations (e.g., random crop, random vertical flip, etc.) However, our early experiments suggest that this method is not consistent across corruption types/domain shifts. This makes sense since image augmentations are often heuristic, and they may be able to simulate certain types of distribution shifts, but not others. Therefore, we opt to use l_∞-norm based perturbations. Empirically, this leads to better performance and also is consistent across domains/tasks.

To summarize, we use the l_∞-norm based perturbations as the transformations of images. Empirically, we also found out that using the reverse KL leads to better performance (although almost all distances/divergences have positive results). We provide explanations/intuitions as to why we should prefer the reverse KL, and why adversarial examples can help with the distribution shift problem in the next subsection. Our final optimization problem is:

$$\min_{\theta} \mathbb{E}_{p_T(x)} \left[\max_{||x'-x||_\infty \leq \epsilon} \mathrm{KL}\left[\hat{p}_\theta(y|x') | \hat{p}_{\overline{\theta}}(y|x)\right] \right].$$

We name our method **TIPI** (**T**est t**I**me ada**P**tation with transformation **I**nvariance). To save computational cost, we use the single-step adversarial method FGSM [Goodfellow et al., 2015] to find the adversarial example x'. Note that one can also use more advanced methods such as PGD [Madry et al., 2018] to find the adversarial perturbation to get better results (but with the trade-off of speed). We set ϵ to 2/255 in all of our experiments.

5.3.2.3 Additional Discussions

Why preferring reverse KL over forward KL? Natural choices for D are the KL-divergences (forward or reverse KL), since they can be computed straightforwardly for the predictive distributions, and also are commonly used in practice for similar

problems (e.g., VAT [Miyato et al., 2018] uses the forward KL). Our experiments empirically suggest that using the reverse KL as the divergence D leads to better results compared to the forward KL. Here, we also give theoretical intuition as to why this is the case. First of all, let's consider the gradient of θ for both the forward and reverse KL:
For the forward KL:

$$\nabla_\theta \mathrm{KL}[\hat{p}_{\bar{\theta}}(y|x)|\hat{p}_\theta(y|x')] = \sum_{y\in\mathcal{Y}} -\frac{\hat{p}_{\bar{\theta}}(y|x)}{\hat{p}_\theta(y|x')}\nabla_\theta \hat{p}_\theta(y|x'). \tag{5.2}$$

Meanwhile, for the reverse KL:

$$\begin{aligned}&\nabla_\theta \mathrm{KL}[\hat{p}_\theta(y|x')|\hat{p}_{\bar{\theta}}(y|x)] \\ &\quad= \sum_{y\in\mathcal{Y}} \left(-\log \frac{\hat{p}_{\bar{\theta}}(y|x)}{\hat{p}_\theta(y|x')} + 1\right) \nabla_\theta \hat{p}_\theta(y|x').\end{aligned} \tag{5.3}$$

Compare Eq. 5.3 to Eq. 5.2, we can see that the coefficient of $\nabla_\theta \hat{p}_\theta(y|x')$ in the reverse KL gets squashed down by the log function, leading to much smaller variance. We conjecture that the small variance in gradient is the reason why reverse KL performs better than forward KL.

How can invariance enforcing with adversarial examples help with the distribution shift problem? In our previous subsection, we provide theoretical evidence that motivates us to find transformations that can simulate common distribution shifts in practice. We argue here that l_∞ adversarial perturbation is a good candidate for this. First of all, l_∞ adversarial perturbation is a task-agnostic transformation that does not require much tuning, and we do not need to design heuristic transformations per domain. Secondly, a lot of the common domain shifts in practice are image corruptions that add noises to the images (Gaussian noise, Gaussian blur, JPEG compression noise, etc.), which can be simulated very well by adversarial noises. Lastly, even for more global domain shifts as seen in the DIGITS and VisDA17 experiments (Section 5.4), we conjecture that training a network that is robust against local and small changes (adversarial noise) would also help to make it more robust against larger shifts. Our empirical experiments also agree with this, as TIPI performs consistently across multiple types of domain shifts.

Implementation Details: Following TENT [Wang et al., 2021] and other test-time surrogate optimization methods, we only optimize the affine parameters of the model (i.e., the weight and bias parameters of the norm layers: BatchNorm, LayerNorm, InstanceNorm, or GroupNorm). This has been shown to lead to a more stable optimization, and also reduces the computational cost, as opposed to optimizing all parameters of the network. However, note that optimizing over the entire representation network (similar to SHOT [Liang et al., 2020]) is also possible. Furthermore, in case the network uses BatchNorm layers, we replace each BatchNorm with two BatchNorm layers (following Xie et al. [2020]), one for the clean images and one for the adversarial ones. This is because the batch statistics of the adversarial examples are much different from that of the clean images [Xie et al., 2020], thus mixing them might cause performance degradation. Note that these two BatchNorm layers only keep separate batch statistics (feature mean and variance), while sharing the affine parameters (weight and bias) – the parameters being optimized.

5.4 Experiments

We test the effectiveness of our method on a wide variety of tasks (robustness against image corruptions and domain adaptation). Especially, we performed extensive experiments on various batch sizes of the test-time data batches (as small as 2 and as large as 200). The highlight of our empirical findings is that: **TIPI outperforms other test-time adaptation baselines in all scenarios and problem settings.** We discuss these experiments in more detail in the following subsections. As a side note, we also found that using TIPI with TENT further improves the performance of both methods and helps TENT avoid network collapse. Details of our experiments (such as learning rate) can be found in our publicly available source code.

5.4.1 Evaluation Protocol and Baselines:

Evaluation Protocol: We consider a setting where the model is trained normally on the source domain. When the models are available online (e.g., ImageNet-trained Resnet50), we download the weights and keep them as is. Otherwise, we train the models ourselves. Following TENT [Wang et al., 2021], we perform online adaptation and report the average performance across all batches (while the model is being adapted). This mimics a real-world scenario.

Baselines: The main purpose of our experiments is to compare our invariance regularizer against predictive entropy as an unsupervised surrogate subjective. Therefore, **TENT** [Wang et al., 2021] is our main baseline in all of the experiments. We also consider recent and relevant fully test-time adaptation methods as our baselines, such as BatchNorm with test-time statistics (**BN**) [Schneider et al., 2020, Nado et al., 2020] and **T3A** [Iwasawa and Matsuo, 2021]. We also report the performance of the source-only model (**source** – no adaptation). Note that we also have tested T3A with test-time batch norm statistics (**T3A+BN**), but this model's performance is almost identical to (or even a little worse than) that of **BN** for all batch sizes, so we do not include it here. For a fair comparison, we rerun all methods in our experiment (to eliminate artifacts such as source model training), using their official source code. However, our numbers for these methods are very close to the reported numbers.

TTT [Sun et al., 2020] and TTT++ [Liu et al., 2021b] are not fully test-time adaptation methods since they require altering the training process of the source domain (to incorporate a self-supervised task). Thus, they are not optimal for off-the-shelf models that have not been trained with the self-supervised task (e.g., a ResNet50 trained on ImageNet downloaded directly from `torchvision`). For this reason, we do not include these two methods in the comparisons. Note that as reported in Table 2 of the TTT++ paper [Liu et al., 2021b], the variant of their model which has not been trained with the self-supervised task (which is the same as our evaluation protocol and most of the literature) largely underperforms our method.

EATA [Niu et al., 2022] proposes a data points selection strategy for TENT, and achieves state-of-the-art performance on the test time adaptation task. As discussed before, we believe this line of research is complementary to ours. Indeed, we are able to incorporate **TIPI into EATA, thereby improving its robustness** (EATA only uses TENT and is not robust against small batch sizes). We present this comparison in Appendix D.2.1.

5.4.2 Robustness to Image Corruptions

5.4.2.1 Experimental Settings

Following the literature, we conduct experiments on widely-used image-corruption datasets, namely:

- **CIFAR10-C** and **CIFAR100-C** [Hendrycks and Dietterich, 2019]: Common image corruptions are applied to the CIFAR10 and CIFAR100 datasets [Krizhevsky

Table 5.1: **CIFAR10-C**: Results are averaged over 15 types of corruption with the highest level of severity

Method	Accuracy						
Source	56.5						
	Test Time Adaptation Batch Size						
	2	5	10	20	50	200	Average
T3A	57.8	57.8	57.8	57.8	57.9	57.9	57.8
BN	61.8	70.8	75.1	77.5	78.9	79.6	74.0
TENT	17.4	60.7	74.0	78.6	81.0	81.4	65.5
TIPI (ours)	**78.6**	**79.8**	**80.3**	**81.0**	**81.2**	**81.5**	**80.4**

Table 5.2: **CIFAR100-C**: Results are averaged over 15 types of corruption with the highest level of severity

Method	Accuracy						
Source	37.5						
	Test Time Adaptation Batch Size						
	2	5	10	20	50	200	Average
T3A	39.0	39.0	39.0	39.0	39.0	39.0	39.0
BN	26.0	48.0	54.1	57.4	59.5	60.7	50.9
TENT	1.5	7.4	39.8	59.9	64.4	65.7	39.8
TIPI (ours)	**56.3**	**62.3**	**64.1**	**65.0**	**65.5**	**65.8**	**63.2**

et al., 2009]. We report the results for the highest level of severity, averaging over 15 types of corruption.

- **ImageNet-C**: Similar with the above two datasets, but for ImageNet [Russakovsky et al., 2015b]. We report the performance of all methods averaged over 15 types of corruption and 5 levels of severity.

For the CIFAR10-C and CIFAR100-C datasets, we use a WideResnet28-10 [Zagoruyko and Komodakis, 2016] model, which is the base model in Robustbench [Croce et al., 2021] and also used in TENT's official GitHub repository[1]. Note that in the original paper, TENT conducts experiments with a Resnet26 model [He et al., 2016]; however, the model/code was not publicly released, and there have been numerous reproducibility

[1]https://github.com/DequanWang/tent

Table 5.3: **ImageNet-C**: Results are averaged over 15 types of corruption and 5 levels of severity

Method	Accuracy						
Source	39.5						
	Test Time Adaptation Batch Size						
	2	4	8	16	32	64	Average
T3A	30.2	30.2	30.2	30.3	30.3	30.3	30.3
BN	13.9	33.4	42.8	46.9	48.9	49.9	47.2
TENT	0.8	13.1	44.3	53.7	56.8	57.3	45.2
TIPI (ours)	**45.2**	**53.3**	**56.7**	**57.9**	**58.3**	**58.4**	**55.0**

issues as reported in their GitHub page[2]. We use the Adam optimizer [Kingma and Ba, 2015] for these two datasets.

For the ImageNet-C dataset, we follow previous works and use a Resnet50 network. We use SGD to optimize our invariance regularizer.

5.4.2.2 Results

Table 5.1, 5.2 and 5.3 show that our method is the best performer among the baselines. In the CIFAR10-C and CIFAR100-C experiments, our method performs similarly with TENT for large batch sizes (200). However, as the batch size decreases, TENT collapses and TIPI outperforms it significantly. For the challenging ImageNet-C dataset, our method performs better than TENT even for large batch sizes. TENT still collapses with small batch sizes while TIPI is very robust to this parameter. T3A is not an optimization-based adaptation approach so its performance is robust against the batch size, but it largely underperforms TIPI in all settings. Even more concerning, T3A performs extremely purely on ImageNet-C (worse than the source model) – we conjecture this is because the non-parametric classifier in T3A is not effective when there is a large number of classes (1000) and each class's support set has a few data points (50).

Table 5.4: **DIGITS**: Online Domain Adaptation from SVHN to MNIST, MNIST-M and USPS

Method	SVHN→MNIST		SVHN→MNIST-M		SVHN→USPS	
	B.Size 4	B.Size 128	B.Size 4	B.Size 128	B.Size 4	B.Size 128
Source	85.8		60.3		84.9	
T3A	**86.3**	86.3	60.7	60.7	84.9	84.9
BN	75.6	83.5	56.0	60.9	72.4	80.8
TENT	27.0	90.7	19.2	61.4	38.7	90.4
TIPI (ours)	**86.3**	**91.6**	**64.2**	**63.8**	**86.8**	**93.9**

Table 5.5: **VisDA17**: Online Domain Adaptation of the VisDA classification challenge

Method	Accuracy							
Source	46.4							
	Test Time Adaptation Batch Size							
	2	4	8	16	32	64	128	Average
T3A	48.8	48.8	48.9	48.9	48.9	48.9	48.9	48.9
BN	38.3	49.5	54.6	57.3	58.7	59.4	59.8	53.9
TENT	12.3	26.5	53.0	59.9	62.1	62.0	62.0	48.3
TENTx2	10.0	22.1	45.9	59.2	62.5	63.1	63.2	46.6
TIPI (ours)	**51.2**	**60.3**	**62.7**	**64.0**	**64.3**	**64.6**	**64.5**	**61.7**

5.4.3 Domain Adaptation

5.4.3.1 Experimental Settings

In this problem, we consider two main adaptation tasks, with the following two datasets:

- **DIGITS**: the task is digit classification (10 classes), and we have to adapt a model from SVHN [Netzer et al., 2011] to MNIST [LeCun et al., 2010], MNIST-M [Ganin et al., 2016], and USPS [Hull, 1994]. For the MNIST and MNIST-M datasets, we only use the test set (10000 images) for the online adaptation. For the USPS dataset, we use both the training and test set to have roughly the same number of images for adaptation (9298 in total).
- **VisDA17** (12-class classification): is a more challenging dataset, with the adaption task from synthetic to real images. The source dataset contains 152397

[2]https://github.com/DequanWang/tent/issues

3D-rendered synthetic images, while the target dataset contains 55388 real images.

Following the experiments performed in TENT, we use a ResNet26 for DIGITS and a ResNet50 for the VisDA17 dataset. For the test-time optimization of our surrogate loss, we use Adam for DIGITS and SGD for VisDA17. Other details such as the learning rate can be found in the provided source code.

5.4.3.2 Results

Table 5.4 and Table 5.5 show a similar trend as observed in the image corruption experiments. In particular, TIPI performs the best among all the baselines. Especially, our method does not collapse with small batch sizes as most other models (such as BN or TENT) do. It is clear that TIPI achieves preferable performance in all settings while being much more robust compared to TENT.

5.4.4 Runtime and memory consumption

We would also like to discuss the runtime and memory (VRAM) consumption of our method. As TIPI is also an optimization-based adaptation, it is natural to compare it against TENT. Comparing our implementation of TIPI with TENT's official implementation, we observe that TIPI and TENT require roughly the same amount of VRAM, while our method is about two times slower than TENT (because it needs to find the adversarial examples). We believe that this is worth the trade-off, since our method clearly outperforms TENT, and is much more robust against small batch sizes – which is often the case in practice. To be more specific, using an NVIDIA Quadro RTX 6000, TIPI can reach 160 FPS (including both adaptation and prediction) with the standard 224x224 images of ImageNet and VisDA17.

However, just to be completely fair, we also compare TIPI with a variant of TENT where we perform two adaptation steps for each batch (thus its runtime is roughly the same as TIPI's), namely TENTx2 (second-last row in Table 5.5). We perform this comparison on the challenging adaptation dataset VisDA17 (for smaller datasets such as CIFAR and DIGITS, runtime and memory consumption are rarely a problem). TENTx2 does perform better than TENT (in the large-batch-size regime), but is still largely inferior to TIPI. Additionally, the collapsing problem for small batch sizes persists in this variant.

5.5 Conclusion

In this work, we propose to use transformation invariance as an unsupervised surrogate loss function for online domain adaptation. Similar to TENT, our method (TIPI) is task-agnostic and does not require altering the source training process of the model, thus can be applied easily to a wide variety of off-the-shelf models. Remarkably, compared to TENT, our method is much more robust against small batch sizes, which proves to be extremely useful in challenging real-world scenarios. A promising future research direction is to investigate a data point selection strategy for optimizing the TIPI objective, to further improve the performance of the method.

Chapter 6

uCAP: An Unsupervised Prompting Method for Vision-Language Models

Abstract

This chapter addresses a significant limitation that prevents Contrastive Language-Image Pre-trained Models (CLIP) from achieving optimal performance on downstream image classification tasks. The key problem with CLIP-style zero-shot classification is that it requires domain-specific context in the form of prompts to better align the class descriptions to the downstream data distribution. In particular, prompts for vision-language models are domain-level texts (e.g., "a centered satellite image of ...") which, together with the class names, are fed into the text encoder to provide more context for the downstream dataset. These prompts are typically manually tuned, which is time-consuming and often sub-optimal. To overcome this bottleneck, this work proposes uCAP, a method to automatically learn domain-specific prompts/contexts using only unlabeled in-domain images. We achieve this by modeling the generation of images given the class names and a domain-specific prompt with an unsupervised likelihood distribution, and then performing inference of the prompts. We validate the proposed method across various models and datasets, showing that uCAP consistently outperforms manually tuned prompts and related baselines on the evaluated datasets: ImageNet, CIFAR-10, CIFAR-100, OxfordPets (up to 2%), SUN397 (up to 5%), and Caltech101 (up to 3%).

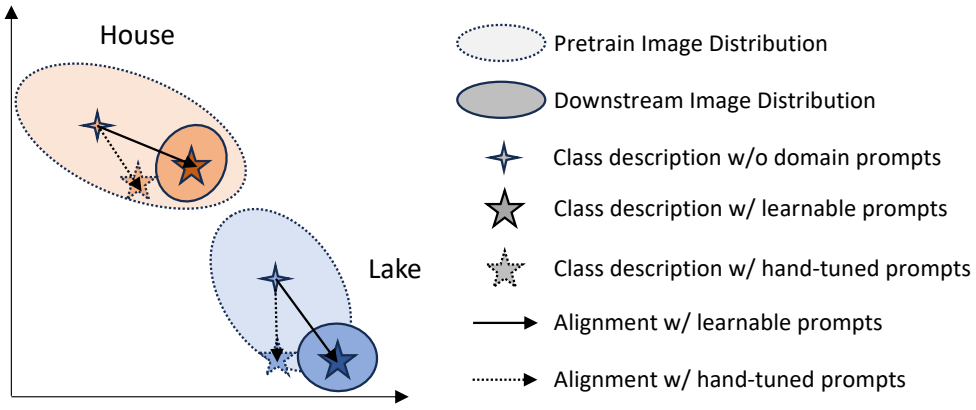


Figure 6.1: **Class Description Alignment:** The downstream image distribution might be shifted from the pre-training distribution (e.g., "satellite images of houses" is a sub-domain of "house"), thus the class description without any domain-specific context might be misaligned. This necessitates alignment through the use of domain-specific contexts. However, alignment with hand-tuned prompts (dotted line) is suboptimal. Our method, uCAP, automatically learns prompts in an unsupervised manner, leading to better alignment (solid line).

6.1 Introduction

Contrastive Language-Image Pre-trained Models are trained to align images and natural language representations. Models trained for representation alignment across modalities (e.g., CLIP [Radford et al., 2021], CLIPA [Li et al., 2023b], CoCa [Yu et al., 2022]) excel at tasks such as zero-shot classification and information retrieval. In zero-shot classification, for example, one first collects a class description for each class, and then, for each image at test time, the predicted class is the nearest class description in the representation space. Typically, to retrieve these class descriptions with high fidelity, practitioners would often need to combine the class name with a hand-designed text prompt template. These prompt templates contain the domain-specific context, which helps to better align the class names and the image distribution (Fig. 6.1). For example, for a satellite image dataset, existing zero-shot classification methods use the prompt "a centered satellite image of {}", whereas, for ImageNet, they use prompts such as "a bad photo of {}, a graffiti of {}". We argue that these prompts are domain-specific information that is obtained by either prior domain knowledge or by inspecting the image dataset.

However, there are several problems with this approach of hand-designing domain-specific prompt templates: they are **ad-hoc and potentially dataset-specific**; it is **time- and labor-intensive** to tune for each dataset; and the process might **not**

be truly zero-shot (as humans may use a small labeled validation set to tune the prompts).

Learning prompts automatically without human intervention is crucial, and we have seen some early attempts at this [Zhou et al., 2022b, Chen et al., 2023, Zhou et al., 2022a, Lu et al., 2022]. However, these methods require labeled images, which goes against the spirit of zero-shot learning. Our method, on the other hand, offers a true zero-shot approach that does not rely on labeled data. While our method can be used in conjunction with few-shot prompt learning methods for optimal performance, we choose to isolate the two settings in this work to evaluate our method's effectiveness independently.

We propose a method to learn the domain-specific prompts/contexts in an unsupervised manner using unlabeled target images. Here, we treat each downstream task as a domain; and introduce a graphical model of the data generation process, which assumes that a latent domain-specific context and the given class names generated the image dataset. To define the likelihood of an image being generated by this process, we leverage the distance (which is often the cosine distance in practice) learned by these pre-trained vision-language models. Specifically, we use the above distance between an image and a class description to form a scalar energy function, which subsequently defines an energy-based likelihood distribution. The intuition behind this is as follows: if an image was generated by this generative process, it should be close to at least one class description in the representation space. Finally, we perform posterior inference for the prompts, which are then used for image classification. We name our method **uCAP** (**u**nsupervised **C**ontrastive **A**uto **P**rompting).

Main Contributions and Insights:

- We propose a novel unsupervised prompt learning method for CLIP-styled zero-shot classification. Our method is motivated by a natural and intuitive data generation process, in which a domain-specific text prompt, together with the class names, generates the downstream image data. This formulation allows us to perform unsupervised learning of the prompts, unlike most existing prompt learning works [Zhou et al., 2022b,a, Chen et al., 2023, Lu et al., 2022] that require labels for optimization. To the best of our knowledge, uCAP is the first unsupervised prompt learning method that explicitly leverages the learned inductive bias of vision-language models (in the form of our energy-based likelihood).

- Through extensive evaluations with a wide range of models, datasets, and settings, we find that our method consistently outperforms prompt hand-tuning and other unsupervised learning methods. Thus, uCAP is a robust method that requires little hyperparameter tuning and generalizes well across a large number of settings.
- The prompts learned with our method exhibit strong transferability across settings. In particular, the prompts can be transferred across different distributions of the same task, and even to completely new tasks.

6.2 Related Work

Contrastive Language-Image Pre-training: The first and perhaps most influential work for cross-modality contrastive pre-training is CLIP [Radford et al., 2021]. CLIP jointly embeds text and images, which subsequently enables a wide range of tasks such as text-to-image generation [Rombach et al., 2022], image captioning [Mokady et al., 2021], and general visual question answering [Li et al., 2023a]. Since then, there have been several follow-up works that either expand to other modalities [Girdhar et al., 2023] or scale up model and training data [Li et al., 2023b, Cherti et al., 2023]. Our unsupervised prompt learning method can be applied straightforwardly to all of these multi-modal contrastive pre-trained models [Girdhar et al., 2023, Li et al., 2023b, Cherti et al., 2023].

Prompting Contrastive Language-Image Pre-trained Models: Prompt learning has gained popularity alongside the rise of large and/or multimodal language models, as it offers an effective and efficient way to alter the behavior of such models. In the context of vision-language models, prompt learning helps to better align the representations of class descriptions and images, leading to better predictive performance. There are some early attempts to learn the prompt for CLIP [Zhou et al., 2022b, Chen et al., 2023, Zhou et al., 2022a, Lu et al., 2022, Lee et al., 2023] using labeled images of a downstream task. The drawback of these methods is that they need to collect some labeled data for training, and thus are not truly zero-shot. Their results are thus not directly comparable to ours. However, we will present results comparing our method against those that utilize few-shot transfer learning (that does not use labels from the target set). There also exist methods [Pratt et al., 2022, Menon and Vondrick, 2023] that use large language models to generate the prompts for CLIP's zero-shot

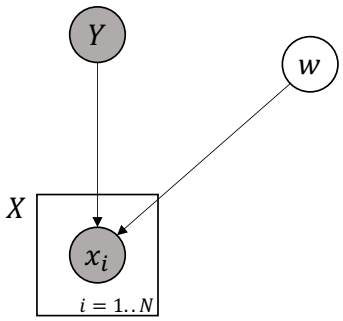


Figure 6.2: **Graphical Model:** We assume a generative model $p(x|Y, w)$ that goes from language to images as follows: The class labels Y together with the domain-specific context w generate the image representations. For example, if the class labels are "animal" and "plant", two different domain contexts "in water" and "on land" will lead to two completely different datasets.

classification. However, these methods rely on much larger generative language models to generate the class descriptions and still require partial human tuning. Additionally, our method can be used on top of these class descriptions to further improve the performance (as illustrated in Appendix E.2).

Unsupervised domain adaptation and its application to vision-language models: Our problem setting can also be viewed as unsupervised domain adaptation since we adapt the pre-trained vision-language models during test time to the downstream classification tasks. Existing test time adaptation methods such as Pseudo Labels, Entropy Minimization [Wang et al., 2021], and Invariance Enforcing [Nguyen et al., 2023] typically fine-tune the backbone network (which would be the vision encoder in our case) with a surrogate loss. However, modifying the vision encoder would break the alignment learned by the two encoders, degrading the model's performance. The astute reader might wonder if such unsupervised surrogate objectives can be used to learn the prompts. In fact, [Huang et al., 2022] has tried this idea (with Pseudo Labels) for prompting Vision-Language models. To address any concerns here, we present results that show that our method is superior to these unsupervised training objectives in Subsection 6.4.2.

6.3 Approach: uCAP

Assume that the set of class names $Y = \{y_1, y_2, ..., y_K\}$ together with the domain-specific context w generated our dataset $X = \{x_1, x_2, ..., x_N\}$ (N image data points)

according to the generative model in Figure 6.2. In particular, w is a text prompt that shows the domain context for the dataset generation. For example, if the class names are "plant" and "animal", two prompts "in water" or "on land" will yield completely different image datasets. Note that our formulation only requires knowing the set of class names $Y = \{y_1, y_2, ..., y_K\}$, not the specific class of each image (thus our method is unsupervised). We use x_i as the image representation, which is the embedding from the pre-trained vision encoder for each image, for simplicity and efficiency (e.g., we only have to collect and store the image representations for the optimization). For exposition (noting that uCAP is unsupervised), assume that the class names Y and the dataset X are given, and that our goal is to infer the domain-specific context w. Here, and throughout this chapter, we use "domain" to mean a downstream task, e.g., ImageNet classification.

Note that throughout this section, we refer to the data points as images. However, our method can be easily extended to other modalities, such as video and audio. In Appendix E.3, we present an experiment with a video dataset (UCF101 [Soomro et al., 2012]) to further support this.

To define the joint distribution for the inference of w, we need to define the prior of w and the likelihood of the data given the domain-specific context w. Following standard practice [Zhou et al., 2022b, Chen et al., 2023] in the prompt learning literature, we perform "soft prompt" learning, meaning that we learn the prompt in the continuous token embedding space.

Prior (for a prompt of length L):

$$p(w) = \prod_{l=1}^{L} p(w_l) = \prod_{l=1}^{L} \mathcal{N}(w_l; \mu, \Sigma), \tag{6.1}$$

where μ is the mean and Σ is the covariance matrix. These quantities can be hard-coded (e.g., $\mu = 0, \Sigma = \alpha I$), or can be computed from the empirical token embeddings of the model. In practice, we set μ and Σ to be the mean and covariance of the token embeddings across the vocabulary. This prior ensures that the soft prompt stays within the range of the natural token embeddings during training. We model the prior as a Gaussian distribution because of its simplicity, and that it works well in all of our experiments.

Likelihood: We use the inductive bias (i.e., the representation alignment between texts and images of similar concepts) learned by the fixed pre-trained model to define the likelihood, in the form of an energy-based distribution. Specifically:

$$p(X|Y,w) = \prod_{i=1}^{N} p(x_i|Y,w) = \prod_{i=1}^{N} \frac{e^{-E(x_i,Y,w)}}{Z_Y(w)},$$

where $Z_Y(w)$ is the normalizing constant:

$$Z_Y(w) = \int_{\mathbb{R}^k} e^{-E(x,Y,w)} d^k x,$$

where k is the dimension of x and $d^k x$ is the k-dimensional volume differential.

The scalar energy function is defined as:

$$E(x,Y,w) = \min_m D\left[x, f_\phi(w \oplus y_m)\right], \tag{6.2}$$

where f_ϕ is the pre-trained language encoder and D is the distance of choice (the CLIP models use the cosine distance). With some abuse of notation, $\oplus$ denotes the concatenation in the token embedding space (i.e., the class name y_m is first converted to the continuous token embedding, then concatenated to w).

Intuitively, this simple energy function means that, in order for an image to be created by Y and w, it should be close (in representation) to one domain-specific class description $w \oplus y_m$ of a class m. In other words, the domain-specific context w needs to bring the class descriptions closer to the image dataset.

Given this joint distribution, our goal is to infer the posterior $p(w|X,Y)$ and use it for image classification as:

$$\hat{p}(y^*|x^*) = \mathbb{E}_{p(w|X,Y)}\left[\hat{p}(y^*|x^*,w)\right], \tag{6.3}$$

where $\hat{p}(y^*|x^*,w)$ is the typical CLIP-styled predictive distribution given an image representation x^* and the prompt w.

Given the prior and the likelihood, there are several options for posterior inference: approximate inference (e.g., variational inference), exact sampling (e.g., stochastic gradient Langevin dynamics), or Maximum a Posteriori (MAP) point estimations then ensemble. In practice, we use the last option (MAP point estimations and then ensemble) as approximate Bayesian inference due to its simplicity and effectiveness (although this interpretation might be controversial and contested [Wilson and Izmailov, 2021]). Simply put, we use gradient descent to find several points w (from different

initializations) that maximize the MAP objective (Eq. 6.4) and ensemble them for the final prediction (Eq. 6.3).

Note that the above likelihood function is fully unsupervised. This allows us to do unsupervised inference of w. Specifically, the MAP objective is:

$$\begin{aligned} &\max_w \left[\frac{1}{N}\log p(w) + \frac{1}{N}\sum_{i=1}^{N}\log p(x_i|Y,w)\right] \\ =&\max_w \left[\frac{1}{N}\log p(w) - \frac{1}{N}\sum_{i=1}^{N} E(x_i,Y,w) - \log Z_Y(w)\right]. \end{aligned} \tag{6.4}$$

While $\log p(w)$ (Eq. 6.1) and $E(x_iY,w)$ (Eq. 6.2) are easy to compute, $\log Z_Y(w)$ is intractable. Fortunately, we can compute the gradient of $\log Z_Y(w)$ w.r.t. w as below (the detailed derivation is provided in Appendix E.1):

$$\nabla_w \log Z_Y(w) = -\mathbb{E}_{p(x|Y,w)}\left[\nabla_w E(x,Y,w)\right].$$

Therefore, our gradient estimator of Eq. 6.4 is:

$$\begin{aligned} &\frac{1}{N}\nabla_w \log p(w) - \frac{1}{N}\sum_{i=1}^{N}\nabla_w E(x_i,Y,w) + \mathbb{E}_{p(x|Y,w)}\left[\nabla_w E(x,Y,w)\right] \\ \approx&\frac{1}{N}\nabla_w \log p(w) - \frac{1}{N}\sum_{i=1}^{N}\nabla_w E(x_i,Y,w) + \frac{1}{M}\sum_{i=1}^{M}\nabla_w E(\tilde{x}_i,Y,w), \end{aligned}$$

where $\tilde{x}_i \sim p(x|Y,w), i \in \{1,2,...,M\}$.

Note that $\{\tilde{x}_i\}_{i=1}^M$ are the data points sampled from the current distribution $p(x|Y,w)$, not our real observed data points $\{x_i\}_{i=1}^N$. To sample from the distribution $p(x|Y,w)$ (knowing only its energy function), we use Stochastic Gradient Langevin Dynamics [Welling and Teh, 2011]:

$$\begin{aligned} &\tilde{x}^{(0)} \sim \pi(x), \\ &\tilde{x}^{(t+1)} = \tilde{x}^{(t)} - \epsilon\nabla_x E(x,Y,w) + \sqrt{2\epsilon}z^{(t)}, \end{aligned} \tag{6.5}$$

where $z^{(t)} \sim \mathcal{N}(0,I)$ and $\pi(x)$ is a sampling prior distribution (often set to standard Gaussian). We set $\epsilon = 1e-6$ and perform 200 (very light-weight) iterations of Eq. 6.5 for the sampling.

On the complexity of using an ensemble of prompts: As aforementioned, we find several MAP estimations of w and ensemble them for the final prediction (Eq. 6.3). Similarly, CLIP also ensembles a number of hand-tuned prompts. It should be noted that the inference cost (after the prompts are learned) with multiple prompts is almost unchanged, since the class embeddings are precomputed for each prompt, and we only have to run the vision encoder once for each image. Only the training cost will scale (linearly) with the number of prompts. This is often not a problem in practice.

Implementation Details: In practice, we replace the coefficient $\frac{1}{N}$ of $\nabla_w \log p(w)$ by a tunable hyperparameter α to make it independent of the datasets – this is also a standard practice in Bayesian inference. Additionally, we set $M = N$, similar to other works on energy-based models. Note also that we initialize the prompt w randomly for each MAP estimation. We do not use the dataset-specific hand-tuned prompts at any stage during optimization (some existing works use these prompts for either prompt initialization or pseudo label generation). We emphasize that we only perform *very light hyper-parameter tuning for our method* – this illustrates the robustness of our method and shows that it requires minimal human tuning in practice. In particular, we use the same sets of hyper-parameters for all datasets; and only use two configurations of hyper-parameters (one for all CLIP variants and one for OpenCLIP and CLIPA, due to the differences in model sizes and their training datasets):

- CLIP variants: Adam [Kingma and Ba, 2015] optimizer for 40 epochs, `lr=0.001`, α`=0.0001` (coefficient of the log prior), prompt w of length 10.
- OpenCLIP and CLIPA: Adam optimizer for 40 epochs, `lr=0.01`, α`=0.001` (coefficient of the log prior), prompt w of length 5.

6.4 Experiments

In this section, we extensively validate our method with different models, datasets, and settings. Our main results (Subsection 6.4.1 and 6.4.2) illustrate that our method is superior to prompt hand-tuning and other unsupervised methods, and the additional experiments in Subsection 6.4.3 show that our method is more favorable when compared to few-shot prompt learning methods, and that the learned prompts can be transferred in various settings.

6.4.1 Experiment Settings

In this subsection, we describe the setting of our main experiments, which include a wide range of models and datasets.

Models: We consider popular and state-of-the-art contrastive image-language pre-trained models, namely:

- All variants of CLIP [Radford et al., 2021]: RN50, RN101, ViT-B/32, ViT-B/16, ViT-L/14, and ViT-L/14@336px.
- The best performing variant of Open-CLIP [Cherti et al., 2023]: ViT-G/14.
- The best performing variant of CLIPAv2 [Li et al., 2023b], ViT-H/14, which is also state-of-the-art for zero-shot classification among the open-sourced models.

Datasets: We benchmark several popular image classification datasets, including ImageNet [Russakovsky et al., 2015a], CIFAR-10 [Krizhevsky et al., 2009], CIFAR-100 [Krizhevsky et al., 2009], OxfordPets [Parkhi et al., 2012], SUN397 [Xiao et al., 2010], and Caltech101 [Li et al., 2022]. Our evaluation setting is identical to that of CLIP [Radford et al., 2021].

Baselines:

- **Class names only**: The baseline where we only use the class names as the class descriptions, without any prompts.
- **Class names + Hand-tuned prompts (HTP)**: This baseline uses the hand-tuned prompts reported in the CLIP, CLIPA, and Open-CLIP papers. Note that the number of hand-tuned prompts varies among datasets (presumably due to lack of human labor to excessively tune all datasets): ImageNet has the most number of hand-tuned prompts (80). We re-evaluate all the models on the chosen datasets (our re-evaluated numbers are very close to the reported ones).
- **Prompt Learning with Pseudo Labels**: We also apply some of the most common unsupervised domain adaptation methods, namely Entropy Minimization and Pseudo Labels, for the prompt learning setting. However, Entropy Minimization often collapses in practice (this agrees with the observations in

Table 6.1: **Main Result**: Reported numbers are from 5 runs.

Model	Method	ImageNet	CIFAR-10	CIFAR-100	OxfordPets	SUN397	Caltech101
CLIP RN50	Classnames + HTP	59.6	71.5	41.9	85.8	59.5	83.1
	Classnames Only	54.8	66.3	36.2	79.4	53.1	75.4
	Pseudo Labels	57.6	67.6	36.3	80.4	55.9	77.1
	uCAP (1)	60.5	70.2	41.8	86.6	62.9	85.0
	uCAP (10)	**61.9**	**72.4**	**43.3**	**87.5**	**64.6**	**86.2**
CLIP RN101	Classnames + HTP	62.2	80.7	48.9	**86.8**	57.7	84.4
	Classnames Only	58.4	77.3	43.9	80.4	52.8	81.1
	Pseudo Labels	59.2	79.2	45.1	81.8	54.5	84.1
	uCAP (1)	62.2	82.6	50.3	86.0	62.7	88.0
	uCAP (10)	**63.7**	**83.5**	**51.7**	**86.8**	**64.0**	**88.5**
CLIP ViT-B/32	Classnames + HTP	63.2	89.9	65.1	**87.5**	62.1	87.1
	Classnames Only	58.5	85.9	60.3	80.0	58.5	84.8
	Pseudo Labels	61.1	87.5	61.4	81.0	62.1	86.1
	uCAP (1)	62.8	89.9	65.4	85.8	65.7	88.0
	uCAP (10)	**64.7**	**90.9**	**66.5**	87.2	**67.0**	**88.6**
CLIP ViT-B/16	Classnames + HTP	68.6	90.8	68.3	89.1	64.1	88.0
	Classnames Only	63.5	86.5	62.0	81.6	59.6	85.0
	Pseudo Labels	66.1	89.3	63.8	82.7	62.8	87.4
	uCAP (1)	68.1	91.6	68.1	88.5	67.9	89.3
	uCAP (10)	**69.7**	**91.9**	**69.2**	**89.9**	**69.8**	**89.9**
CLIP ViT-L/14	Classnames + HTP	75.3	95.6	78.2	93.6	67.3	90.6
	Classnames Only	71.0	89.4	72.7	87.7	62.6	86.2
	Pseudo Labels	73.3	92.7	75.2	89.1	65.5	88.0
	uCAP (1)	74.6	96.1	78.9	94.5	71.6	91.2
	uCAP(10)	**76.5**	**96.4**	**79.9**	**95.1**	**73.2**	**91.7**
CLIP ViT-L/14 @336px	Classnames + HTP	76.2	94.9	77.0	93.8	68.2	**91.3**
	Classnames Only	72.2	89.0	72.3	87.7	63.7	86.4
	Pseudo Labels	73.9	92.1	74.3	89.4	66.3	87.9
	uCAP (1)	75.6	95.7	77.5	94.8	72.1	91.0
	uCAP (10)	**77.2**	**96.0**	**78.7**	**95.5**	**73.8**	**91.3**
OpenCLIP ViT-G/14	Classnames + HTP	80.1	98.2	87.6	95.2	74.3	**92.8**
	Classnames Only	76.7	86.6	77.8	90.9	69.4	90.0
	Pseudo Labels	78.3	88.7	81.8	91.3	72.8	92.3
	uCAP (1)	79.3	98.6	87.5	94.9	76.4	92.6
	uCAP (10)	**80.3**	**98.7**	**88.1**	**95.3**	**77.5**	**92.8**
CLIPA ViT-H/14	Classnames + HTP	**81.0**	98.8	89.1	95.3	73.2	92.9
	Classnames Only	79.4	98.6	87.0	93.2	70.8	91.4
	Pseudo Labels	80.0	98.9	88.1	93.3	72.0	92.4
	uCAP (1)	80.3	99.0	89.5	95.1	75.0	92.5
	uCAP (10)	**81.0**	**99.1**	**90.0**	**95.6**	**75.3**	**93.0**

[Wang et al., 2021, Nguyen et al., 2023]). Therefore, we only report the results for the Pseudo Labels baseline. Note that [Huang et al., 2022] proposed the same unsupervised prompt learning method with Pseudo Labels. However, they only report results for CLIP-RN50, and some numbers are different from existing works (e.g., largely different than the numbers reported in CLIP [Radford et al., 2021]). Therefore, we re-evaluate the method for all the models and datasets considered in this chapter, with our standardized setting.

- **uCAP (ours)**: Our proposed method. We test the variants with 1 prompt and 10 prompts.

6.4.2 Main Results

Table 6.1 shows the results of uCAP as well as the baselines for the eight models and six datasets. Overall, using more prompts leads to better performance (similar to the hand-tuned prompts case), and our method outperforms other baselines in virtually all settings. This indicates that the hand-tuned prompts are often not optimal (while being time- and labor-intensive). For datasets where the hand-selected prompts are relatively under-tuned (e.g., SUN397), uCAP offers up to 5% improvement over the human-selected prompts. Note that Pseudo Labels (a classic unsupervised domain adaptation method) does offer some improvement over the Class Names Only baseline. However, using that method alone is not sufficient (compared to hand-tuned prompts and our method), since it does not directly leverage the inductive bias (representation alignment) captured by the encoders.

We also investigate the effect of the number of prompts on the final classification performance. Figure 6.3 illustrates this relationship for both natural (hand-tuned) prompts and the prompts learned by uCAP. In both cases, more prompts generally translate to better performance. Interestingly, the performance of our method scales much faster with the number of prompts. We recommend using 5-10 prompts in practice.

6.4.3 Additional Results and Discussion

Comparison with (few-shot) supervised prompt learning methods: There are also methods [Zhou et al., 2022b,a, Chen et al., 2023, Lu et al., 2022] that learn the prompts with labeled data points. Since these methods require labels for the prompt optimization, they need to do transfer learning from a labeled source dataset

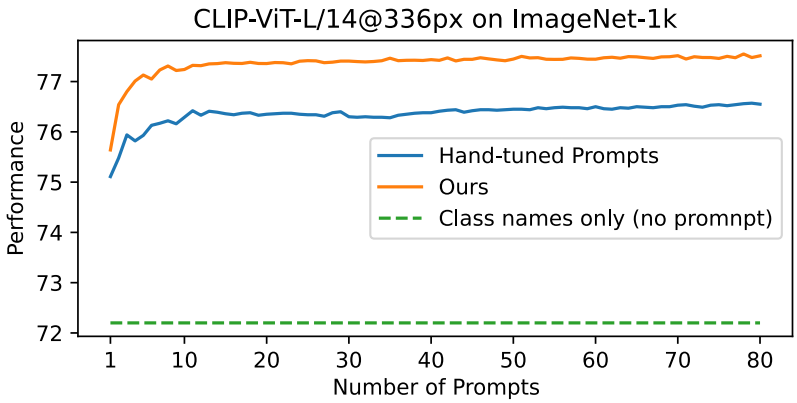


Figure 6.3: Effect of the number of prompts on zero-shot classification performance. For both hand-tuned and learned prompts, more prompts generally lead to better performance. Interestingly, our performance scales much faster with the number of prompts when compared to hand-tuned prompts.

when the target dataset is unlabeled. We compare uCAP to two such approaches, namely CoOp [Zhou et al., 2022b] and Plot [Chen et al., 2023], since they **1)** use the same domain-level (not instance-level) prompts as ours and **2)** conduct the transfer learning experiment. Specifically:

- CoOp [Zhou et al., 2022b], Plot [Chen et al., 2023]: perform transfer learning from a labeled source dataset to an unlabeled target dataset. In their experiment, CoOp and Plot perform transfer learning from ImageNet to ImageNetV2 [Recht et al., 2019], ImageNet-Sketch [Wang et al., 2019], ImageNet-R [Hendrycks et al., 2021a], ImageNet-A [Hendrycks et al., 2021b]. We use the reported performance from their paper. Note that Plot only reports results for CLIP-RN50, while CoOp reports results for CLIP-RN50, RN101, ViT-B/32, and ViT-B/16. Plot uses 4000 ImageNet labels in total, while CoOp uses 16000 labels.

- uCAP (ours): performs **unsupervised learning** on the unlabeled dataset directly. In this sense, our method tackles a harder problem than CoOp and Plot, since it does not have/need access to a labeled source dataset.

We focus on CLIP variants RN50, RN101, ViT-B/32, and ViT-B/16, as these are the models investigated in the CoOp paper. Our findings, presented in Table 6.2, show that our method uCAP surpasses CoOp by approximately 1.5% on average. Notably, uCAP outperforms CoOp and Plot in most datasets, including ImageNet-Sketch, ImageNet-A, and ImageNet-R, demonstrating its effectiveness and versatility. This is remarkable, given that uCAP does not rely on labeled source data from ImageNet, unlike CoOp

Table 6.2: **Comparison with CoOp**. CoOp performs transfer learning from labeled ImageNet to other target datasets, while our method performs unsupervised prompt learning directly on each dataset (thus does not require a labeled source dataset). Reported numbers are from 5 runs

Model	Method	ImageNetV2	ImageNet-Sketch	ImageNet-A	ImageNet-R	Average
CLIP RN50	Classnames + HTP	51.84	32.89	23.95	56.73	41.35
	Classnames Only	48.64	30.27	20.83	56.87	39.15
	Plot (4000 labels)	55.11	33.00	21.86	55.61	41.39
	CoOp (16000 labels)	**55.40**	34.67	23.06	56.60	42.43
	uCAP	54.70	**36.21**	**24.07**	**61.80**	**44.19**
CLIP RN101	Classnames + HTP	55.90	39.74	30.95	65.96	48.14
	Classnames Only	51.85	35.80	27.56	64.43	44.91
	CoOp (16000 labels)	**58.60**	40.40	29.60	64.98	48.39
	uCAP	57.47	**41.51**	**31.02**	**69.09**	**49.77**
CLIP ViT-B/32	Classnames + HTP	55.40	40.92	32.13	66.66	48.79
	Classnames Only	51.50	37.37	29.19	65.93	46.00
	CoOp (16000 labels)	**58.24**	41.48	31.34	65.78	49.21
	uCAP	57.74	**42.33**	**32.90**	**70.39**	**50.84**
CLIP ViT-B/16	Classnames + HTP	61.49	46.96	50.28	75.36	58.22
	Classnames Only	58.04	43.08	45.39	74.61	55.28
	CoOp (16000 labels)	**64.56**	47.89	49.93	75.14	59.38
	uCAP	63.38	**48.84**	**51.76**	**78.40**	**60.60**

and Plot. However, CoOp and Plot exhibit better performance in ImageNetV2, and we hypothesize that this is likely due to the relatively small distribution shift between ImageNet and ImageNetV2, which enables labeled data from ImageNet to provide substantial information for ImageNetV2.

Do the prompts have semantic meaning? A natural question is if the learned prompts have any meaning. Recall that we optimize in the token embedding space (soft prompt). Therefore, the best way we can check for semantic meaning is to map the soft prompts back to actual words by algorithms such as nearest neighbour. Similar to the observations from other recent soft prompt learning works [Zhou et al., 2022b], we also find that the prompts do not have any semantic meaning (nor are they even legible English).

Table 6.3: **Transfer learning**. Unsupervised prompt learning on ImageNet and inference on other target datasets. We use 10 prompts for uCAP. Reported numbers are from 5 runs.

		Same Task (shifted dist.)				Across Tasks				
Model	Method	ImageNetV2	ImageNet-Sketch	ImageNet-A	ImageNet-R	CIFAR10	CIFAR100	OxfordPets	SUN397	Caltech101
CLIP RN50	Classnames + HTP	51.8	32.9	**24.0**	56.7	71.5	41.9	85.8	59.5	83.1
	Classnames Only	48.6	30.3	20.8	56.9	66.3	36.2	79.4	53.1	75.4
	uCAP	**54.4**	**35.1**	**24.0**	**58.9**	**72.0**	**44.1**	**86.1**	**62.8**	**85.9**
CLIP RN101	Classnames + HTP	55.9	39.7	**31.0**	66.0	80.7	48.9	**86.8**	57.7	84.4
	Classnames Only	51.9	35.8	27.6	64.4	77.3	43.9	80.4	52.8	81.1
	uCAP	**56.9**	**40.9**	30.5	**66.7**	**82.8**	**51.7**	**86.8**	**63.6**	**88.5**
CLIP ViT-B/32	Classnames + HTP	55.4	40.9	32.1	66.7	89.9	65.1	**87.5**	62.1	87.1
	Classnames Only	51.5	37.4	29.2	65.9	85.9	60.3	80.0	58.5	84.8
	uCAP	**57.1**	**42.1**	**32.3**	**68.0**	**90.0**	**66.5**	87.3	**65.6**	**89.0**
CLIP ViT-B/16	Classnames + HTP	61.5	47.0	50.3	75.4	90.8	68.3	89.1	64.1	88.0
	Classnames Only	58.0	43.1	45.4	74.6	86.5	62.0	81.6	59.6	85.0
	uCAP	**62.7**	**48.7**	**50.8**	**77.4**	**91.1**	**69.7**	**89.8**	**68.0**	**90.9**
CLIP ViT-L/14 @336px	Classnames + HTP	70.3	60.3	77.5	88.7	94.9	77.0	**93.8**	68.2	**91.3**
	Classnames Only	65.7	56.6	71.0	86.0	89.0	72.3	87.7	63.7	86.4
	uCAP	**71.0**	**60.7**	**78.7**	**89.2**	**95.3**	**78.6**	92.6	**72.0**	91.0

Transferability of the prompts: Another natural question is whether the learned prompts can be transferred, i.e., learn the prompts in one setting and test them in another setting. We first re-emphasize that uCAP is easily applicable to virtually any inference setting (just use the unlabeled test data for prompt learning), and thus transfer learning is often not needed. However, it is still interesting to see how the unsupervised-learned prompts perform in a transfer learning setting, in case of a sudden change in the data distribution (e.g., the camera sensor gets replaced, continual task change, etc.). Note that this result should not be compared to the transfer learning setting of CoOp, since CoOp uses labeled data from the (ImageNet) source dataset. Our findings are as follows:

- Transferability across models: This is not feasible because each model has a different token embedding look-up table (the token embedding dimension might

even change across models).

- Transferability across different domains of the same task: For this experiment, we perform transfer learning from ImageNet to ImageNetV2, ImageNet-Sketch, ImageNet-A, and Imagenet-R. Note that we only use the unlabeled image from the source dataset (ImageNet). Table 6.3 (first half) shows the result of this experiment, indicating that the prompts can be transferred across different domains and withstand some degree of distribution shift.

- Transferability across tasks: This is not guaranteed, since the prompts are learned in a task-specific manner. However, in practice, we observe a strong capability of transferring the prompts across tasks, especially from a more complex task (ImageNet) to other tasks. We report the performance of prompts learned by ImageNet on other tasks (CIFAR-10, CIFAR-100, OxfordPets, SUN397, and Caltech101) in the second half of Table 6.3. Overall, the learned prompts offer a large performance gain compared to the Class Names Only baseline and are even better than the hand-tuned prompts in most cases.

6.5 Conclusion

To conclude, we propose uCAP, an unsupervised method to automatically learn the prompts for zero-shot prediction in vision-language models. In particular, we model a data generation process that includes a domain-specific context (which, together with the class names, generates the images), and use an unsupervised energy-based distribution as the likelihood. This allows us to infer back the domain contexts, and use them for the classification. Our method shows superior performance when compared to hand-tuned prompts and relevant baselines. Future research directions include: to generalize the energy-based formulation to other prompt learning settings and to consider the instance-level prompts for zero-shot classification (although our preliminary experiments suggest that the performance gain is not worth the added complexity of additional networks).

Chapter 7

Conclusions

7.1 Concluding Remarks

This thesis has delved into the topic of distributional robustness, exploring different settings and frameworks to tackle the distribution shift problem. These settings include multi-source domain generalization, single-source unsupervised domain adaptation, domain generalization with federated learning, test time adaptation, and domain adaptation for foundation vision-language models. We believe this thesis provides the tools to help **(1)** learn robust and generalizable machine learning models and **(2)** perform active adaptation and refinement of the models when needed, for a wide range of real-world scenarios and tasks.

This thesis also underscores the importance of considering robustness as a fundamental aspect of model development and deployment, particularly in scenarios where performance consistency and models' reliability are paramount. To speak more broadly, robustness, reliability, safety, and trustworthiness should be top priorities when developing and deploying machine learning models. In the era of foundation models, these aspects should be emphasized and warrant additional attention.

7.2 Future Vision

We have entered the era of foundation models, which are known to be more robust to distribution shifts. These models typically have billions of parameters (and increasing) and are pre-trained on an "astronomically" large amount of data. As a result, foundation models offer higher-quality and more robust representation and predictive distribution. However, I envision that distribution shift will remain a central topic of discussion to ensure safe and reliable deployments of machine learning models.

Specifically:

- First, although large foundation models are gaining popularity, smaller task-specific models offer unique benefits and are here to stay. In particular, they are more efficient, more accurate for some specific tasks, and easier to iterate through multiple rounds of training/adaptation. A major future research direction would be to leverage the robustness of foundation models and transfer it to smaller task-specific models.

- Secondly, foundation models also require additional safeguards for safe and reliable deployments. An emerging problem of foundation models is the alignment problem, in which model output is trained to align with human preferences and values. However, such human preferences and values can and often do change over time. This motivates effective and efficient model adaptation methods to keep the models aligned, safe, and trustworthy.

As we continue to navigate the evolving landscape of artificial intelligence and machine learning, I hope that this thesis can provide the tools and motivation to develop safe and reliable AI systems. By prioritizing robustness in model development and deployment, we can strive to create safer, more reliable, and more resilient AI systems that effectively serve the diverse needs of society.

References

David Acuna, Guojun Zhang, Marc T Law, and Sanja Fidler. f-Domain-Adversarial Learning: Theory and Algorithms. *ICML*, 2021.

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-Adversarial Neural Networks. *arXiv*, 2014.

Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. Adversarial Invariant Feature Learning with Accuracy Constraint for Domain Generalization. *ECML PKDD*, 2019.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep Variational Information Bottleneck. *ICLR*, 2017.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated Learning with Personalization Layers. *arXiv*, 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv*, 2019.

Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized Learning for Domain Adaptation under Label Shifts. *ICLR*, 2019.

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. MetaReg: Towards Domain Generalization using Meta-Regularization. *NeurIPS*, 2018.

Harkirat Behl, Atılım Güneş Baydin, and Philip H.S. Torr. Alpha maml: Adaptive model-agnostic meta-learning. *ICML Workshop on Automated Machine Learning*, 2019.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 2010.

James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *JMLR*, 2012.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. ReMixMatch: Semi-Supervised Learning with Distribution Matching and Augmentation Anchoring. *ICLR*, 2020.

Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from Several Related Classification Tasks to a New Unlabeled Sample. *NIPS*, 2011.

Vladimir I Bogachev. Measure Theory. *Springer Science & Business Media*, 2007.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised Pixel–Level Domain Adaptation with Generative Adversarial Networks. *CVPR*, 2017.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to Statistical Learning Theory. *Summer School on Machine Learning*, 2003.

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain Generalization by Solving Jigsaw Puzzles. *CVPR*, 2019.

Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to Balance Specificity and Invariance for In and Out of Domain Generalization. *ECCV*, 2020.

Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. PLOT: Prompt Learning with Optimal Transport for Vision-Language Models. *ICLR*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *ICML*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners. *NeurIPS*, 2020b.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. *CVPR*, 2023.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. *CVPR*, 2018.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse Image Synthesis for Multiple Domains. *CVPR*, 2020.

Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoff Gordon. Domain Adaptation with Conditional Distribution Matching and Generalized Label Shift. *NeurIPS*, 2020.

Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning Bounds for Importance Weighting. *NIPS*, 2010.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. *NeurIPS Datasets and Benchmarks Track*, 2021.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive Personalized Federated Learning. *arXiv*, 2020.

Zhengming Ding and Yun Fu. Deep Domain Generalization With Structured Low-Rank Constraint. *IEEE Transactions on Image Processing*, 2017.

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. *ICLR*, 2021.

Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees GM Snoek, and Ling Shao. Learning to Learn with Variational Information Bottleneck for Domain Generalization. *ECCV*, 2020.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized Federated Learning: A Meta-Learning Approach. *arXiv*, 2020.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-Aware Minimization for Efficiently Improving Generalization. *ICLR*, 2021a.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware Minimization for Efficiently Improving Generalization. *ICLR*, 2021b.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *JMLR*, 2016.

Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain Generalization for Object Recognition with Multi-task Autoencoders. *ICCV*, 2015.

Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. ImageBind: One Embedding Space To Bind Them All. *CVPR*, 2023.

Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv*, 2016.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *NIPS*, 2014.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *ICLR*, 2015.

Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and Zico Kolter. Test-Time Adaptation via Conjugate Pseudo-labels. *NeurIPS*, 2022.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *JMLR*, 2012.

Aditya Grover, Christopher Chute, Rui Shu, Zhangjie Cao, and Stefano Ermon. AlignFlow: Cycle Consistent Learning from Multiple Domains via Normalizing Flows. *AAAI*, 2020.

Ishaan Gulrajani and David Lopez-Paz. In Search of Lost Domain Generalization. *ICLR*, 2021.

Farzin Haddadpour and Mehrdad Mahdavi. On the Convergence of Local Descent Methods in Federated Learning. *arXiv*, 2019.

Filip Hanzely and Peter Richtárik. Federated Learning of a Mixture of Global and Local Models. *arXiv*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *ICLR*, 2019.

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*, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural Adversarial Examples. *CVPR*, 2021b.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. *ICML*, 2018.

Tony Huang, Jack Chu, and Fangyun Wei. Unsupervised Prompt Learning for Vision-Language Models. *arXiv*, 2022.

Jonathan J. Hull. A Database for Handwritten Text Recognition Research. *PAMI*, 1994.

Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. DIVA: Domain Invariant Variational Autoencoders. *MIDL*, 2020.

Yusuke Iwasawa and Yutaka Matsuo. Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization. *NeurIPS*, 2021.

Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving Federated Learning Personalization via Model Agnostic Meta Learning. *arXiv*, 2019.

Fredrik D Johansson, David Sontag, and Rajesh Ranganath. Support and Invertibility in Domain-Invariant Representations. *AISTATS*, 2019.

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive Adaptation Network for Unsupervised Domain Adaptation. *CVPR*, 2019.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter Theory for Local SGD on Identical and Heterogeneous Data. *AISTATS*, 2020.

Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A Efros, and Antonio Torralba. Undoing the Damage of Dataset Bias. *ECCV*, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *ICLR*, 2014.

Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On Convergence and Stability of GANs. *arXiv*, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features from Tiny Images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST Database of handwritten digits. 2010.

Dongjun Lee, Seokwon Song, Jihee Suh, Joonmyeong Choi, Sanghyeok Lee, and Hyunwoo J Kim. Read-only Prompt Optimization for Vision-Language Few-shot Learning. *ICCV*, 2023.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Deeper, Broader and Artier Domain Generalization. *ICCV*, 2017.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to Generalize: Meta-Learning for Domain Generalization. *AAAI*, 2018a.

Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101. 2022. doi: 10.22002/D1.20086.

Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain Generalization With Adversarial Feature Learning. *CVPR*, 2018b.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. *ICML*, 2023a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization for Heterogeneous Networks. *MLSys*, 2020a.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the Convergence of FedAvg on Non-IID Data. *ICLR*, 2020b.

Xianhang Li, Zeyu Wang, and Cihang Xie. CLIPA-v2: Scaling CLIP Training with 81.1% Zero-shot ImageNet Accuracy within a $10, 000 Budget; An Extra $4, 000 Unlocks 81.8% Accuracy. *arXiv*, 2023b.

Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain Generalization via Conditional Invariant Representations. *AAAI*, 2018c.

Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep Domain Generalization via Conditional Invariant Adversarial Networks. *ECCV*, 2018d.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation. *ICML*, 2020.

Ming-Yu Liu and Oncel Tuzel. Coupled Generative Adversarial Networks. *NIPS*, 2016.

Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. FedDG: Federated Domain Generalization on Medical Image Segmentation via Episodic Learning in Continuous Frequency Space. *CVPR*, 2021a.

Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive? *NeurIPS*, 2021b.

Yuning Lu, Jianzhuang Liu, Yonggang Zhang, Yajing Liu, and Xinmei Tian. Prompt Distribution Learning. *CVPR*, 2022.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *ICLR*, 2018.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain Adaptation: Learning Bounds and Algorithms. *COLT*, 2009.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *AISTATS*, 2017.

Sachit Menon and Carl Vondrick. Visual Classification via Description from Large Language Models. *ICLR*, 2023.

Tom Minka. Divergence measures and message passing. *Citeseer*, 2005.

Ishan Misra and Laurens van der Maaten. Self-Supervised Learning of Pretext-Invariant Representations. *CVPR*, 2020.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *PAMI*, 2018.

Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic Federated Learning. *ICML*, 2019.

Ron Mokady, Amir Hertz, and Amit H Bermano. ClipCap: CLIP Prefix for Image Captioning. *arXiv*, 2021.

Thomas Möllenhoff and Mohammad Emtiyaz Khan. SAM as an Optimal Relaxation of Bayes. *ICLR*, 2023.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain Generalization via Invariant Feature Representation. *ICML*, 2013.

Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating Prediction-Time Batch Normalization for Robustness under Covariate Shift. *arXiv*, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

A Tuan Nguyen, Toan Tran, Yarin Gal, and Atilim Gunes Baydin. Domain Invariant Representation Learning with Domain Density Transformations. *NeurIPS*, 2021.

A. Tuan Nguyen, Philip Torr, and Ser-Nam Lim. FedSR: A Simple and Effective Domain Generalization Method for Federated Learning. *NeurIPS*, 2022a.

A Tuan Nguyen, Toan Tran, Yarin Gal, Philip HS Torr, and Atılım Güneş Baydin. KL Guided Domain Adaptation. *ICLR*, 2022b.

A Tuan Nguyen, Thanh Nguyen-Tang, Ser-Nam Lim, and Philip HS Torr. TIPI: Test Time Adaptation with Transformation Invariance. *CVPR*, 2023.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient Test-Time Model Adaptation without Forgetting. *ICML*, 2022.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and Dogs. *CVPR*, 2012.

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. *NeurIPS*, 2019.

Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv*, 2017.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment Matching for Multi-Source Domain Adaptation. *ICCV*, 2019.

Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated Adversarial Domain Adaptation. *ICLR*, 2020.

Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient Domain Generalization via Common-Specific Low-Rank Decomposition. *ICML*, 2020.

Sarah Pratt, Rosanne Liu, and Ali Farhadi. What does a platypus look like? Generating customized prompts for zero-shot image classification. *ICCV*, 2022.

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. *ICML*, 2021.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet Classifiers Generalize to ImageNet? *ICML*, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *CVPR*, 2022.

Walter Rudin. Real and Complex Analysis. *Tata McGraw-hill education*, 2006.

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*, 2015a.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015b.

Paolo Russo, Fabio M Carlucci, Tatiana Tommasi, and Barbara Caputo. From Source to Target and Back: Symmetric Bi-Directional Adaptive GAN. *CVPR*, 2018.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting Visual Category Models to New Domains. *ECCV*, 2010.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning. *NIPS*, 2016.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *NeurIPS*, 2020.

Seonguk Seo, Yumin Suh, Dongwan Kim, Jongwoo Han, and Bohyung Han. Learning to Optimize Domain Specific Normalization for Domain Generalization. *ECCV*, 2019.

Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing Across Domains via Cross-Gradient Training. *ICLR*, 2018.

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein Distance Guided Representation Learning for Domain Adaptation. *AAAI*, 2018.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated Multi-Task Learning. *NIPS*, 2017.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *arXiv*, 2012.

Sebastian U Stich. Local SGD Converges Fast and Communicates Little. *ICLR*, 2019.

Baochen Sun and Kate Saenko. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. *ECCV*, 2016.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. *ICML*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR*, 2014.

Ajay Tanwani. DIRL: Domain-Invariant Representation Learning for Sim-to-Real Transfer. *CoRL*, 2020.

Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 1996.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv*, 2000.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *JMLR*, 2008.

Cristina Vasconcelos, Hugo Larochelle, Vincent Dumoulin, Rob Romijnders, Nicolas Le Roux, and Ross Goroshin. Impact of Aliasing on Generalization in Deep Convolutional Networks . *CVPR*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *NIPS*, 2017.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep Hashing Network for Unsupervised Domain Adaptation. *CVPR*, 2017.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully Test-Time Adaptation by Entropy Minimization. *ICLR*, 2021.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning Robust Global Representations by Penalizing Local Predictive Power. *NeurIPS*, 2019.

Jianyu Wang and Gauri Joshi. Cooperative SGD: A Unified Framework for the Design and Analysis of Local-Update SGD Algorithms. *JMLR*, 2021.

Mei Wang and Weihong Deng. Deep Visual Domain Adaptation: A Survey. *Neurocomputing*, 2018.

Ziqi Wang, Marco Loog, and Jan van Gemert. Respecting Domain Relations: Hypothesis Invariance for Domain Generalization. *ICPR*, 2020.

Max Welling and Yee W Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. *ICML*, 2011.

Andrew Gordon Wilson and Pavel Izmailov. Deep Ensembles as Approximate Bayesian Inference. 2021.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. *CVPR*, 2010.

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial Examples Improve Image Recognition. *CVPR*, 2020.

Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial Domain Adaptation with Domain Mixup. *AAAI*, 2020.

Xiang Xu, Xiong Zhou, Ragav Venkatesan, Gurumurthy Swaminathan, and Orchid Majumder. d-SNE: Domain Adaptation using Stochastic Neighborhood Embedding. *CVPR*, 2019.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. CoCa: Contrastive Captioners are Image-Text Foundation Models. *arXiv*, 2022.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *BMVC*, 2016.

Liling Zhang, Xinyu Lei, Yichun Shi, Hongyu Huang, and Chao Chen. Federated Learning with Domain Generalization. *arXiv*, 2021.

Richard Zhang. Making Convolutional Networks Shift-Invariant Again. *ICML*, 2019.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging Theory and Algorithm for Domain Adaptation. *ICML*, 2019.

Chao Zhao and Weiming Shen. Adversarial Mutual Information-Guided Single Domain Generalization Network for Intelligent Fault Diagnosis. *IEEE Transactions on Industrial Informatics*, 2022.

Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On Learning Invariant Representations for Domain Adaptation. *ICML*, 2019.

Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain Generalization via Entropy Regularization. *NeurIPS*, 2020.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated Learning with Non-IID Data. *arXiv*, 2018.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional Prompt Learning for Vision-Language Models. *CVPR*, 2022a.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to Prompt for Vision-Language Models. *IJCV*, 2022b.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *ICCV*, 2017.

Yongchun Zhu, Fuzhen Zhuang, Jindong Wang, Guolin Ke, Jingwu Chen, Jiang Bian, Hui Xiong, and Qing He. Deep Subdomain Adaptation Network for Image Classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Appendix A

Domain Invariant Representation Learning with Domain Density Transformations

A.1 Proofs

For the following proofs, we treat the variables as continuous variables and always use the integral. If one or some of the variables are discrete, it is straightforward to replace the corresponding integral(s) with summation sign(s) and the proofs still hold.

A.1.1 Remark 2.1

Proof.

i) If there exists a representation z defined by the mapping $p(z|x)$ that aligns both the marginal and conditional distribution, then $\forall d, d', y$ we have:

$$\begin{aligned} p(y, z|d) &= p(z|d)p(y|z, d) \\ &= p(z|d')p(y|z, d') = p(y, z|d'). \end{aligned} \tag{A.1}$$

By marginalizing both sides of Eq A.1 over z, we get $p(y|d) = p(y|d')$.

ii) If $p(y|d)$ is unchanged w.r.t. the domain d, then we can always find a domain invariant representation, for example, $p(z|x) = \delta_0(z)$ for the deterministic case (that maps all x to 0), or $p(z|x) = \mathcal{N}(z; 0, 1)$ for the probabilistic case.

These representations are trivial and not of our interest since they are uninformative of the input x. However, the readers can verify that they do align both the marginal and conditional distributions of data.

□

A.1.2 Remark 2.2

Proof.

- If $I(z,d)=0$, then $p(z|d)=p(z)$, which means $p(z|d)$ is invariant w.r.t. d.
- If $p(z|d)$ is invariant w.r.t. d, then $\forall z,d$:

$$\begin{aligned} p(z) &= \int p(z|d')p(d')\mathrm{d}d' = \int p(z|d)p(d')\mathrm{d}d' \\ &\quad (\text{since } p(z|d') = p(z|d)\forall d') \\ &= p(z|d)\int p(d')\mathrm{d}d' = p(z|d) \\ &\implies I(z,d)=0. \end{aligned}$$

□

A.1.3 Theorem 2.1

Proof.

i) Marginal alignment: $\forall z$ we have:

$$\begin{aligned} p(z|d) &= \int p(x|d)p(z|x)\mathrm{d}x \\ &= \int p(f_{d',d}(x')|d)p(z|f_{d',d}(x'))\left|\det\, J_{f_{d',d}}(x')\right|\mathrm{d}x' \end{aligned}$$

(by applying variable substitution in multiple integral: $x'=f_{d,d'}(x)$)

$$= \int p(x'|d')\left|\det\, J_{f_{d',d}}(x')\right|^{-1} p(z|x')\left|\det\, J_{f_{d',d}}(x')\right|\mathrm{d}x'$$

(since $p(f_{d',d}(x')|d) = p(x'|d')\left|\det\, J_{f_{d',d}}(x')\right|^{-1}$ and $p(z|f_{d',d}(x')) = p(z|x')$)

$$\begin{aligned} &= \int p(x'|d')p(z|x')\mathrm{d}x' \\ &= p(z|d'). \end{aligned}$$

ii) Conditional alignment: $\forall z,y$ we have:

$$\begin{aligned} p(z|y,d) &= \int p(x|y,d)p(z|x)\mathrm{d}x \\ &= \int p(f_{d',d}(x')|y,d)p(z|f_{d',d}(x'))\left|\det\, J_{f_{d',d}}(x')\right|\mathrm{d}x' \end{aligned}$$

(by applying variable substitution in multiple integral: $x' = f_{d,d'}(x)$)

$$= \int p(x'|y,d') \left|\det\ J_{f_{d',d}}(x')\right|^{-1} p(z|x') \left|\det\ J_{f_{d',d}}(x')\right| \mathrm{d}x'$$

(since $p(f_{d',d}(x')|y,d) = p(x'|y,d') \left|\det\ J_{f_{d',d}}(x')\right|^{-1}$ and $p(z|f_{d',d}(x')) = p(z|x')$)

$$= \int p(x'|y,d')p(z|x')\mathrm{d}x'$$
$$= p(z|y,d').$$

Note that

$$p(y|z,d) = \frac{p(y,z|d)}{p(z|d)} = \frac{p(y|d)p(z|y,d)}{p(z|d)}.$$

Since $p(y|d) = p(y) = p(y|d'), p(z|y,d) = p(z|y,d')$ and $p(z|d) = p(z|d')$, we have:

$$p(y|z,d) = \frac{p(y|d')p(z|y,d')}{p(z|d')} = p(y|z,d').$$

□

A.2 Discussion on the choice of the distance metric between representations

Table A.1: **Ablation study**: Rotated MNIST experiments with $\mathcal{M}_{75}$ as the target domain.

Distance Metric	Accuracy
Squared Error Distance	97.1±0.3
Contrastive Distance	95.8±0.9
Cosine Distance	90.1±0.3

As discussed in Section 2.3.2, we enforce the representation network g_θ to be invariant under the domain transformation $f_{d,d'}$ (with any two domains d, d'), meaning that $g_\theta(x) = g_\theta(f_{d,d'}(x))$.

In our implementation, we use the squared error distance as the distance between $g_\theta(x)$ and $g_\theta(f_{d,d'}(x))$. Admittedly, this distance tends to have the side effect of making the norm of the representation smaller. However, as visualized in Section 2.5.3, it does successfully align the distributions of the representation.

We also considered other distances such as contrastive distance and the cosine distance. We present in Table A.1 an ablation study with different choices of the distance metrics, for the Rotated Mnist experiment with the target domain $\mathcal{M}_{75}$. Note that in this Rotated Mnist dataset, domains $\mathcal{M}_{75}$ and $\mathcal{M}_0$ are (equally) the hardest target domains. Therefore, we choose $\mathcal{M}_{75}$ for this ablation study to compare the performance of the variants.

As the Squared Error Distance gives the best performance and also is the most stable one in practice, we decide use it in our implementation.

Appendix B

KL Guided Domain Adaptation

B.1 Proofs

For the following proofs, we treat the variables as continuous variables and always use the integral. If one or some of the variables are discrete, it is straightforward to replace the corresponding integral(s) with summation sign(s) and the proofs still hold.

B.1.1 Proposition 3.1

Proof. We have:

$$\begin{aligned} l_{test} &\leq \mathbb{E}_{p_T(z,y)}[-\log \hat{p}(y|z)] \\ &= \int -\log \hat{p}(y|z) p_T(z,y) dzdy \\ &= \int -\log \hat{p}(y|z) p_S(z,y) dzdy + \int -\log \hat{p}(y|z) \left[p_T(z,y) - p_S(z,y)\right] dzdy \\ &= l_{train} + \int -\log \hat{p}(y|z) \left[p_T(z,y) - p_S(z,y)\right] dzdy. \end{aligned}$$

Let $\mathcal{A} = \{(z,y)|p_T(z,y) - p_S(z,y) \geq 0\}$ and $\mathcal{B} = \{(z,y)|p_T(z,y) - p_S(z,y) < 0\}$, using

the fact that $-\log \hat{p}(y|z) \geq 0 \; \forall z \in \mathcal{Z}, y \in \mathcal{Y}$ we have:

$$
\begin{aligned}
&\int -\log \hat{p}(y|z) \left[p_T(z,y) - p_S(z,y)\right] dzdy \\
&= \int_{\mathcal{A}} -\log \hat{p}(y|z) \left[p_T(z,y) - p_S(z,y)\right] dzdy + \int_{\mathcal{B}} -\log \hat{p}(y|z) \left[p_T(z,y) - p_S(z,y)\right] dzdy \\
&\leq \int_{\mathcal{A}} -\log \hat{p}(y|z) \left[p_T(z,y) - p_S(z,y)\right] dzdy \\
&= \int_{\mathcal{A}} -\log \hat{p}(y|z) \left|p_T(z,y) - p_S(z,y)\right| dzdy \\
&\leq M \int_{\mathcal{A}} \left|p_T(z,y) - p_S(z,y)\right| dzdy \\
&\quad \text{(since} -\log \hat{p}(y|z) \leq M\text{)},
\end{aligned}
$$

where $|.|$ is the absolute value.

Here, $\int_{\mathcal{A}} \left|p_T(z,y) - p_S(z,y)\right| dzdy$ is also called the total variation of the two distributions $p_T(z,y)$ and $p_S(z,y)$.

Note that:

$$
\begin{aligned}
&\int p_T(z,y) - p_S(z,y) dzdy = 0 \\
\Leftrightarrow &\int_{\mathcal{A}} p_T(z,y) - p_S(z,y) dzdy + \int_{\mathcal{B}} p_T(z,y) - p_S(z,y) dzdy = 0 \\
\Leftrightarrow &\int_{\mathcal{A}} p_T(z,y) - p_S(z,y) dzdy = \int_{\mathcal{B}} p_S(z,y) - p_T(z,y) dzdy \\
\Leftrightarrow &\int_{\mathcal{A}} \left|p_T(z,y) - p_S(z,y)\right| dzdy = \int_{\mathcal{B}} \left|p_T(z,y) - p_S(z,y)\right| dzdy \\
\Leftrightarrow &\int_{\mathcal{A}} \left|p_T(z,y) - p_S(z,y)\right| dzdy = \frac{1}{2} \int \left|p_T(z,y) - p_S(z,y)\right| dzdy.
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
l_{test} &\leq l_{train} + M \int_{\mathcal{A}} \left|p_T(z,y) - p_S(z,y)\right| dzdy \\
&= l_{train} + \frac{M}{2} \int \left|p_T(z,y) - p_S(z,y)\right| dzdy.
\end{aligned}
$$

Using the Pinsker's inequality, we have:

$$
\left(\int \left|p_T(z,y) - p_S(z,y)\right| dzdy \right)^2 \leq 2 \int p_T(z,y) \log \frac{p_T(z,y)}{p_S(z,y)} dzdy.
$$

Therefore, we finally have:

$$
\begin{aligned}
l_{test} &\leq l_{train} + \frac{M}{2}\sqrt{2\int p_T(z,y)\log\frac{p_T(z,y)}{p_S(z,y)}dzdy} \\
&= l_{train} + \frac{M}{\sqrt{2}}\sqrt{\mathrm{KL}[p_T(z,y)|p_S(z,y)]},
\end{aligned}
$$

which concludes our proof.

Also note that the KL divergence between $p_T(z,y)$ and $p_S(z,y)$ can further be decomposed into the marginal misalignment and conditional misalignment as follows:

$$
\begin{aligned}
&\mathrm{KL}[p_T(z,y)|p_S(z,y)] \\
&= \mathbb{E}_{p_T(z,y)}[\log p_T(z,y) - \log p_S(z,y)] \\
&= \mathbb{E}_{p_T(z,y)}[\log p_T(z) + \log p_T(y|z) - \log p_S(z) - \log p_S(y|z)] \\
&= \mathbb{E}_{p_T(z,y)}[\log p_T(z) - \log p_S(z)] + \mathbb{E}_{p_T(z,y)}[\log p_T(y|z) - \log p_S(y|z)] \\
&= \mathbb{E}_{p_T(z)}[\log p_T(z) - \log p_S(z)] + \mathbb{E}_{p_T(z)}\left[\mathbb{E}_{p_T(y|z)}[\log p_T(y|z) - \log p_S(y|z)]\right] \\
&= \mathrm{KL}[p_T(z)|p_S(z)] + \mathbb{E}_{p_T(z)}\left[\mathrm{KL}[p_T(y|z)|p_S(y|z)]\right].
\end{aligned}
$$

□

B.1.2 Proposition 3.2

Proof. According to Assumption 3.1, we have:

$$
\begin{aligned}
& I_S(z,y) = I_S(x,y) \\
\Leftrightarrow & H_S(y) - H_S(y|z) = H_S(y) - H_S(y|x) \\
\Leftrightarrow & H_S(y|z) = H_S(y|x) \\
\Leftrightarrow & \mathbb{E}_{p_S(z,y)}[\log p_S(y|z)] = \mathbb{E}_{p_S(x,y)}[\log p_S(y|x)] \\
\Leftrightarrow & \mathbb{E}_{p_S(x,z,y)}[\log p_S(y|z)] = \mathbb{E}_{p_S(x,y)}[\log p_S(y|x)] \\
\Leftrightarrow & \mathbb{E}_{p_S(x,y)}\left[\mathbb{E}_{p(z|x)}[\log p_S(y|z)]\right] = \mathbb{E}_{p_S(x,y)}[\log p_S(y|x)] \\
\Leftrightarrow & \mathbb{E}_{p_S(x,y)}\left[\log p_S(y|x) - \mathbb{E}_{p(z|x)}[\log p_S(y|z)]\right] = 0.
\end{aligned}
$$

According to Assumption 3.2, $\forall x \in \mathcal{X}, y \in \mathcal{Y}$ we have:

$$
\begin{aligned}
& p_S(y|x) = \mathbb{E}_{p(z|x)}[p_S(y|z)] \\
\Leftrightarrow & \log p_S(y|x) = \log \mathbb{E}_{p(z|x)}[p_S(y|z)] \\
\Rightarrow & \log p_S(y|x) \geq \mathbb{E}_{p(z|x)}[\log p_S(y|z)].
\end{aligned}
$$

Since $\frac{p_T(x,y)}{p_S(x,y)} < \infty$, there exists $N > 0$ such that $\frac{p_T(x,y)}{p_S(x,y)} \leq N \; \forall x \in \mathcal{X}, y \in \mathcal{Y}$. Therefore:

$$\begin{aligned}
&\mathbb{E}_{p_T(x,y)} \left[\log p_S(y|x) - \mathbb{E}_{p(z|x)}[\log p_S(y|z)]\right] \\
=&\mathbb{E}_{p_S(x,y)} \left[\left(\log p_S(y|x) - \mathbb{E}_{p(z|x)}[\log p_S(y|z)]\right) \frac{p_T(x,y)}{p_S(x,y)}\right] \\
\leq& N.\mathbb{E}_{p_S(x,y)} \left[\log p_S(y|x) - \mathbb{E}_{p(z|x)}[\log p_S(y|z)]\right] \\
=&0.
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\mathbb{E}_{p_T(x,y)} \left[\log p_S(y|x) - \mathbb{E}_{p(z|x)}[\log p_S(y|z)]\right] = 0 \\
\Leftrightarrow&\mathbb{E}_{p_T(x,y)} \left[\log p_S(y|x)\right] = \mathbb{E}_{p_T(x,y,z)}[\log p_S(y|z)] \\
\Leftrightarrow&\mathbb{E}_{p_T(x,y)} \left[\log p_S(y|x)\right] = \mathbb{E}_{p_T(z,y)}[\log p_S(y|z)].
\end{aligned} \tag{B.1}$$

We have:

$$\begin{aligned}
&\mathbb{E}_{p_T(z)} \left[\mathrm{KL}[p_T(y|z)|p_S(y|z)]\right] \leq \mathbb{E}_{p_T(x)} \left[\mathrm{KL}[p_T(y|x)|p_S(y|x)]\right] \\
\Leftrightarrow&\mathbb{E}_{p_T(z,y)} \left[\log p_T(y|z) - \log p_S(y|z)\right] \leq \mathbb{E}_{p_T(x,y)} \left[\log p_T(y|x) - \log p_S(y|x)\right].
\end{aligned}$$

Using Eq B.1, we now only need to prove that:

$$\begin{aligned}
&\mathbb{E}_{p_T(z,y)} \left[\log p_T(y|z)\right] \leq \mathbb{E}_{p_T(x,y)} \left[\log p_T(y|x)\right] \\
\Leftrightarrow& - H_T(y|z) \leq -H_T(y|x) \\
\Leftrightarrow& H_T(y) - H_T(y|z) \leq H_T(y) - H_T(y|x) \\
\Leftrightarrow& I_T(z,y) \leq I_T(x,y) \\
&\quad \text{(always true based on the Data Processing Inequality).}
\end{aligned}$$

□

B.2 Review of existing generalization bounds

There have been several works studying the generalization bounds of the Domain Adaptation problem. We briefly review the most important and common ones here with a discussion about their differences from our proposed bound.

B.2.1 [Ben-David et al., 2010]

Ben-David et al. [2010] consider a binary classification problem. Let x be the input with the support set $\mathcal{X}$ and y be the binary label with the support set $\mathcal{Y} = \{0, 1\}$. Consider a source domain with a distribution P_X^s over the input x and the true labeling function $f^s : \mathcal{X} \to \{0, 1\}$; and similarly a target domain with a distribution P_X^t over the input x and the true labeling function $f^t : \mathcal{X} \to \{0, 1\}$. Note that the authors claim that this labeling function can be probabilistic; in that case, $f : \mathcal{X} \to [0, 1]$ denoting the probability. However, we argue that this probabilistic setting is impractical since we would not know that true underlying function in order to calculate/estimate the bounds in practice). Therefore, we found that the bound is only practical for the case of a deterministic labeling mechanism.

The error of the classifier h, which is also a deterministic labeling function, on the source domain is:

$$\epsilon^s(h) = \mathbb{E}_{x \sim P_X^s}[|h(x) - f^s(x)|],$$

and similarly for the target domain:

$$\epsilon^t(h) = \mathbb{E}_{x \sim P_X^t}[|h(x) - f^t(x)|].$$

Here $|.|$ is the absolute value, which means the loss of a data point is the l_1 distance of the labels.

Consider a hypothesis space $\mathcal{H}$ and let a classifier h be any function from that space. The first theorem in [Ben-David et al., 2010] offers a bound of the target loss $\epsilon^t(h)$ based on the source loss $\epsilon^s(h)$, and the total variation between P_X^s and P_X^t, and the difference between the two labeling function f^s and f^t:

Theorem B.1 ([Ben-David et al., 2010])**.**

$$\epsilon^t(h) \leq \epsilon^s(h) + 2d_1(P_X^s, P_X^t) + \min_{P_X \in \{P_X^s, P_X^t\}} \mathbb{E}_{x \sim P_X}[|f^s(x) - f^t(x)|],$$

where $d_1(P_X^s, P_X^t)$ is the total variational distance, i.e., $d_1(P_X^s, P_X^t) := \sup_{\mathcal{A} \in \mathscr{X}}[P_X^s(\mathcal{A}) - P_X^t(\mathcal{A})]$, and $\mathscr{X}$ is the sigma-field of $\mathcal{X}$ (set of all subsets of $\mathcal{X}$).

In this theorem, the term $2d_1(P_X^s, P_X^t)$ presents the marginal misalignment and $\min_{P_X \in \{P_X^s, P_X^t\}} \mathbb{E}_{x \sim P_X}[|f^s(x) - f^t(x)|]$ is the conditional misalignment.

Ben-David et al. [2010] also propose another bound based on a variant of the $\mathcal{H}$-divergence, which is presented in the following theorem:

Theorem B.2 ([Ben-David et al., 2010])**.**

$$\epsilon^t(h) \leq \epsilon^s(h) + d_{\mathcal{H}\Delta\mathcal{H}}(P_X^s, P_X^t) + \lambda_{\mathcal{H}},$$

where the $\mathcal{H}\Delta\mathcal{H}$*-divergence* $d_{\mathcal{H}\Delta\mathcal{H}}(P_X^s, P_X^t) := \sup_{h_1,h_2\in\mathcal{H}}|Pr_{x\sim P_X^s}[h_1(x) \neq h_2(x)] - Pr_{x\sim P_X^t}[h_1(x) \neq h_2(x)]|$*; and* $\lambda_{\mathcal{H}} = \inf_{h\in\mathcal{H}}[\epsilon^s(h) + \epsilon^t(h)]$.

The $\mathcal{H}\Delta\mathcal{H}$-divergence $d_{\mathcal{H}\Delta\mathcal{H}}(P_X^s, P_X^t)$ replaces the total variation to measure the marginal misalignment of the two domains. Meanwhile, $\lambda_{\mathcal{H}}$ measures the conditional misalignment of the two domains (if the two true labeling functions f^s and f^t are the same and belong to the hypothesis space $\mathcal{H}$, this quantity is zero).

The above bounds can also be applied to the representation space (similar to ours), leading to the same bounds where the input x is replaced by its representation z. However, as mentioned in the main text, it is not clear if the conditional misalignment in the representation space (e.g., $\min_{P_Z\in\{P_Z^s,P_Z^t\}} \mathbb{E}_{z\sim P_Z}[|g^s(z) - g^t(z)|]$) is bounded or not.

Difference to our bound First of all, Ben-David et al. [2010] only consider a binary classification problem. Moreover, as discussed above, the bounds in [Ben-David et al., 2010] are only practical with deterministic labeling mechanisms for both domains. This assumption is hard to be true for most datasets since the labeling mechanism is usually probabilistic. This makes it not generalizable to the general case of supervised learning. Furthermore, the loss function is a l_1 distance between the labeling function, which is also not a common choice in practice, which makes it challenging to generalize to the multiclass classification set-up (even if there exists a deterministic labeling function for a multiclass dataset, using the l_1 loss for the one-hot encoded labels would be unreasonable; the common loss function in practice for the multiclass classification problem is the cross-entropy loss). Finally, the total variation and $\mathcal{H}$-divergence might be hard to estimate in practice since it requires the computation of a supremum.

B.2.2 [Mansour et al., 2009]

Mansour et al. [2009] consider a more flexible problem set-up than Ben-David et al. [2010]. Specifically, instead of $\mathcal{Y} = \{0, 1\}$, they consider the cases where $\mathcal{Y} = \{0, 1\}$ (for binary classification) or $\mathcal{Y}$ is a measurable subset of $\mathbb{R}$ (for regression). Note that their bound still cannot work for multiclass classification. They also generalize the l_1 loss function to a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; however, this loss function must

obey the triangle inequality. Although the l_1 distance satisfies this inequality, it is not generally true for other common loss functions in practice (e.g., cross-entropy). They still consider **deterministic** labeling functions f^s and f^t for the source and target domain.

Similar to [Ben-David et al., 2010], with a hypothesis h from the hypothesis space $\mathcal{H}$, the error of the source and target domain are:

$$\epsilon^s(h) = \mathbb{E}_{x\sim P_X^s}[L(h(x), f^s(x))],$$
$$\epsilon^t(h) = \mathbb{E}_{x\sim P_X^t}[L(h(x), f^t(x))].$$

For convenience, denote also the error between two labeling functions h and h' in the source and target distribution as:

$$\epsilon^s(h, h') = \mathbb{E}_{x\sim P_X^s}[L(h(x), h'(x))],$$
$$\epsilon^t(h, h') = \mathbb{E}_{x\sim P_X^t}[L(h(x), h'(x))],$$

which means $\epsilon^s(h) = \epsilon^s(h, f^s)$ and $\epsilon^t(h) = \epsilon^t(h, f^t)$.

Also, let h^{*s} and h^{*t} be the minimizer of $\epsilon^s(h)$ and $\epsilon^t(h)$ respectively. In particular:

$$h^{*s} = \arg\min_{h\in\mathcal{H}} \epsilon^s(h) = \arg\min_{h\in\mathcal{H}} \mathbb{E}_{x\sim P_X^s}[L(h(x), f^s(x))],$$
$$h^{*t} = \arg\min_{h\in\mathcal{H}} \epsilon^t(h) = \arg\min_{h\in\mathcal{H}} \mathbb{E}_{x\sim P_X^t}[L(h(x), f^t(x))].$$

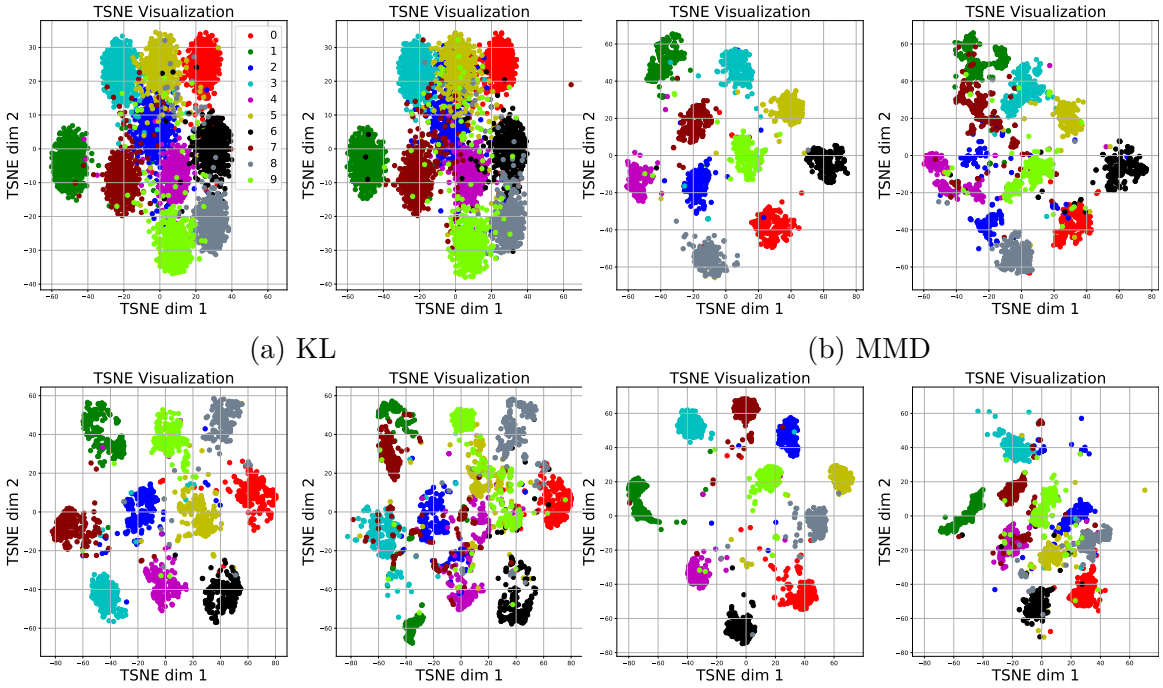
Mansour et al. [2009] introduce a generalization bound as follows:

Theorem B.3 ([Mansour et al., 2009])**.** *Assume that the loss function L is symmetric and obeys the triangle inequality. Then, for any hypothesis $h \in \mathcal{H}$, the following holds:*

$$\epsilon^t(h) \leq \epsilon^t(h^{*t}) + \epsilon^s(h, h^{*s}) + disc(P_X^s, P_X^t) + \epsilon^t(h^{*s}, h^{*t}),$$

where $disc(P_X^s, P_X^t) := \sup_{h,h'\in\mathcal{H}}|\epsilon^s(h, h') - \epsilon^t(h, h')|$, which is a generalized version of the $\mathcal{H}\Delta\mathcal{H}$-divergence.

Here, the first term $\epsilon^t(h^{*t})$ is the ideal target loss (will be zero if the hypothesis space $\mathcal{H}$ contains f_t), the second term $\epsilon^s(h, h^{*s})$ will be zero if we choose $h = h^{*s}$ (which is the common practice, e.g., train the classifier h on the source domain), the third term $\text{disc}(P_X^s, P_X^t)$ measures the marginal misalignment, and the final term $\epsilon^t(h^{*s}, h^{*t})$ is somewhat an indicator of the conditional misalignment (becomes zero if $f^s = f^t \in \mathcal{H}$).



(a) KL (b) MMD

(c) DANN (d) ERM

Figure B.1: Visualization using t-SNE of the representation space of our method KL and the baselines MMD, DANN, ERM. For each method, the left subfigure corresponds to the source domain $\mathcal{M}_0$ and the right one corresponds to the target domain $\mathcal{M}_{45}$. Each color represents a digit class.

Difference to our bound The above bound is based on the ideal target loss, while our bound is based on the source loss. In practice, we have (an estimate of) the source loss calculated on the source domain's training set; meanwhile, the ideal target loss is unknown. This makes the above bound less useful in practice compared to ours. Furthermore, the above bound has similar problems as the ones in [Ben-David et al., 2010]: it does not work for multiclass classification, it assumes a deterministic labeling mechanism (which does not hold in practice), it assumes the loss function obeys the triangle inequality (which generally is not true in practice), and it contains terms that are not easy to compute in practice (supremum and infimum).

B.3 Additional Experimental Results

B.3.1 Visualization of the RotatedMNIST experiments

Figure B.1 shows the representation space (for the RotatedMNIST experiment with source $\mathcal{M}_0$ and target $\mathcal{M}_{45}$) of our method compared to the baselines, visualized using

Table B.1: PACS experiments.

	Model						
Experiments	ERM	ERM (prob)	DANN	MMD	CORAL	WD	KL (ours)
A → C	66.1±1.3	63.5±0.8	71.0±3.2	**79.5±0.4**	62.7±10.4	76.2±0.9	73.1±3.4
A → P	94.3±0.6	93.5±1.3	94.5±0.5	94.5±1.1	86.3±6.8	92.4±1.3	**95.4±1.2**
A → S	53.6±0.8	60.9±3.5	58.6±12.8	62.1±2.0	46.2±3.5	53.9±2.7	**67.4±1.9**
C → A	69.7±1.1	70.8±2.3	76.4±1.7	79.5±3.0	75.9±0.9	69.0±2.1	**83.3±1.1**
C → P	82.0±0.9	81.5±2.1	78.6±3.4	80.8±2.3	78.3±3.6	72.9±8.6	**83.1±7.4**
C → S	72.2±1.4	70.4±1.5	**76.1±1.0**	74.1±1.3	56.9±11.0	48.7±6.1	68.2±0.5
P → A	65.7±2.3	63.3±1.2	68.0±2.7	67.7±1.8	70.0±1.5	62.6±1.5	**75.5±2.5**
P → C	29.1±1.9	27.2±3.3	50.7±5.0	47.4±0.8	47.5±8.6	56.1±1.4	**67.7±1.2**
P → S	38.0±1.0	35.9±2.3	29.3±9.8	59.7±4.8	15.8±5.3	22.3±15.0	**64.5±2.1**
S → A	41.3±6.5	40.9±3.9	39.2±3.5	40.0±3.3	39.1±4.8	36.1±9.5	**48.2±2.4**
S → C	66.7±1.0	**67.9±1.4**	64.3±2.0	65.7±2.3	59.9±1.5	60.5±2.0	63.5±0.4
S → P	**49.3±3.3**	46.0±4.7	44.3±4.0	45.1±0.9	37.4±2.7	38.5±5.6	39.1±3.4
Average	60.6	60.2	62.6	66.3	56.3	57.4	**69.1**

t-SNE [Van der Maaten and Hinton, 2008]. We can clearly see that the color clusters (which correspond to the digit classes) of our method are much more aligned when compared to other baselines such as MMD, DANN, and ERM. This illustrates the effectiveness of our method in aligning the representation in the domain adaptation problem.

B.3.2 PACS

Table B.1 presents the results for PACS, which is a challenging real-world dataset for domain adaptation/generalization. In this dataset, our model outperforms the ERM baselines by roughly 9% on average, indicating the effectiveness of our representation-alignment technique. Our method is the best performer (with a large margin) on 8 out of 12 experiments, showing a clear benefit over other representation alignment techniques. Together with our method, MMD again performs the best among the representation-alignment baselines (DANN, MMD, CORAL, and WD), confirming that a stable training procedure (with no minimax objectives in MMD and our model) is important and often leads to better results. It is also worth noting that our model still outperforms MMD despite being less computationally expensive (in this implementation, MMD needs to compute seven Gaussian kernels for each of three pairs of representation sets in each minibatch).

Interestingly, the ERM baselines perform the best in some experiments (e.g., S → C, S → P). This result also agrees with the one observed in [Gulrajani and Lopez-

Table B.2: Office-31 experiments

Model	A→D	A→W	D→A	D→W	W→A	W→D	Average
DANN	79.7±0.4	82.0±0.4	68.2±0.4	96.9±0.2	67.4±0.5	99.1±0.1	82.2
MMD	75.5±0.6	73.4±0.4	60.8±1.0	97.4±0.5	61.5±1.1	99.5±0.2	78.0
KL	85.6±0.6	87.9±0.4	70.1±1.1	99.0±0.2	69.3±0.7	100.0±0.0	85.3

Table B.3: **Ablation study: Effect of batch size**. Rotated MNIST experiments with $\mathcal{M}_0$ source and $\mathcal{M}_{45}$ target.

Batch size	256	128	64	32
KL (ours)	93.4±0.8	93.4±1.2	93.3±0.3	89.5±0.9

Paz, 2021] that domain generalization/adaptation techniques might have negative effects when applied unsuccessfully. It should be noted that the S (sketch) domain is undoubtedly the most different compared to others (only contains black sketches on white backgrounds while other domains have colors), which might explain the difficulty when learning to transfer between domains.

B.3.3 Office-31

We also provide additional results on the Office-31 dataset [Saenko et al., 2010]. Table B.2 presents the performance of our model and two of the best-performing baselines, namely MMD and DANN. Although our method is not state-of-the-art in this dataset (and understandingly so), it significantly outperforms the relevant baselines considered in this work.

B.3.4 Ablation Study: Effect of batch size

In this subsection, we conduct an ablation study to investigate the effect of the batch size on our model's performance. Table B.3 shows the performance of our method on the RotatedMNIST dataset, with $\mathcal{M}_0$ as the source domain and $\mathcal{M}_{45}$ as the target domain and with various choices of the batch size. As expected, our model's performance tends to benefit from a bigger batch size, since it would alleviate the bias of our objective estimator. We therefore recommend increasing the batch size whenever possible. However, our model performs well even for a batch size as small as 64 (which is considered small in this era of deep learning).

Table B.4: **Ablation study: Effect of auxiliary term**. RotatedMNIST.

Model	$\mathcal{M}_{15}$	$\mathcal{M}_{30}$	$\mathcal{M}_{45}$	$\mathcal{M}_{60}$	$\mathcal{M}_{75}$	Average
KL ($\beta_{aux} = 0$)	97.8±0.5	96.6±0.4	92.0±0.4	68.8±4.6	62.3±4.2	83.5
KL (reported)	97.8±0.1	97.1±0.2	93.4±0.8	75.5±2.4	68.1±1.8	86.4

Table B.5: **Ablation study: Effect of auxiliary term**. DIGITS and VisDA17.

	DIGITS				VisDA17
Model	M → U	U → M	S → M	Average	S → R
KL ($\beta_{aux} = 0$)	98.2±0.2	97.1±0.4	90.0±1.1	95.1	67.8
KL (reported)	98.2±0.2	97.3±0.52	92.5±0.9	96.0	70.5

B.3.5 Ablation Study: Effect of the auxiliary forward KL

In this subsection, we investigate the contribution of the auxiliary forward KL term, by considering a variant of our method without this term ($\beta_{aux} = 0$). Table B.4 and Table B.5 show the results of this ablation experiment. Clearly, most of the improvement comes from the reverse KL regularizer term. And even without the auxiliary forward KL term, our results are still significantly better than the baselines. Note that the auxiliary term requires virtually no extra computation, so we believe adding the auxiliary term is reasonable.

B.4 Detailed Experimental Settings

In each experiment, we split both the source and the target data into two portions: 80% and 20%. We use 80% of the source domain data and 80% of the target domain data (without the labels) as the training data. We use the remaining 20% of the source data as the validation set, and the remaining 20% of the target domain data as the test set. Note that we do not use the labeled data from the target domain during training or validation. This evaluation protocol is recommended by Gulrajani and Lopez-Paz [2021].

B.4.1 Baselines

ERM [**Bousquet et al., 2003**] is the typical empirical risk minimization training procedure, meaning that the model is trained normally in the training data and does not account for the distribution shift (domain adaptation).

ERM (prob): since we use a probabilistic network, we also include the probabilistic version of ERM. This is similar to ERM but uses a probabilistic representation network (same as ours).

DANN [Ganin et al., 2016] utilizes a discriminator to distinguish the representation from the source and target domains. It uses an adversarial loss to enforce that the distributions of the representation from the source domain and the target domain are the same.

MMD [Li et al., 2018c] uses the maximum mean discrepancy (MMD) to align the representation's distributions.

CORAL [Sun and Saenko, 2016] aligns the representation distributions of the source and target domains by matching their first two moments.

WD [Shen et al., 2018] uses the Wasserstein distance to match the distribution of the representation.

B.4.2 Representation Network used in RotatedMnist and DIGITS

We use a simple CNN as the representation network in this experiment. This network is exactly the same as the one used in [Gulrajani and Lopez-Paz, 2021].

Our code is in PyTorch. The network is constructed by the following layers, where `output_dim=256` for a deterministic representation network and `output_dim=512` for a probabilistic representation network (256 for μ and 256 for σ^2):

- `Conv2d(in_channels=1, out_channels=64, kernel_size=3, stride=1, padding=1)`
- `ReLU()`
- `GroupNorm(num_groups=8, num_channels=64)`
- `Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=2, padding=1)`
- `ReLU()`
- `GroupNorm(num_groups=8, num_channels=128)`
- `Conv2d(in_channels=128, out_channels=128, kernel_size=3, stride=1, padding=1)`
- `ReLU()`
- `GroupNorm(num_groups=8, num_channels=128)`
- `Conv2d(in_channels=128, out_channels=output_dim, kernel_size=3,`

```
stride=1, padding=1)
- ReLU()
- GroupNorm(num_groups=8, num_channels=output_dim)
- AdaptiveAvgPool2d(output_size=(1,1))
```

B.4.3 Hyper-parameters tuning

We train each model for 100 epochs. To avoid hyperparameter bias, we tune the hyperparameters (learning rate, regularizer coefficients, weight decay, representation dimension, and dropout rate) for each method and dataset independently. Following Gulrajani and Lopez-Paz [2021], we perform a random search [Bergstra and Bengio, 2012] to tune hyperparameters for the baselines. We use the Adam optimizer [Kingma and Ba, 2015] for all the models. We re-run each set of hyperparameters three times. We train all models on an NVIDIA Quadro RTX 6000 GPU.

The readers can also refer to our source code for the experiment setting.

Below are the hyper-parameters considered by the random search in our experiments for each baseline. $\{.\}$ means a set of hyper-parameters considered, while $[.,.]$ means a range of hyper-parameters considered.

B.4.3.1 RotatedMNIST and DIGITS

The representation's dimension is 128 (details in Section B.4.2). For even more details about the below hyper-parameters, please refer to our provided code.

- **ERM**: learning rate: $[10^{-4.5}, 10^{-2.5}]$, weight decay: 0.0, dropout rate: 0.0, batch size: $[8, 512]$, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3.

- **ERM (prob)**: learning rate: $[10^{-4.5}, 10^{-2.5}]$, weight decay: 0.0, dropout rate: 0.0, batch size: $[8, 512]$, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3.

- **DANN [Ganin et al., 2016]**: learning rate: $[10^{-4.5}, 10^{-2.5}]$, weight decay: 0.0, dropout rate: 0.0, batch size: $[8, 512]$, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, adversarial loss coefficient: $[10^{-2}, 10^{2}]$, weight decay of discriminator: $[10^{-6}, 10^{-2}]$, number of discriminator steps per generator steps: $\{1, 2, 4, 8\}$, grad penalty coefficient: $[10^{-2}, 10^{1}]$

- **MMD [Li et al., 2018c]**: learning rate: $[10^{-4.5}, 10^{-2.5}]$, weight decay: 0.0, dropout rate: 0.0, batch size: $[8, 512]$, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, MMD coefficient: $[10^{-3}, 10^{-1}]$

- **CORAL** [Sun and Saenko, 2016]: learning rate: $[10^{-4.5}, 10^{-2.5}]$, weight decay: 0.0, dropout rate: 0.0, batch size: $[8, 512]$, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, CORAL loss coefficient: $[10^{-3}, 10^{-1}]$

- **WD** [Shen et al., 2018]: learning rate: $[10^{-4.5}, 10^{-2.5}]$, weight decay: 0.0, dropout rate: 0.0, batch size: $[8, 512]$, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, wasserstein distance coefficient: $[10^{-2}, 10^{2}]$, weight decay of network f: $[10^{-6}, 10^{-2}]$, number of f optimization steps per normal optimization steps: $\{1, 2, 4, 8\}$, grad penalty coefficient: $[10^{-2}, 10^{1}]$

- **KL (ours)**: learning rate: $[10^{-4.5}, 10^{-2.5}]$, weight decay: 0.0, dropout rate: 0.0, batch size: 256, number of layers of the classifier ($\hat{p}(y|z)$): 1, $\beta : 0.3$, $\beta_{aux} : 0.1$

B.4.3.2 VisDA17 and PACS

The representation network for VisDA17 is a Resnet50, while the one for PACS is a Resnet18. For even more details about the below hyper-parameters, please refer to our provided code.

- **ERM**: learning rate: $[10^{-5}, 10^{-3.5}]$, weight decay: $[10^{-6}, 10^{-2}]$, dropout rate: {0.0, 0.1, 0.5}, batch size: $[8, 45]$, representation dimension: {16, 128, 256, 512}, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3.

- **ERM (prob)**: learning rate: $[10^{-5}, 10^{-3.5}]$, weight decay: $[10^{-6}, 10^{-2}]$, dropout rate: {0.0, 0.1, 0.5}, batch size: $[8, 45]$, representation dimension: {16, 128, 256, 512}, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3.

- **DANN** [Ganin et al., 2016]: learning rate: $[10^{-5}, 10^{-3.5}]$, weight decay: $[10^{-6}, 10^{-2}]$, dropout rate: {0.0, 0.1, 0.5}, batch size: $[8, 45]$, representation dimension: {16, 128, 256, 512}, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, adversarial loss coefficient: $[10^{-2}, 10^{2}]$, weight decay of discriminator: $[10^{-6}, 10^{-2}]$, number of discriminator steps per generator steps: $\{1, 2, 4, 8\}$, grad penalty coefficient: $[10^{-2}, 10^{1}]$

- **MMD** [Li et al., 2018c]: learning rate: $[10^{-5}, 10^{-3.5}]$, weight decay: $[10^{-6}, 10^{-2}]$, dropout rate: {0.0, 0.1, 0.5}, batch size: $[8, 45]$, representation dimension: {16, 128, 256, 512}, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, MMD coefficient: $[10^{-3}, 10^{-1}]$

- **CORAL** [Sun and Saenko, 2016]: learning rate: $[10^{-5}, 10^{-3.5}]$, weight decay: $[10^{-6}, 10^{-2}]$, dropout rate: {0.0, 0.1, 0.5}, batch size: $[8, 45]$, representation dimension: {16, 128, 256, 512}, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, CORAL loss coefficient: $[10^{-3}, 10^{-1}]$

- **WD** [**Shen et al., 2018**]: learning rate: $[10^{-5}, 10^{-3.5}]$, weight decay: $[10^{-6}, 10^{-2}]$, dropout rate: $\{0.0, 0.1, 0.5\}$, batch size: $[8, 45]$, representation dimension: $\{16, 128, 256, 512\}$, number of layers of the classifier ($\hat{p}(y|z)$): 1 or 3, wasserstein distance coefficient: $[10^{-2}, 10^{2}]$, weight decay of network f: $[10^{-6}, 10^{-2}]$, number of f optimization steps per normal optimization steps: $\{1, 2, 4, 8\}$, grad penalty coefficient: $[10^{-2}, 10^{1}]$

- **KL (ours)**: learning rate: 10^{-4}, weight decay: $[10^{-6}, 10^{-2}]$, dropout rate: 0.0, batch size: 256, representation dimension: 16, number of layers of the classifier ($\hat{p}(y|z)$): 1, $\beta : \{0.1, 0.05, 0.001\}$, $\beta_{aux} : \{0.1, 0.05, 0.01, 0.0\}$

Appendix C

FedSR: A Simple and Effective Domain Generalization Method for Federated Learning

C.1 Proofs

C.1.1 Lemma 4.1

Proof. Recall that the joint distribution of x, y, z for client i is:

$$p_i(x, y, z) = p_i(x, y)p(z|x) = p_i(y)p_i(x|y)p(z|x).$$

We have:

$$\begin{aligned}
I_i(x, z|y) &= \mathbb{E}_{p_i(x,y,z)}\left[\log \frac{p_i(x, z|y)}{p_i(x|y)p_i(z|y)}\right] \\
&= \mathbb{E}_{p_i(x,y,z)}\left[\log \frac{p_i(x|y)p(z|x)}{p_i(x|y)p_i(z|y)}\right] \\
&= \mathbb{E}_{p_i(x,y,z)}\left[\log \frac{p(z|x)}{p_i(z|y)}\right] \\
&= \mathbb{E}_{p_i(x,y,z)}\left[\log \frac{p(z|x)}{p_i(z|y)}\right] \\
&= \mathbb{E}_{p_i(x,y,z)}\left[\log p(z|x) - \log p_i(z|y)\right].
\end{aligned}$$

Notice that:

$$\begin{aligned}
&\mathbb{E}_{p_i(y)}\left[\mathrm{KL}[p_i(z|y)|r(z|y)]\right] \geq 0 \\
\Rightarrow &\mathbb{E}_{p_i(y)}\left[\mathbb{E}_{p_i(z|y)}[\log p_i(z|y) - \log r(z|y)]\right] \geq 0 \\
\Rightarrow &\mathbb{E}_{p_i(y,z)}\left[\log p_i(z|y) - \log r(z|y)\right] \geq 0 \\
\Rightarrow &\mathbb{E}_{p_i(x,y,z)}\left[\log p_i(z|y) - \log r(z|y)\right] \geq 0 \\
\Rightarrow &\mathbb{E}_{p_i(x,y,z)}\left[\log p_i(z|y)\right] \geq \mathbb{E}_{p_i(x,y,z)}\left[\log r(z|y)\right].
\end{aligned}$$

Therefore:

$$\begin{aligned}
I_i(x, z|y) &= \mathbb{E}_{p_i(x,y,z)}\left[\log p(z|x) - \log p_i(z|y)\right] \\
&\leq \mathbb{E}_{p_i(x,y,z)}\left[\log p(z|x) - \log r(z|y)\right] \\
&= \mathbb{E}_{p_i(x,y)}\left[\mathbb{E}_{p(z|x)}[\log p(z|x) - \log r(z|y)]\right] \\
&= \mathbb{E}_{p_i(x,y)}\left[\mathrm{KL}[p(z|x)|r(z|y)]\right].
\end{aligned}$$

□

C.1.2 Lemma 4.2

Proof. We need to prove:

$$\begin{aligned}
&\mathbb{E}_{p_i(x,y)}\left[\mathrm{KL}[p(z|x)|r(z|y)]\right] \geq \mathbb{E}_{p_i(y)}\left[\mathrm{KL}[p_i(z|y)|r(z|y)]\right] \\
\Leftrightarrow &\mathbb{E}_{p_i(x,y,z)}[\log p(z|x) - \log r(z|y)] \geq \mathbb{E}_{p_i(y,z)}[\log p_i(z|y) - \log r(z|y)] \\
\Leftrightarrow &\mathbb{E}_{p_i(x,y,z)}[\log p(z|x)] \geq \mathbb{E}_{p_i(y,z)}[\log p_i(z|y)] \\
&\quad (\text{since } \mathbb{E}_{p_i(x,y,z)}[\log r(z|y)] = \mathbb{E}_{p_i(y,z)}[\log r(z|y)]) \\
\Leftrightarrow &\mathbb{E}_{p_i(y)}\left[\mathbb{E}_{p_i(x,z|y)}[\log p(z|x)]\right] \geq \mathbb{E}_{p_i(y)}\left[\mathbb{E}_{p_i(z|y)}[\log p_i(z|y)]\right].
\end{aligned}$$

Therefore, we only need to prove that $\mathbb{E}_{p_i(x,z|y)}[\log p(z|x)] \geq \mathbb{E}_{p_i(z|y)}[\log p_i(z|y)]\ \forall y$.

This is equivalent to:

$$\begin{aligned}
&\int_z \int_x p_i(x, z|y) \log p(z|x) dx dz \geq \int_z p_i(z|y) \log p_i(z|y) dz \\
\Leftrightarrow &\int_z \int_x p_i(x|y)[p(z|x) \log p(z|x)] dx dz \geq \int_z p_i(z|y) \log p_i(z|y) dz \\
\Leftrightarrow &\int_z \mathbb{E}_{p_i(x|y)}[p(z|x) \log p(z|x)] dz \geq \int_z p_i(z|y) \log p_i(z|y) dz. \qquad \text{(C.1)}
\end{aligned}$$

Notice that the function $f(a) = a \log a, a > 0$ is a convex function since $f''(a) = \frac{1}{a} > 0 \; \forall a > 0$. Hence, due to Jensen's Inequality, we have:

$$\mathbb{E}_{p_i(x|y)}[p(z|x) \log p(z|x)] \geq b \log b,$$

where $b = \mathbb{E}_{p_i(x|y)}[p(z|x)] = p_i(z|y)$.

Therefore, $\mathbb{E}_{p_i(x|y)}[p(z|x) \log p(z|x)] \geq p_i(z|y) \log p_i(z|y)$. Then, inequality C.1 holds, and we conclude our proof. □

C.2 Details on the ℓ_i^{CMI} objective

As mentioned in the main text, we use a Gaussian distribution for both $p(z|x)$ and $r(z|y)$, so that the KL term can be computed analytically.

Let w_1 be the parameter of the (probabilistic) representation network. Given an input x_0 (of a datapoint (x_0, y_0)), the network output the mean and standard deviation of the Gaussian distribution $p(z|x_0)$ (both from the final layer of the network), denoted $\mu_{w_1}(x_0)$ and $\sigma_{w_1}(x_0)$. This means that $p(z|x_0) = \mathcal{N}(z; \mu_{w_1}(x_0), \sigma^2_{w_1}(x_0))$.

Also as pointed out earlier, for a classification problem (where there is a finite number of class labels), we set $r(z|y) = \mathcal{N}(z; \mu_y, \sigma_y^2)$, where μ_y, σ_y $(y = \overline{1..C})$ are the variational parameters to be optimized. Therefore, for the datapoint x_0, y_0, we have $r(z|y_0) = \mathcal{N}(z; \mu_{y_0}, \sigma^2_{y_0})$.

Note that when z is a K-dimensional vector, $\mu_{w_1}(x_0), \sigma_{w_1}(x_0), \mu_{y_0}$ and σ_{y_0} are all K-dimensional.

The KL term in ℓ^{CMI} can be computed analytically as:

$$\log \sigma_{y_0} - \log \sigma_{w_1}(y_0) + \frac{\sigma^2_{w_1}(x_0) + (\mu_{w_1}(x_0) - \mu_{y_0})^2}{2\sigma^2_{y_0}} - \frac{1}{2}. \tag{C.2}$$

If $K > 1$ (i.e., z is high dimensional), the calculations in Eq. C.2 are element-wise, and the result is summed across the dimension K at the end.

As this computation is deterministic with respect to w_1 and μ_y, σ_y $(y \in \overline{1..C})$, the gradient w.r.t. these parameters can be computed straightforwardly.

C.3 Experiments

C.3.1 Hyper-parameters:

As mentioned in the main text, we use random search to tune the hyper-parameters of our method, namely α^{CMI} and α^{L2R}. Specifically, the tuned value for those parameters are:

RotatedMNIST

- FedL2R: $\alpha^{L2R} = 0.1$
- FedCMI: $\alpha^{CMI} = 0.3$
- FedSR: $\alpha^{L2R} = 0.1, \alpha^{CMI} = 0.3$

PACS

- FedL2R: $\alpha^{L2R} = 0.01$
- FedCMI: $\alpha^{CMI} = 0.01$
- FedSR: $\alpha^{L2R} = 0.01, \alpha^{CMI} = 0.001$

OfficeHome

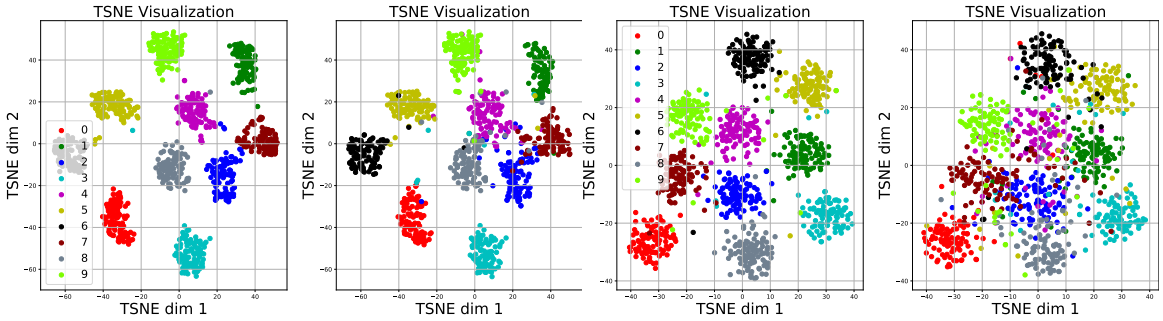
- FedL2R: $\alpha^{L2R} = 0.05$
- FedCMI: $\alpha^{CMI} = 0.0005$
- FedSR: $\alpha^{L2R} = 0.05, \alpha^{CMI} = 0.0005$

DomainNet

- FedL2R: $\alpha^{L2R} = 0.01$
- FedCMI: $\alpha^{CMI} = 0.005$
- FedSR: $\alpha^{L2R} = 0.01, \alpha^{CMI} = 0.0005$

C.3.2 Visualization of the Representation space

We also visualize the representation space of our method and observe that it aligns the representation much better than the conventional FL baseline FedAVG (which does not actively attempt to align the representation distributions). Specifically, Figure C.1 shows the t-SNE [Van der Maaten and Hinton, 2008] visualization of the representation of FedSR and FedAVG in the RotatedMNIST experiment, with $\mathcal{M}_0$ (the target domain) and $\mathcal{M}_{15}$ (one of the source domains). We can clearly see that both the marginal and the conditional distributions of the representation are aligned better between the two domains for FedSR.



(a) FedSR (b) FedAVG

Figure C.1: Visualization using t-SNE of the representation space of our method FedSR and the baselines FedAVG. For each method, the left subfigure corresponds to one source domain $\mathcal{M}_{15}$ and the right one corresponds to the target domain $\mathcal{M}_0$. Each color represents a digit class.

Appendix D

TIPI: Test Time Adaptation with Transformation Invariance

D.1 Derivations and Proofs

D.1.1 Assumption 5.1

First of all, the formulas for the two mutual information terms are:

$$I_T(x,y) = \int_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_T(x,y) \log \frac{p_T(x,y)}{p_T(x)p_T(y)},$$
$$I_T(x^t,y) = \int_{x^t\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_T(x^t,y) \log \frac{p_T(x^t,y)}{p_T(x^t)p_T(y)}.$$

Recall that Assumption 1 states:

$$\begin{aligned}
&I_T(x,y) = I_T(x^{tr},y)\\
\Leftrightarrow &\mathbb{E}_{p_T(x,y)}\left[\log \frac{p_T(x,y)}{p_T(x)p_T(y)}\right] = \mathbb{E}_{p_T(x^{tr},y)}\left[\log \frac{p_T(x^{tr},y)}{p_T(x^{tr})p_T(y)}\right]\\
\Leftrightarrow &\mathbb{E}_{p_T(x,y)}\left[\log \frac{p_T(y|x)}{p_T(y)}\right] = \mathbb{E}_{p_T(x^{tr},y)}\left[\log \frac{p_T(y|x^{tr})}{p_T(y)}\right] = \mathbb{E}_{p_T(x^{tr},y)}\left[\log \frac{p_T(x^{tr},y)}{p_T(x^{tr})p_T(y)}\right]\\
\Leftrightarrow &\mathbb{E}_{p_T(x,x^{tr},y)}\left[\log \frac{p_T(y|x)}{p_T(y)}\right] = \mathbb{E}_{p_T(x,x^{tr},y)}\left[\log \frac{p_T(y|x^{tr})}{p_T(y)}\right]\\
\Leftrightarrow &\mathbb{E}_{p_T(x,x^{tr},y)}\left[\log p_T(y|x) - \log p_T(y|x^{tr})\right] = 0\\
\Leftrightarrow &\mathbb{E}_{p_T(x,x^{tr})}\left[\mathbb{E}_{p_T(y|x)}\left[\log p_T(y|x) - \log p_T(y|x^{tr})\right]\right] = 0\\
\Leftrightarrow &\mathbb{E}_{p_T(x,x^{tr})}\left[\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]\right] = 0.
\end{aligned}$$

D.1.2 Proposition 5.1

Note that our result also holds if l is other distance/divergence (such as the Jensen Shannon divergence), only with a modified coefficient of the regularizer term. However, in this work, we consider l as the l_1 distance.

Proof. First of all, due to Pinsker's inequality, we have that:

$$\begin{aligned}
& l[p_T(y|x), p_T(y|x^{tr})] \leq \sqrt{2\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]} \\
\Rightarrow & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_T(y|x^{tr})]\right] \\
& \leq \mathbb{E}_{p_T(x,x^{tr})}\left[\sqrt{2\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]}\right] \\
& \leq \sqrt{\mathbb{E}_{p_T(x,x^{tr})}\left[2\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]\right]} \\
& = 0 \\
\Rightarrow & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_T(y|x^{tr})]\right] = 0.
\end{aligned}$$

We have:

$$\begin{aligned}
& \ell(\theta, p_T(x,y)) - \ell(\theta, p_T(x^{tr}, y)) \\
= & \mathbb{E}_{p_T(x)}\left[l[p_T(y|x), p_\theta(y|x)]\right] - \mathbb{E}_{p_T(x^{tr})}\left[l[p_T(y|x^{tr}), p_\theta(y|x^{tr})]\right] \\
= & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_\theta(y|x)]\right] - \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x^{tr}), p_\theta(y|x^{tr})]\right] \\
& - \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_T(y|x^{tr})]\right] \\
\leq & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_\theta(y|x^{tr}), p_\theta(y|x)]\right].
\end{aligned}$$

This is because l satisfies the triangle inequality, so:

$$\begin{aligned}
& l[p_T(y|x), p_\theta(y|x)] - l[p_T(y|x^{tr}), p_\theta(y|x^{tr})] - l[p_T(y|x), p_T(y|x^{tr})] \\
& \leq l[p_\theta(y|x^{tr}), p_\theta(y|x)].
\end{aligned}$$

Using Pinsker's inequality again, we have that:

$$\begin{aligned}
& \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_\theta(y|x^{tr}), p_\theta(y|x)]\right] \\
\leq & \mathbb{E}_{p_T(x,x^{tr})}\left[\sqrt{2\mathrm{KL}[p_\theta(y|x^{tr}), p_\theta(y|x)]}\right] \\
\leq & \mathbb{E}_{p_T(x)}\left[\sqrt{2 \max_{x^{tr}\sim T(x^{tr}|x)} \mathrm{KL}[p_\theta(y|x^{tr}), p_\theta(y|x)]}\right],
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{E}_{p_T(x,x^{tr})}\left[l[p_\theta(y|x^{tr}), p_\theta(y|x)]\right] \\
\leq &\mathbb{E}_{p_T(x,x^{tr})}\left[\sqrt{2\mathrm{KL}[p_\theta(y|x), p_\theta(y|x^{tr})]}\right] \\
\leq &\mathbb{E}_{p_T(x)}\left[\sqrt{2 \max_{x^{tr}\sim T(x^{tr}|x)} \mathrm{KL}[p_\theta(y|x), p_\theta(y|x^{tr})]}\right].
\end{aligned}$$

This concludes our proof.

□

D.2 Additional Experiments

D.2.1 Incorporating TIPI into EATA [Niu et al., 2022]

Niu et al. [2022] propose EATA, a data points selection strategy for TENT, which achieves state-of-the-art performance on the test time adaptation task. As discussed before, this line of research (data point selection for surrogate optimization) is complementary to ours. Indeed, in this subsection, we show that TIPI can be incorporated into EATA, thereby improving EATA's robustness (EATA only uses TENT and is not robust against small batchsizes). We refer the readers to [Niu et al., 2022] for a detailed discussion on their data point selection strategy.

We conduct the experiment with ETA, the variation without weight regularization (ETA outperforms EATA for out-of-distribution data). Specifically, we compare ETA to ETA+TIPI, a variant of ETA with the same data point selection strategy, but with the TIPI surrogate objective. Following Niu et al. [2022], we conduct the experiment on ImageNet-C with the highest level of severity. The 15 types of corruption are: gaussian noise, shot noise, impulse noise, defocus blur, glass blur, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic transform, pixelate, and jpeg compression.

Table D.1 and Table D.2 show the results of this comparison for batch sizes 64 and 2. For a large batch size, ETA+TIPI further improves over ETA, pushing a new SOTA result. For a small batch size, incorporating the TIPI objective help to stabilize the optimization, thus avoiding network collapse.

Furthermore, note that we keep the data point selection strategy in EATA as is in this experiment. This strategy is developed specifically for the TENT objective and we did not make any modifications for TIPI. Yet, using it for TIPI already

Table D.1: **ImageNet-C, Batchsize 64**: Results for 15 types of corruption with the highest level of severity

Method	Classification Error (lower is better)															
	Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	Average
Source	97.0	96.3	97.4	82.1	90.3	85.3	77.5	83.4	76.9	76	40.9	94.6	83.5	79.1	67.4	81.8
ETA	64.9	62.1	63.4	66.1	67.1	52.2	47.4	48.1	54.2	39.9	32.1	55.0	42.1	39.1	45.1	51.9
ETA+TIPI	**63.0**	**60.9**	**62.1**	**65.2**	**65.4**	**51.6**	**46.7**	**47.6**	**53.7**	**39.9**	**31.6**	**54.1**	**41.2**	**38.2**	**43.6**	**51.0**

Table D.2: **ImageNet-C, Batchsize 2**: Results for 15 types of corruption with the highest level of severity

Method	Classification Error (lower is better)															
	Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	Average
Source	97.0	96.3	97.4	**82.1**	90.3	85.3	77.5	83.4	76.9	76	40.9	94.6	83.5	79.1	67.4	81.8
ETA	97.7	97.4	97.5	97.8	97.9	95.1	93.3	92.2	92.3	89.2	78.4	96.4	90.2	90.1	92.3	93.2
ETA+TIPI	**87.8**	**88.8**	**87.4**	88.3	**89.3**	**82.4**	**70.5**	**73.2**	**75.8**	**60.7**	**40.8**	**93.9**	**64.0**	**58.8**	**65.4**	**75.4**

improves the model's performance. This shows that indeed the two lines of research are complementary. It also suggests that investigating a data point selection strategy tailored for TIPI is a promising direction, and could potentially improve TIPI's performance even further.

Appendix E

uCAP: An Unsupervised Prompting Method for Vision-Language Models

E.1 Gradient of $\log Z_Y(w)$

$$
\begin{aligned}
\nabla_w \log Z_Y(w) &= \frac{\nabla_w Z_Y(w)}{Z_Y(w)} = \frac{\int \nabla_w e^{-E(x,Y,w)} dx}{Z_Y(w)} \\
&= \frac{\int e^{-E(x,Y,w)} \nabla_w \log e^{-E(x,Y,w)} dx}{Z_Y(w)} \\
&= \int \frac{e^{-E(x,Y,w)}}{Z_Y(w)} \nabla_w \log e^{-E(x,Y,w)} dx \\
&= -\int p(x|Y,w) \nabla_w E(x,Y,w) dx \\
&= -\mathbb{E}_{p(x|Y,w)} \left[\nabla_w E(x,Y,w) \right] \\
&= -\frac{1}{M} \sum_{i=1}^{M} \nabla_w E(\tilde{x}_i, Y, w),
\end{aligned}
$$

where $\tilde{x}_i \sim p(x|Y,w), i \in \{1,2,...,M\}$, and the last equation is obtained from Monte Carlo sampling.

E.2 Classification via Descriptions

To show that our method is truly versatile, in this section, we combine uCAP with *Classification via Descriptions* [Menon and Vondrick, 2023] (which use class descriptions generated by a large language model instead of the default class names). We use the same class descriptions as released in their official github page [1], and additionally

[1] https://github.com/sachit-menon/classify_by_description_release

Table E.1: Comparison with "Classification via Descriptions".

Method	ViT-B/16			ViT-L/14@336px		
	ImageNet	OxfordPets	CUB	ImageNet	OxfordPets	CUB
Classification via Description	67.9	86.9	58.1	76.2	91.7	64.9
uCAP + Descriptions	**69.6**	**90.1**	**58.3**	**77.6**	**94.7**	**65.2**

Table E.2: Results for UCF101 action recognition dataset with CLIP.

Method	RN50	RN101	ViT-B/32	ViT-B/16	ViT-L/14	ViT-L/14@336
Classnames Only	61.79	63.68	66.28	68.26	74.23	73.82
Classnames + HTP	68.39	67.52	69.18	73.83	80.44	79.75
uCAP	**68.64**	**70.50**	**70.71**	**74.46**	**80.77**	**80.74**

learn domain-level prompts with uCAP. The results of this experiment is presented in Table E.1, which shows that uCAP can further improve the performance of class descriptions generated by Large Language Models.

E.3 Video Experiment

We also perform an experiment on the UCF101 video dataset [Soomro et al., 2012]. This is an action recognition dataset which consists of 101 actions. For each video, we sample 60 frames randomly, and average the frames' representation to get the video representation. For the hand-tuned prompts baseline, we use the same prompts as proposed in the CLIP paper [Radford et al., 2021]. As illustrated in Table E.2, our method uCAP can be straightforwardly extended to this modality and the learned prompts lead to significant improvement in accuracy.

Appendix F

Publications and Contributions List

F.1 Domain Invariant Representation Learning with Domain Density Transformations

Publication Status: Published at NeurIPS 2021

Publication Details: A. Tuan Nguyen, Toan Tran, Yarin Gal, and Atılım Güneş Baydin. Domain Invariant Representation Learning with Domain Density Transformations. *NeurIPS*, 2021.

Contributions:

- A. Tuan Nguyen is the sole first author. Tuan came up with the idea and theory, designed and executed the experiments, and led the writing effort.
- Yarin Gal and Atılım Güneş Baydin supervised the project and helped revise the paper.
- Toan Tran provided some discussion on the experiments.

F.2 KL Guided Domain Adaptation

Publication Status: Published at ICLR 2022

Publication Details: A. Tuan Nguyen, Toan Tran, Yarin Gal, Philip H. S. Torr, and Atılım Güneş Baydin. KL Guided Domain Adaptation. *ICLR*, 2022.

Contributions:

- A. Tuan Nguyen is the sole first author. Tuan came up with the idea and theory, designed and executed the experiments, and led the writing effort.

- Yarin Gal, Philip Torr, and Atılım Güneş Baydin supervised the project and helped revise the paper.
- Toan Tran provided some discussion on the experiments.

F.3 FedSR: A Simple and Effective Domain Generalization Method for Federated Learning

Publication Status: Published at NeurIPS 2022

Publication Details: A. Tuan Nguyen, Philip Torr, and Ser-Nam Lim. FedSR: A Simple and Effective Domain Generalization Method for Federated Learning. *NeurIPS*, 2022.

Contributions:

- A. Tuan Nguyen is the sole first author. Tuan came up with the idea of using simple representation for improved generalization, designed and executed the experiments, and led the writing effort.
- Philip Torr and Ser-Nam Lim supervised the project and helped revise the paper.

F.4 TIPI: Test Time Adaptation with Transformation Invariance

Publication Status: Published at CVPR 2023

Publication Details: A. Tuan Nguyen, Thanh Nguyen-Tang, Ser-Nam Lim, and Philip Torr. TIPI: Test Time Adaptation with Transformation Invariance. *CVPR*, 2023.

Contributions:

- A. Tuan Nguyen is the sole first author. Tuan came up with the method, designed and executed the experiments, and led the writing effort.
- Ser-Nam Lim and Philip Torr supervised the project and helped revise the paper.
- Thanh Nguyen-Tang provided some discussion about the theory.

F.5 uCAP: An Unsupervised Prompting Method for Vision-Language Models

Note: This work was done partially while Tuan was a research intern at Meta AI.

Publication Status: Submitted for Publication

Publication Details: A. Tuan Nguyen, Kai Sheng Tai, Bor-Chun Chen, Satya Narayan Shukla, Hanchao Yu, Philip Torr, Taipeng Tian, and Ser-Nam Lim. uCAP: An Unsupervised Prompting Method for Vision-Language Models. *Under Review*, 2024.

Contributions:

- A. Tuan Nguyen is the sole first author. Tuan came up with the method, designed and executed the experiments, and led the writing effort.
- Philip Torr, Taipeng Tian, and Ser-Nam Lim supervised the project and helped revise the paper.
- Kai Sheng Tai helped revise the approach section.
- Other authors provided discussion in various sections and various stages of the project.