ABSTRACT

Title of Dissertation: ROBUST LEARNING UNDER DISTRIBUTIONAL SHIFTS

Yogesh Balaji
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

Professor Soheil Feizi
Department of Computer Science

Designing robust models is critical for reliable deployment of artificial intelligence systems. Deep neural networks perform exceptionally well on test samples that are drawn from the same distribution as the training set. However, they perform poorly when there is a mismatch between training and test conditions, a phenomenon called *distributional shift*. For instance, the perception system of a self-driving car can produce erratic predictions when it encounters a new test sample with a different illumination or weather condition not seen during training. Such inconsistencies are undesirable, and can potentially create life-threatening conditions as these models are deployed in safety-critical applications.

In this dissertation, we develop several techniques for effectively handling distributional shifts in deep learning systems.

In the first part of the dissertation, we focus on detecting out-of-distribution

shifts that can be used for flagging outlier samples at test-time. We develop a likelihood estimation framework based on deep generative models for this task. In the second part, we study the domain adaptation problem where the objective is to tune the neural network models to adapt to a specific target distribution of interest. We design novel adaptation algorithms, understand and analyze them under various settings. In the last part of the dissertation, we develop robust learning algorithms that can generalize to novel distributional shifts. In particular, we focus on two types of shifts - *covariate* and *adversarial* shifts. All developed algorithms are rigorously evaluated on several benchmark datasets.

ROBUST LEARNING UNDER DISTRIBUTIONAL SHIFTS

by

Yogesh Balaji

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Soheil Feizi, Co-Advisor
Professor Abhinav Shrivastava
Professor Tom Goldstein
Professor Wojciech Czaja (Dean's Representative)

To my mother

Acknowledgments

First and foremost, I would like to thank my advisors Rama Chellappa and Soheil Feizi. This dissertation wouldn't be possible without your valuable guidance and support.

It was my honor to work with a wonderful mentor and a great researcher like Rama Chellappa. Since my first meeting with him as a nervous graduate student, I have learnt countless life lessons from him over the last five years. Throughout my PhD, he has provided me freedom to pursue ideas that interest me, and this made my entire graduate school experience most memorable. He was kind, and has always been a constant source of motivation and support even during some of my hardest times, for which I would be forever grateful.

Since my third year, I was fortunate to work with Soheil Feizi. Soheil is an extremely prolific researcher with whom I have had some of the most intellectual discussions. His passion for research and teaching have always inspired me. He has spend countless hours helping me with research, talks and career, for which I am indebted forever.

Thank you to my committee members Abhinav Shrivastava, Tom Goldstein and Wojciech Czaja. I have enjoyed my interactions with you during my time here at UMD.

I would like to thank all my amazing collaborators with whom I have had the opportunity to work with over the last five years. When interning at Facebook, I was fortunate to be mentored by Judy Hoffman and Tom Goldstein. Judy was an excellent researcher who helped me a lot with research and career. I learnt a great deal from her on how to approach a problem, how to stay organized, and improve my presentation skills. It was a pleasure working with her. I am very thankful to her for having continued collaborations even after our time at FAIR. And thanks Facebook for all the free food! I would like to thank Deepmind Mountain View team for hosting me as an intern in the summer of 2020. Despite being a remote internship, it was really fun working there. Thank you Ang, Mehrdad, Dong, Dilan, Alex and Nevena. You guys were always there to help me. And thank you for the numerous TPUs I got to play with!

At UMD, I got to work with some amazing fellow PhD students. Thank you Swami for helping me a lot in the initial stages of my PhD. A special thanks to all students I got to co-author with: Luyu, Neha, Andy, Phil, Gowthami, Priyatham and Mazda. It was a pleasure working with you all. I was also fortunate to work with PhD students from other universitites: Prithvijit, Bingyuan and Mohammadmahdi. You guys were amazing. And thanks to students in Rama's lab with whom I have had countless discussions about research and life. Thank you Anirudh, Ilya, Ankan, Steve and Rajeev. I would also like to thank UMIACS staff for providing valuable tech support over the years.

I would like to thank all my friends who helped me through the struggles of graduate school. In particular, I thank my childhood friends Manoj, Vicky, Sathiya,

Sandeep and Asif. You guys stayed with me for the last 20 years, and no matter what happens in my life, I know I can always count on you. And to all my amazing friends I made during graduate school: Dacha, Vaishnavi, Luyu, Hari, Shravan, Ajay and Susmija. You guys were my pillars of support and got me through some tough times. Thank you all.

Finally, I want to thank my family: my father, my brother Aswin, my uncle and my grandparents who provided me with love and support throughout my graduate school. And to my mother, the one person who loved me unconditionally, who always had belief in me, who sacrificed so much for me, who provided me with endless support, encouragement and love till her last moments. There are no words to express my gratitude to you. None of this would have been possible without you, mother.

This dissertation was supported by in part by a MURI program from the Army Research Office under the grant W911NF17-1-0304, NSF CAREER AWARD 1942230, HR00112090132, HR001119S0026, HR00112090132, NIST 60NANB20D134, a grant from Capital One, AWS Machine Learning Research Award and Simons Fellowship on "Foundations of Deep Learning.". I thank all these agencies for their gracious support.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Deep neural networks have revolutionized the field of machine learning in the last decade, achieving impressive performance in several tasks including visual recognition[5, 6, 7], speech processing [8], natural language understanding [9, 10, 11], reinforcement learning [12, 13, 14, 15] and robotics [16, 17, 18]. This success is in part fuelled by the availability of large datasets and powerful computational resources in the form of modern GPU hardware. In the last decade, significant progress has been made in designing and deploying efficient deep learning systems in several research, industrial and consumer applications.

Despite this success, one of the key limitations of deep neural networks is a lack of robustness to distributional shifts. By distributional shifts, we mean inputs at test time having different distributional statistics than those in the training datasets. Deep networks are designed to perform well on samples that are drawn from the same distribution as the training set. This assumption seldom holds true in practice as test inputs often include variations not contained in the training datasets. These variations can be due to differences in the environments in which the models are deployed, differences in image acquisition, changes in illumination and lighting

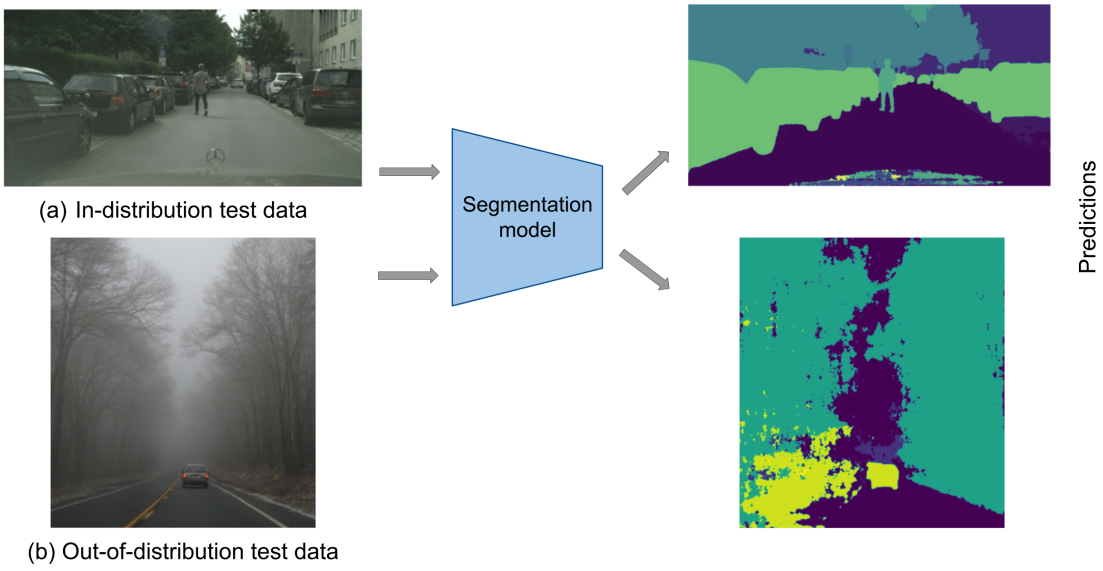


Figure 1.1: Performance of a semantic segmentation model trained on Cityscapes dataset [19] on (a) *in-distribution* test image taken from the Cityscapes test set, and (b) *out-of-distribution* test image of foggy street scene taken from Sakaridis et al. [20]. While smooth and consistent segmentation maps are obtained for *in-distribution* test images, inferior predictions are obtained for *out-of-distribution* inputs. Each color in prediction maps denote a semantic label.

conditions, etc. While humans are extremely good at reliable decision making under such distributional shifts, performance of deep networks can drop drastically. This lack of robustness is undesirable and can even be fatal in several safety-critical applications such as autonomous driving and medical diagnosis.

To illustrate the issue of distributional shift, consider the task of semantic segmentation on street scenes taken from an autonomous driving dataset (Cityscapes [19]). The objective of semantic segmentation is to assign semantic labels to every pixel in an image. In the panel (a) of Fig. 1.1, we observe that smooth and meaningful segmentation maps are obtained on the in-distribution test image i.e., image taken

from the same dataset in which the model was trained on. In panel (b), we show the prediction map obtained by the same model when tested on an out-of-distribution image of a foggy street scene. Such images are not present in the training dataset. We observe extremely noisy segmentation outputs. If these models are deployed in the perception system of a real autonomous vehicle, the unreliable predictions these models produce can result in life-threatening consequences. Hence, there is a pressing need to design machine learning systems that are robust to such distributional shifts. This is the focus of this dissertation.

1.2 Contributions

In this dissertation, we develop several techniques for handling out-of-distribution inputs in deep learning systems. In this section, we provide a summary of these approaches.

Likelihood Estimation: In the first part of the dissertation, we focus on detecting out-of-distribution samples using likelihood estimation. The objective is to build a probability model of data, which can then be used for flagging out-of-distribution samples as outliers. We utilize deep generative models (Generative Adversarial Networks, in particular) for this task. Once the probability model is built, we can utilize the trained model to compute sample likelihoods at test-time. Samples which are least likely to have been generated from the model can be flagged as outliers. We develop a theory of likelihood estimation for Generative Adversarial Networks (GANs), and show how out-of-distribution detection can be performed using GANs

at test time.

Domain Adaptation: In the second part of the dissertation, we develop adaptation algorithms with the goal of improving performance on out-of-distribution datasets (also called target distributions). To do this, we utilize unlabeled samples from the target distributions. Models are then trained using a combination of labeled training data and unlabeled target data so that the performance on the target distribution improves. This class of techniques is also called *unsupervised domain adaptation*. In this dissertation, we develop novel domain adaptation algorithms involving hybrid generative-discriminative approaches and variants of optimal transport distances. The developed techniques are extensively evaluated on several benchmark domain adaptation datasets.

Domain Generalization: In the third part of the dissertation, we focus on the problem of domain generalization. Unlike domain adaptation, in domain generalization, we do not assume access to any target data during training. The goal is to train models using several training data distributions so that the models can generalize to novel test distributions. In this dissertation, we develop regularized training mechanisms for deep networks that can generalize to out-of-distribution shifts. In particular, we learn a regularization function using meta-learning, which can then used for regularizing the model training. The regularized models are shown be more robust to out-of-distribution shifts.

Adversarial Robustness: In the final part of the dissertation, we shift our focus to a different type of distributional shift, called the *adversarial shifts*. Adversarial shifts are noisy inputs which are created by a malicious adversary with the intent of breaking the machine learning systems. We develop novel training techniques for improving the robustness of deep networks to these adversarial attacks, while preserving the generalization on clean unperturbed inputs.

1.3 Organization

In Chapter 2, we introduce some background concepts of object recognition, generative modeling and optimal transport which will be used in the rest of the dissertation. Chapter 3 presents likelihood estimation using GANs, and how they can used for out-of-distribution detection. In Chapter 4, we discuss how unsupervised domain adaptation can be performed using generative adversarial networks. Novel variants of optimal transport for domain adaptation are discussed in chapters 5 and 6. In Chapter 7, we discuss the meta-learning framework for domain generalization problem. In Chapter 8, we develop an algorithm for robust training under adversarial shifts, called instance adaptive adversarial training, and study the robustness-accuracy tradeoffs. Finally, in Chapter 9, we conclude the dissertation and suggest some future directions for research.

Chapter 2: Background

2.1 Object Recognition

In this dissertation, we primarily focus on the classification task. In image classification, we take as input an image represented as a grid of pixels. The task is to classify the input image into one of several predefined object categories. An example is shown in Figure 2.1 - Given these inputs, an object classification system is required to recognize the images as Robin, mud turtle and persian cat, respectively.

Formally, let $\mathcal{X} = \mathbb{R}^{c \times h \times w}$ denote the input space. Here, c, h and w are the number of channels, height and width of the image respectively. Let $\mathcal{Y} = \{1, 2, \ldots n_c\}$ denote the discrete label space for a classification task with n_c labels. The objective of a learning system is to learn a model $F : \mathcal{X} \rightarrow \mathcal{Y}$ that assigns correct predictions to a test image.

2.1.1 Deep Learning

Classical approaches to solve the object recognition problem comprised of a two stage process. The first stage involves extracting the feature representations of inputs, which are consise representations of the images that capture maximum

Figure 2.1: Image classification task. Given the images shown above as inputs, an object recognition system should classify the images as *Robin*, *mud turtle* and *persian cat*, respectively. Figure from Imagenet (Deng et al. [21])

information needed for a given task. In the second stage, the extracted feature representations are passed to a machine learning model, which is trained to make good test-time predictions. Some popular methods for feature extraction include Scale Invariant Feature Transform (SIFT) [22], Histogram of Oriented Gradients (HOG) [23], Speeded up Robust Features (SURF) [24], etc. These hand-crafted features relied on low-level information such as corners, edges and gradients of images.

Starting 2012, the focus shifted towards end-to-end learning, in which both the feature extraction and classification steps are embedded into a single model. Deep neural networks have emerged as the ideal choice for such joint models. While neural networks were introduced several decades ago [25, 26, 27, 28], it was the work of Krizhevsky et al. [29] that led to the resurgence of deep networks for large scale machine learning applications. The availability of large datasets and modern GPU hardware helped train high capacity deep neural network models, which eventually resulted in state-of-the art performance on several machine learning tasks.

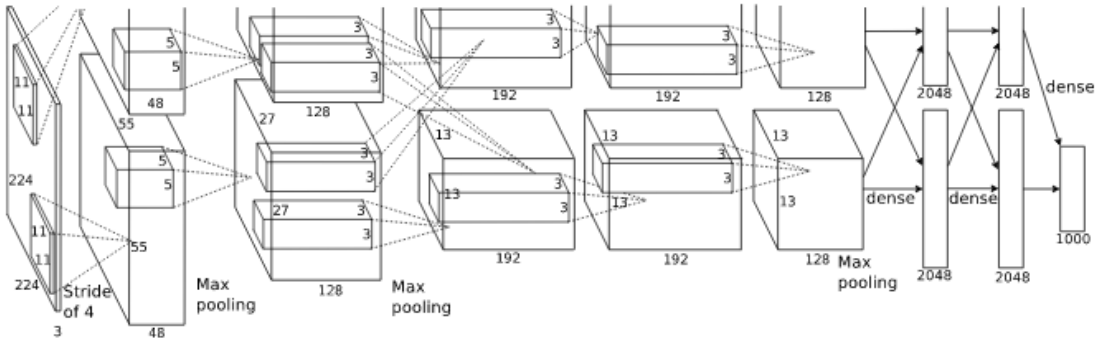


Figure 2.2: Architecture of deep convolutional neural network used in Krizhevsky et al. [29]

For visual recognition tasks, Deep Convolutional Neural Networks (DCNN) emerged as the most popular architectural choice. DCNNs typically comprise of the following elements: (1) convolutional layer, that performs convolution operation on a given input map using weights as filters, (2) max pooling layer, that performs downsampling operation along the spatial dimension, (3) Rectified Linear Unit (ReLU), which applies an elementwise non-linear operation of the form $ReLU(x) = max(0, x)$, (4) Batch Normalization layer, that performs normalization of the feature maps and (5) Fully connected layers, which are linear transformations applied on the input feature vector. These componenents are stacked together sequentially multiple times, and this results in a deep convolutional neural network. A visualization of one such DCNN architecture is shown in Figure 2.2.

Over the years, several architectural designs have been proposed for DCNNs [5, 30, 31, 32, 33]. Two prominant architectures that are widely used till date are Resnet [5] and VGGnet [31]. In VGGnet, authors use small convolutions filter sizes

(3×3 in specific) and make the network very deep (16-19 layer deep). In Resnet, authors propose using residual connections as a way to avoid the vanishing gradient problems in model training. This enabled the authors to train models much deeper (upto 152 layers deep) models. In this dissertation, we mainly use VGGnet and Resnet in our experiments.

Training: DCNN-based models typically have millions of parameters that need to be optimized. These models are trained using mini-batch stochastic gradient descent. For an input $\mathbf{x}$, let $F_\theta(\mathbf{x})$ denote the output of the neural network with parameters θ. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote the dataset used for training the model. Let $\tilde{y}$ be the one-hot encoding of the label y. To train the model $F(\cdot)$, we use the cross-entropy loss given by

$$\mathcal{L}_{cls} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[(\tilde{y})^t F_\theta(\mathbf{x}) \right] \tag{2.1}$$

Let $\theta^{(i)}$ denote the parameters at the i^{th} step of training. Then, the gradient descent updates of the parameter vector θ can be written as

$$\theta^{(i+1)} = \theta^{(i)} - \eta \nabla_\theta \mathcal{L}_{cls}$$

Here, η is the learning rate. For a deep neural network, the gradients $\nabla_\theta \mathcal{L}_{cls}$ for each layer can be efficiently computed using an algorithm called back-propagation.

Testing: Once the models are trained, they are evaluated by measuring the performance on a held-out test set. For classification problems, top-1 accuracy,

precision and recall are some of the commonly used evaluation metrics.

2.2 Generative Models

In the previous section, we discussed the classification task where the objective was to learn a decision boundary between different object classes. These methods fall under the category of discriminative modeling. In this section, we will discuss the other prominent class of approaches in machine learning, called generative modeling. In generative modeling, the objective is to learn the underlying data distribution directly. In this dissertation, we focus on deep generative modeling, in which deep networks are used for the generative modeling task. In particular, we focus on two types of models - GANs [34] and Variational Autoencoders (VAEs) [35].

2.2.1 Generative Adversarial Networks

Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\} \sim p_{data}$ denote the input dataset. The objective of GANs is to train a model to synthesize samples that resemble the input distribution p_{data}. To do this, we begin by generating samples from a prior distribution p_z (which is typically $\mathcal{N}(\mathbf{0}, I)$) and passing the samples through a transformation network G, which is also known as the generator network. The generated samples are then distinguished from the real samples using a discriminator network D. The generator

network is trained to fool the discriminator at this task.

$$\mathbf{z}_1, \mathbf{z}_2 \ldots \mathbf{z}_n \sim p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, I)$$

$$p_g(\mathbf{z}) := G(p_z(\mathbf{z}))$$

Then, the objective of GANs can be written as the following min-max game.

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z}[\log(1 - D(G(\mathbf{z})))] \tag{2.2}$$

As shown in Eq. (2.2), the discriminator response $D(\mathbf{x})$ assigns the probability that a sample $\mathbf{x}$ came from data rather than the generated distribution. The generator G, on the other hand, minimizes this loss. Hence, the generator G is trained to generate samples that fool the discriminator. The global optimum is achieved only when $p_g = p_{data}$ i.e., when generated distribution equals the data distribution. For the proof of this claim, please refer to Goodfellow et al. [34].

GANs are typically optimized using alternating stochastic gradient descent updates between discriminator and generator parameters. In practice, this optimization is extremely unstable and often leads to poor local optima. In recent years, several modifications have been proposed to improve the stability and convergence of GANs, including novel variants of the GAN objective such as Wasserstein GAN [36], least squares GAN [37], hinge loss GAN [38] and f-divergence GAN [39], regularization approaches such as gradient penalty [40], spectral normalization [41] and feature matching [42], and several architectural improvements [43, 44, 45]. These

(a) BigGAN (b) StyleGAN2

Figure 2.3: Photorealistic sample generation from GANs. The figure on the left shows samples generated from a BigGAN model [43], while the figure on the right shows the images of faces generated from a StyleGAN2 model [45].

modifications led to significant improvements in the quality of generated samples, achieving photo-realistic synthesis in several image-based datasets. Some examples taken from BigGAN and StyleGAN, two of the state-of-the-art GAN models, are shown in Figure 2.3.

2.2.2 Variational Autoencoders

Recall, that the objective of generative modeling is to estimate the underlying probability density $p(\mathbf{x})$ of a data distribution. As discussed in the previous section, GANs provide a framework for sampling from a generative distribution that resembles $p(\mathbf{x})$. However, there is no way to compute the sample likelihood scores. Variational autoencoders, on the other hand, directly optimizes for a lower-bound of the data likelihood, thereby enabling sample likelihood computation.

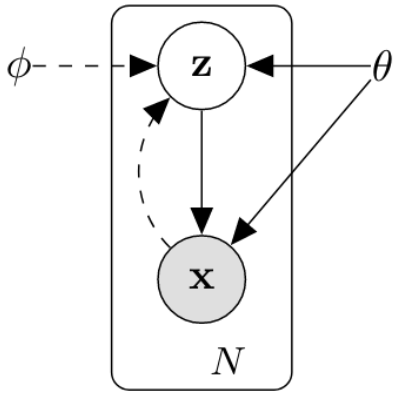


Figure 2.4: Graphical model considered in VAEs. Solid lines denote the generative model $p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$, while the dashed lines denote the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$. Figure taken from Kingma and Welling [35].

Let $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ denote a dataset of n i.i.d samples of some continuous random variable $\mathbf{x}$. We assume that the data is generated by some random process involving an unobserved latent variable $\mathbf{z}$. The data generating process consists of the two steps: (1) The latent variable $\mathbf{z}_i$ is first generated from some prior distribution $p_{\theta*}(\mathbf{z})$; (2) The input $\mathbf{x}_i$ is then generated from the likehood model $p_{\theta*}(\mathbf{x}|\mathbf{z})$. Both prior and likelihood are assumed to come from a parametric families $p_\theta(\mathbf{x}|\mathbf{z})$ and $p_\theta(\mathbf{z})$. Estimating $p_\theta(\mathbf{x}|\mathbf{z})$ requires computing $p_\theta(\mathbf{z}|\mathbf{x})$ which is intractable. Hence, a variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ is used for approximating $p_\theta(\mathbf{z}|\mathbf{x})$. This leads to a graphical model as shown in Figure 2.4.

We can write the marginal likelihood of individual datapoints as follows:

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \left(\int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \right) \\
&= \log \left(\int p_\theta(\mathbf{x}|\mathbf{z}) \frac{q(\mathbf{z})}{q(\mathbf{z})} p(\mathbf{z}) d\mathbf{z} \right) \\
&= \log \left(\mathbb{E}_{q(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} \right] \right) \\
&\geq \mathbb{E}_{q(\mathbf{z})} \left[\log \left(p_\theta(\mathbf{x}|\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} \right) \right] \qquad \text{(Jensen's Inequality)} \\
&\geq \mathbb{E}_{q(\mathbf{z})} \left[\log \left(p_\theta(\mathbf{x}|\mathbf{z}) \right) \right] - KL(q(\mathbf{z})||p(\mathbf{z}))
\end{aligned}$$

Since the above inequality holds for every $q(\mathbf{z})$, we replace it with $q_\phi(\mathbf{z}|\mathbf{x})$. This gives us the following bound.

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left(p_\theta(\mathbf{x}|\mathbf{z}) \right) \right] - KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{2.3}$$

Eq. (2.3) is the popular *Evidence Lower Bound* (ELBO). Now, to train a variational autoencoder, instead of maximizing the data likelihood $p(\mathbf{x})$, we can maximize the evidence lower bound i.e., r.h.s. of Eq. (2.3). For mathematical convenience, $q_\phi(\mathbf{z}|\mathbf{x})$ is modeled as a Gaussian distribution $\mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$. The mean and covariance of this Gaussian distribution $\mu_\phi(\mathbf{x})$ and $\Sigma_\phi(\mathbf{x}))$ are implemented using a neural network with parameters ϕ. The prior distribution $p(\mathbf{z})$ is usually modeled as a isotropic Gaussian $\mathcal{N}(\mathbf{0}, I)$.

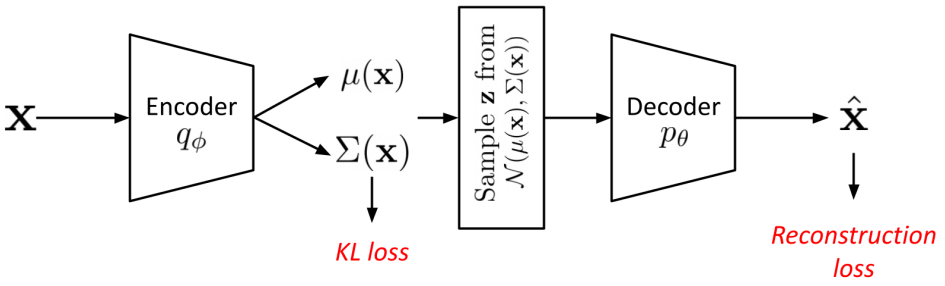


Figure 2.5: VAE framework. The input $\mathbf{x}$ is first passed to an encoder network q_ϕ to produce the mean and covariance vectors. The latents are then sampled using the *reparameterizartion trick*. The sampled latents are reconstructed back ussing a decoder network p_θ.

Using these models, the KL divergence term in Eq. (2.3) can be simplified as

$$KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2}\left[tr(\Sigma_\phi(\mathbf{x})) + \mu_\phi(\mathbf{x})^\intercal \mu_\phi(\mathbf{x}) - k - \log\det(\Sigma_\phi(\mathbf{x}))\right]$$

The first term in Eq. (2.3) is tricky as it involves sampling over the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and taking the expectation. Taking the gradients of this expectation term involves backpropagating over the sampling step, which is non-trivial. To fix this issue, a simple technique called *reparameterization trick* is used. The idea is to move the sampling step to an input layer. That is, to sample from $\mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$, we can first sample $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$ and then compute $\mathbf{z} = \mu_\phi(\mathbf{x}) + \Sigma^{1/2}(\mathbf{x}) * \epsilon$. Since sampling ϵ does not have any parameters, and we can backprogate through the network ϕ.

Combining all these tricks, we can train a Variational autoencoder by maximizing the expected lower-bound of Eq. (2.3). The framework of variational au-

toencoder invoves an encoder network q_ϕ that produces the mean and covariance vectors, and a decoder model $p_\theta(\mathbf{x}|\mathbf{z})$ that decodes the latents back into the image space. Please refer to Fig. 2.5 for an illustration. For a complete treatement, please refer to Doersch [46].

2.3 Optimal Transport

Estimating distances between probability distributions lies at the heart of several machine learning and statistics applications. Some distance measures include KL divergence [47], f-divergence, MMD distance [48], etc. In this section, we discuss optimal transport which is one popular framework for distributional distance estimation. Given two distributions, optimal transport finds the minimum cost plan for transporting one distribution to the other.

Let $\mathcal{X}$ denote a compact metric space, and $Prob(\mathcal{X})$ denote the space of probability measures defined on $\mathcal{X}$. Given two probability distributions $p_X, p_Y \in Prob(\mathcal{X})$ and a continuous cost function $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, optimal transport finds the minimum cost for transporting the density p_X to p_Y. This can be written as the following optimization problem

$$\mathcal{W}(p_X, p_Y) = \min_{\pi \in \Pi(p_X, p_Y)} \int \int c(\mathbf{x}, \mathbf{y}) \pi(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \tag{2.4}$$

where $\Pi(p_X, p_Y)$ is the set of all joint distributions whose marginals are p_X and p_Y,

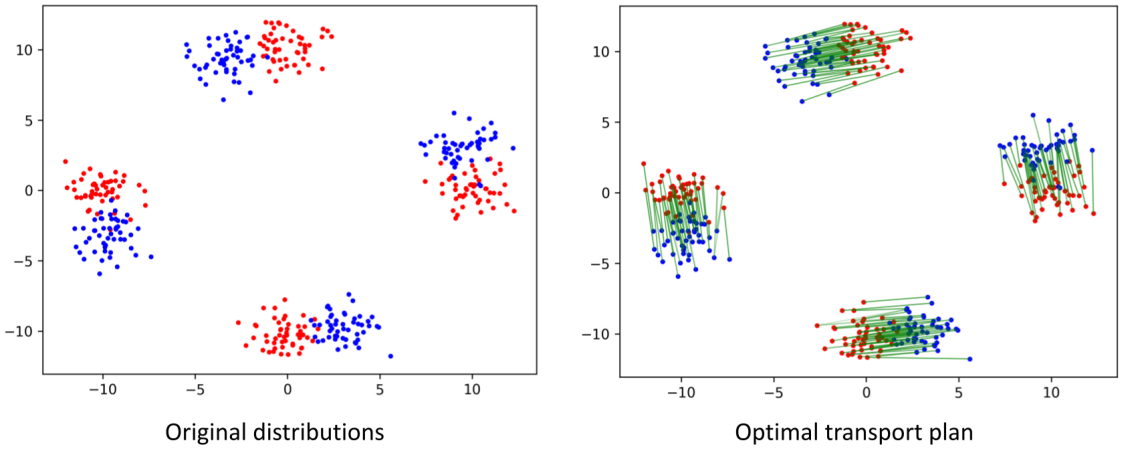


Figure 2.6: Optimal Transport. In the left panel, we show two distributions in red and blue, for which we are interested in computing the optimal transport plan. The computed transportation plan is shown in the right panel. The green lines are the couplings i.e., they show how each point in one distribution is coupled to every point in the other distribution.

respectively. That is,

$$\Pi(p_X, p_Y) = \{\pi(\mathbf{x}, \mathbf{y}) | \int \pi(\mathbf{x}, \mathbf{y}) d\mathbf{x} = p_Y, \int \pi(\mathbf{x}, \mathbf{y}) d\mathbf{y} = p_X\}$$

When the cost function $c(x, y)$ is ℓ_2 distance, then the optimal transport distance is called Wasserstein distance. Observe that Eq. (2.4) is a linear program since the objective and constraints are both linear in the optimization variable π.

A visualization of the optimal transport computation is shown in Figure. 2.6. Here, we are interested in computing the optimal transport plan between two mixtures of Gaussians shown in red and blue, respectively. The obtained transportation plan is shown in green lines. We observe that each Gaussian in one mixture distri-

Figure 2.7: Some samples generated from a Wasserstein GAN trained on LSUN-Bedrooms dataset. Figure taken from Arjovsky et al. [36].

bution is coupled to the nearest Gaussian in the other distribution.

2.3.1 Kantrovich-Rubinstein Duality

Let us assume that the cost function $c(\cdot)$ is a distance in some metric space. Then, the following duality holds

$$\min_{\pi \in \Pi(p_X, p_Y)} \int \int c(\mathbf{x}, \mathbf{y}) \pi(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} = \max_{\phi \in Lip-1} \int \phi(\mathbf{x}) dp_X - \int \phi(\mathbf{x}) dp_Y \tag{2.5}$$

Here, $Lip - 1$ denotes the set of functions that are 1-Lipschitz. Please refer to [49] for the proof of this duality. In practice, especially for neural networks, optimizing the dual form is much easier than the primal since optimization is over a set of 1-Lipschitz functions which are much easier to implement.

2.3.2 Applications: Wasserstein GAN

One of the popular applications of Wasserstein distance is in training a Generative Adversarial Network. Given a parameteric model for generating samples, a Wasserstein GAN can be trained by minimizing the Wasserstein distance between real data distribution and the generative distribution. As discussed in Sec. 2.2.1, the generative distribution p_g is modeled as a parameteric transformation applied to a latent space - $G(\mathbf{z})$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)$.

Let p_{data} denote the input data distribution. Then, the Wasserstein GAN model can be trained by minimizing

$$\begin{aligned}\min_G W(p_{data}, p_g) &= \min_G \min_{\pi \in \Pi(p_{data}, p_g)} \int\int c(\mathbf{x}, \mathbf{y})\pi(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= \min_G \max_{D \in Lip-1} \mathbb{E}_{\mathbf{x} \sim p_{data}} D(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} D(G(\mathbf{z})) \qquad \text{From (2.5)}\end{aligned}$$

The last equation follows from the Kantrovich-Rubinstein duality, where we replaced the dual variable ϕ with D. In this formulation, G is the generator and D is the discriminator. We observe that this optimization is similar to the GAN objective in Eq. (2.2), with the difference being the log term in the expectation is replaced with identity, and $1-$Lipschitzness is imposed on the discriminator.

In practice, the generator and discriminator functions are implemented as deep neural networks. 1-Lipschitz constrained is imposed using tricks such as weight clipping [36], gradient penalty [40] or spectral normalization [41]. Samples generated by Wasserstein GAN trained on LSUN-Bedrooms dataset is shown in Fig. 2.7

Part I

Likelihood Estimation

Chapter 3: Entropic GANs meet VAEs

3.1 Introduction

Consider an object recognition system that is deployed in the wild. While the classification model is trained on images of certain pre-defined object categories, it could encounter noisy out-of-distribution images that do not look like any of the objects of interest. In this case, our object recognition system should be able to filter out these anomalous samples instead of making a prediction. One approach for addressing the out-of-distribution detection problem is to learn a probability model of the data distribution, and using sample likelihood scores to filter out the anomalous samples.

Deep generative models provide a framework for modeling the input data distribution. As discussed in Section. 2.2, VAEs [35] compute a generative model by maximizing a variational lower-bound on average sample log-likelihoods using an explicit probability model for the data. GANs, however, learn a generative model by minimizing a distance between observed and generated distributions without considering an explicit probability model. Empirically, GANs have been shown to produce higher-quality generative samples than that of VAEs [50]. However, since GANs do not consider an explicit probability model for the data, we are

unable to compute sample likelihoods using their generative models. Obtaining sample likelihoods and *posterior* distributions of latent variables are critical in several statistical inference applications. Inability to obtain such statistics within GAN's framework severely limits their applications in statistical inference problems.

In this chapter, we resolve this issue for a general formulation of GANs by providing a theoretically-justified approach to compute sample likelihoods using GAN's generative model. Our results facilitate the use of GANs in massive-data applications such as model selection, sample selection, hypothesis-testing, etc.

We first state our main results *informally* without going into technical details while precise statements of our results are presented in Section 3.2. Let X and $\hat{X} := \mathbf{G}(Z)$ represent observed (i.e. real) and generative (i.e. fake or synthetic) variables, respectively. Z (i.e. the latent variable) is the random vector used as the input to the generator $\mathbf{G}(.)$. Consider the following explicit probability model of the data given a latent sample $Z = \mathbf{z}$:

$$p_{X|Z=\mathbf{z}}(\mathbf{x}) \propto \exp(-\ell(\mathbf{x}, \mathbf{G}(\mathbf{z}))), \tag{3.1}$$

where $\ell(.,.)$ is a loss function. $p_{X|Z=\mathbf{z}}(\mathbf{x})$ is the model that we are considering for the underlying data distribution. This is a reasonable model for the data as the function $\mathbf{G}$ can be an arbitrarily complex function. Similar data models have been used in VAEs. Under this explicit probability model, we show that minimizing the objective of an *optimal transport* GAN (e.g. Wasserstein GAN [36]) with the cost function $\ell(.,.)$ and an entropy regularization [51, 52] maximizes a variational lower-bound on

average sample likelihoods. That is

$$\underbrace{\mathbb{E}_{p_X}\left[\log p_X(X)\right]}_{\text{ave. sample log likelihoods}} \geq -\frac{1}{\lambda}\underbrace{\left\{\mathbb{E}_{\mathbb{P}_{X,\hat{X}}}\left[\ell(X,\hat{X})\right] - \lambda H\left(\mathbb{P}_{X,\hat{X}}\right)\right\}}_{\text{entropic GAN objective}} + \textit{constants}.$$

This result provides a statistical justification for GAN's optimization and puts it in par with VAEs whose goal is to maximize a lower bound on sample likelihoods. We note that entropy regularization has been proposed primarily to improve computational aspects of GANs [53]. Our results provide an additional statistical justification for this regularization term. Moreover, using the GAN's training, we obtain a coupling between the observed variable X and the latent variable Z. This coupling provides the conditional distribution of the latent variable Z given an observed sample $X = \mathbf{x}$. The explicit model of (3.1) acts similar to the *decoder* in the VAE framework, while the coupling computed using GANs acts as an *encoder*.

Another key question is how to estimate the likelihood of a new sample $\mathbf{x}^{\text{test}}$ given the generative model trained using GANs. For instance, if we train a GAN on *stop-sign* images, upon receiving a new image, one may wish to compute the likelihood of the new sample $\mathbf{x}^{\text{test}}$ according to the trained generative model. In standard GAN formulations, the support of the generative distribution lies on the range of the optimal generator function. Thus, if the observed sample $\mathbf{x}^{\text{test}}$ does not lie in that range (which is very likely in practice), there is no way to assign a sensible likelihood score to the sample. Below, we show that using the explicit probability

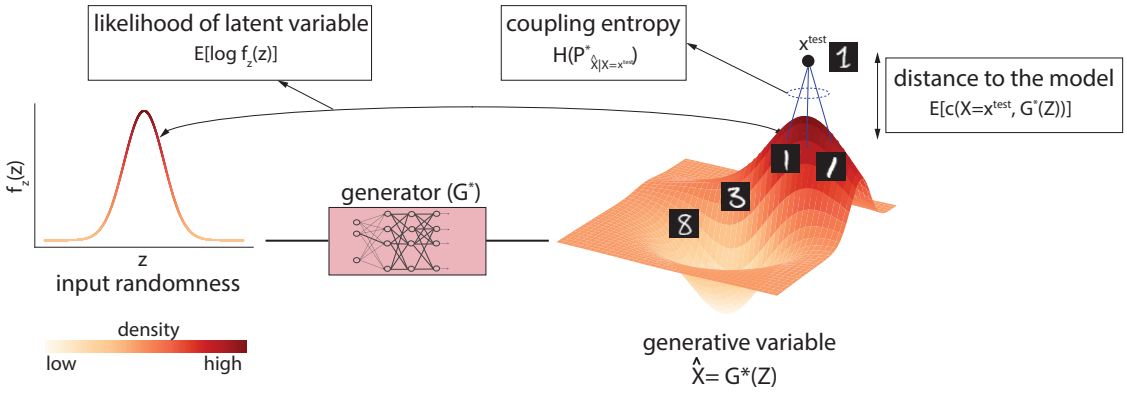


Figure 3.1: A statistical framework for GANs. By training a GAN model, we first compute optimal generator $\mathbf{G}^*$ and optimal coupling between the observed variable X and the latent variable Z. The likelihood of a test sample $\mathbf{x}^{\text{test}}$ can then be lower-bounded using a combination of three terms: (1) the expected distance of $\mathbf{x}^{\text{test}}$ to the distribution learnt by the generative model, (2) the entropy of the coupled latent variable given $\mathbf{x}^{\text{test}}$ and (3) the likelihood of the coupled latent variable with $\mathbf{x}^{\text{test}}$.

model of (3.1), we can *lower-bound* the likelihood of this sample $\mathbf{x}^{\text{test}}$. This is similar to the variational lower-bound on sample likelihoods used in VAEs. Our numerical results in Section 3.4 show that this lower-bound well-reflects the expected trends of the true sample likelihoods.

Let $\mathbf{G}^*$ and $\mathbb{P}^*_{X,Z}$ be the optimal generator and the optimal coupling between real and latent variables, respectively. The optimal coupling $\mathbb{P}^*_{X,Z}$ can be computed efficiently for entropic GANs as we explain in Section 3.3. For other GAN architectures, one may approximate such couplings as we explain in Section 3.4. The log likelihood of a new test sample $\mathbf{x}^{\text{test}}$ can be lower-bounded as

$$\underbrace{\log p_X(\mathbf{x}^{\text{test}})}_{\text{log likelihood}} \geq -\underbrace{\mathbb{E}_{\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}}\left[\ell(\mathbf{x}^{\text{test}}, \mathbf{G}^*(\mathbf{z}))\right]}_{\text{distance to the generative model}} + \underbrace{H\left(\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}\right)}_{\text{coupling entropy}} + \underbrace{\mathbb{E}_{\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}}\left[-\frac{\|\mathbf{z}\|^2}{2}\right]}_{\text{likelihood of latent variable}}. \tag{3.2}$$

We present the precise statement of this result in Corollary 2. This result combines three components in order to approximate the likelihood of a sample given a trained generative model: (1) if the distance between $\mathbf{x}^{\text{test}}$ to the generative model is large, the likelihood of observing $\mathbf{x}^{\text{test}}$ from the generative model is small, (2) if the entropy of the coupled latent variable is large, the coupled latent variable has large randomness, thus, this contributes positively to the sample likelihood, and (3) if the likelihood of the coupled latent variable is large, the likelihood of the observed test sample will be large as well. Figure 3.1 provides a pictorial illustration of these components.

To summarize, we have made the following **theoretical contributions**:

- We have constructed an explicit probability model for a family of optimal transport GANs (such as the Wasserstein GAN) that can be used to compute likelihood statistics within GAN's framework (eq. (3.6) and Corollary 2).

- We have proved that, under this probability model, the objective of an entropic GAN is a variational lower bound for average sample log likelihoods (Theorem 1). This result makes a principled connection between two modern generative models, namely GANs and VAEs.

Moreover, we have made the following **empirical contributions**:

- We have computed likelihood statistics for GANs trained on Gaussian, MNIST, SVHN, CIFAR-10 and LSUN datasets and shown the consistency of these empirical results with the proposed theory (Section 3.4).
- We have demonstrated the tightness of the variational lower bound of entropic GANs for both linear and non-linear generators (Section 3.4.4).

3.1.1 Related Work

Connections between GANs and VAEs have been investigated in some of the recent works as well [54, 55]. In [54], GANs are interpreted as models performing variational inference on a generative model in the label space. In [54], observed data samples are treated as latent variables while the generative variable is the indicator of whether data is real or fake. The method in [55], on the other hand, uses an auxiliary discriminator network to rephrase the maximum-likelihood objective of a VAE as a two-player game similar to the objective of a GAN. Our method is different from both of these approaches as we consider an explicit probability model for the data, and show that the entropic GAN objective maximizes a variational lower bound under this probability model, thus allowing sample likelihood computation in GANs similar to VAEs.

Of relevance to our work is [56], in which annealed importance sampling (AIS) is used to evaluate the approximate likelihood of decoder-based generative models. More specifically, a Gaussian observation model with a fixed variance is used as the

generative distribution for GAN-based models on which the AIS is computed. Gaussian observation models may not be proper specially in high-dimensional spaces. Our approach, on the other hand, makes a connection between GANs and VAEs by constructing a theoretically-motivated model for the data distribution in GANs. We then leverage this approach in computing sample likelihood estimates in GANs.

3.2 A Variational Bound for GANs

Let $X \in \mathbb{R}^d$ represent the real-data random variable with a probability density function $p_X(\mathbf{x})$. GAN's goal is to find a generator function $\mathbf{G} : \mathbb{R}^r \to \mathbb{R}^d$ such that $\hat{X} := \mathbf{G}(Z)$ has a similar distribution to X. Let Z be an r-dimensional random vector with a fixed probability density function $p_Z(\mathbf{z})$. Here, we assume $p_Z(.)$ is the density of a Gaussian distribution. In practice, we observe m samples $\{\mathbf{x}_1, ..., \mathbf{x}_m\}$ from X and generate m' samples from $\hat{X}$, i.e., $\{\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_{m'}\}$ where $\hat{\mathbf{x}}_i = \mathbf{G}(\mathbf{z}_i)$ for $1 \leq i \leq m$. We represent these empirical distributions by $\mathbb{P}_X$ and $\mathbb{P}_{\hat{X}}$, respectively. Note that the number of generative samples m' can be arbitrarily large.

GAN computes the optimal generator $\mathbf{G}^*$ by minimizing a distance between the observed empirical distribution $\mathbb{P}_X$ and the generative one $\mathbb{P}_{\hat{X}}$. Common distance measures include *optimal transport* measures (e.g. Wasserstein GAN [36], WGAN+Gradient Penalty [40], GAN+Spectral Normalization [41], WGAN+Truncated Gradient Penalty [57], relaxed WGAN [58]), and *divergence* measures (e.g. the original GAN's formulation [34], f-GAN [39]), etc.

We focus on GANs based on optimal transport (OT) distance [36, 49] defined

for a general loss function $\ell(.,.)$ as follows

$$W_\ell(\mathbb{P}_X, \mathbb{P}_{\hat{X}}) := \min_{\mathbb{P}_{X,\hat{X}}} \mathbb{E}\left[\ell(X, \hat{X})\right]. \tag{3.3}$$

$\mathbb{P}_{X,\hat{X}}$ is the joint distribution whose marginal distributions are equal to $\mathbb{P}_X$ and $\mathbb{P}_{\hat{X}}$, respectively. If $\ell(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2$, this distance is called the first-order Wasserstein distance and is referred to by $W_1(.,.)$, while if $\ell(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$, this measure is referred to by $W_2(.,.)$ where W_2 is the second-order Wasserstein distance [49]. The OT GAN is formulated using the following optimization problem [36, 49]:

$$\min_{\mathbf{G}\in\mathcal{G}} W_\ell(\mathbb{P}_X, \mathbb{P}_{\hat{X}}), \tag{3.4}$$

where $\mathcal{G}$ is the set of generator functions. Examples of the OT GAN are WGAN [36] corresponding to the first-order Wasserstein distance [1] and the quadratic GAN (or, the W2GAN) [59] corresponding to the second-order Wasserstein distance.

Note that optimization 3.4 is a *min-min* optimization. The objective of this optimization is not smooth in $\mathbf{G}$ and it is often computationally expensive to obtain a solution for it [60]. One approach to improve computational aspects of this optimization problem is to add a regularization term to make its objective *strongly* convex [51, 52]. The Shannon entropy function is defined as $H(\mathbb{P}_{X,\hat{X}}) := -\mathbb{E}\left[\log \mathbb{P}_{X,\hat{X}}\right]$. The negative Shannon entropy is a common strongly-convex regularization term. This leads to the following optimal transport GAN formulation with the entropy

[1] Note that some references (e.g. [36]) refer to the first-order Wasserstein distance simply as the Wasserstein distance. In this chapter, we explicitly distinguish between different Wasserstein distances.

regularization, or for simplicity, the *entropic GAN* formulation:

$$\min_{\mathbf{G}\in\mathcal{G}} \min_{\mathbb{P}_{X,\hat{X}}} \mathbb{E}\left[\ell(X,\hat{X})\right] - \lambda H\left(\mathbb{P}_{X,\hat{X}}\right), \tag{3.5}$$

where λ is the regularization parameter.

There are two approaches to solve the optimization problem 3.5. The first approach uses an iterative method to solve the *min-min* formulation [61]. Another approach is to solve an equivelent *min-max* formulation by writing the dual of the inner minimization [52, 60]. The latter is often referred to as a GAN formulation since the min-max optimization is over a set of generator functions and a set of discriminator functions (as discussed in Section. 2.3). The details of this approach are further explained in Section 3.3.

In the following, we present an explicit probability model for entropic GANs under which their objective can be viewed as maximizing a lower bound on average sample likelihoods.

Theorem 1. *Let the loss function be shift invariant, i.e.,* $\ell(\mathbf{x},\hat{\mathbf{x}}) = h(\mathbf{x}-\hat{\mathbf{x}})$. *Let*

$$p_{X|Z=\mathbf{z}}(\mathbf{y}) = C\exp(-\ell(\mathbf{x},\mathbf{G}(\mathbf{z}))/\lambda), \tag{3.6}$$

be an explicit probability model for X *given* $Z=\mathbf{z}$ *for a well-defined normalization*

$$C := \frac{1}{\int_{\mathbf{x}\in\mathbb{R}^d}\exp(-\ell(\mathbf{x},G(\mathbf{z}))/\lambda)}. \tag{3.7}$$

Then, we have

$$\underbrace{\mathbb{E}_{\mathbb{P}_X}\left[\log p_X(X)\right]}_{\textit{ave. sample likelihoods}} \geq -\frac{1}{\lambda}\underbrace{\left\{\mathbb{E}_{\mathbb{P}_{X,\hat{X}}}\left[\ell(X,\hat{X})\right] - \lambda H\left(\mathbb{P}_{X,\hat{X}}\right)\right\}}_{\textit{entropic GAN objective}} + \textit{constants}. \quad (3.8)$$

In words, the entropic GAN maximizes a lower bound on sample likelihoods according to the explicit probability model of (3.6).

Proof. Using the Baye's rule, one can compute the log-likelihood of an observed sample $\mathbf{x}$ as follows:

$$\begin{aligned}\log p_X(\mathbf{x}) &= \log p_{X|Z=\mathbf{z}}(\mathbf{z}) + \log p_Z(\mathbf{z}) - \log p_{Z|X=\mathbf{x}}(\mathbf{z}) \\ &= \log C - \ell(\mathbf{x}, G(\mathbf{z})) - \log\sqrt{2\pi} - \frac{\|\mathbf{z}\|^2}{2} - \log p_{Z|X=\mathbf{x}}(\mathbf{z}),\end{aligned} \quad (3.9)$$

where the second step follows from Eq. (3.6).

Consider a joint density function $\mathbb{P}_{Z,X}$ such that its marginal distributions match $\mathbb{P}_Z$ and $\mathbb{P}_X$. Note that the equation 3.9 is true for every $\mathbf{z}$. Thus, we can take the expectation of both sides with respect to a distribution $\mathbb{P}_{Z|X=\mathbf{x}}$. This leads to the following equation:

$$\log p_X(\mathbf{x}) = \mathbb{E}_{\mathbb{P}_{Z|X=\mathbf{x}}}\Big[-\frac{\ell(\mathbf{x}, \mathbf{G}(\mathbf{z}))}{\lambda} + \log C - \frac{1}{2}\log 2\pi - \frac{\|\mathbf{z}\|^2}{2} - \log p_{Z|X=\mathbf{x}}(\mathbf{z})\Big] \tag{3.10}$$

$$\begin{aligned}
&= \mathbb{E}_{\mathbb{P}_{Z|X=\mathbf{x}}}\Big[-\ell(\mathbf{x}, \mathbf{G}(\mathbf{z}))/\lambda + \log C - \frac{1}{2}\log 2\pi - \frac{\|\mathbf{z}\|^2}{2} - \log f_{Z|X=\mathbf{x}}(\mathbf{z}) \\
&\quad + \log\left(\mathbb{P}_{Z|X=\mathbf{x}}(\mathbf{z})\right) - \log\left(\mathbb{P}_{X|Y=\mathbf{y}}(\mathbf{x})\right)\Big] \\
&= -\mathbb{E}_{\mathbb{P}_{Z|X=\mathbf{x}}}\left[\ell(\mathbf{x}, \mathbf{G}(\mathbf{z}))/\lambda\right] - \frac{1}{2}\log 2\pi + \log C + \mathbb{E}_{\mathbb{P}_{Z|X=\mathbf{x}}}\left[-\frac{\|\mathbf{z}\|^2}{2}\right] \\
&\quad + \mathrm{KL}\left(\mathbb{P}_{Z|X=\mathbf{x}} || p_{Z|X=\mathbf{x}}\right) + H\left(\mathbb{P}_{Z|X=\mathbf{x}}\right),
\end{aligned} \tag{3.11}$$

where $H(.)$ is the Shannon-entropy function.

Next we take the expectation of both sides with respect to $\mathbb{P}_X$:

$$\begin{aligned}
\mathbb{E}\left[\log p_X(X)\right] = &-\frac{1}{\lambda}\mathbb{E}_{\mathbb{P}_{Z,X}}\left[\ell(\mathbf{x}, G(\mathbf{z}))\right] - \frac{1}{2}\log 2\pi + \log C + \mathbb{E}_{p_Z}\left[-\frac{\|\mathbf{z}\|^2}{2}\right] \\
&+ \mathbb{E}_{\mathbb{P}_X}\left[\mathrm{KL}\left(\mathbb{P}_{Z|X=\mathbf{x}} || p_{Z|X=\mathbf{x}}\right)\right] + H\left(\mathbb{P}_{Z,X}\right) - H\left(\mathbb{P}_X\right).
\end{aligned} \tag{3.12}$$

Here, we replaced the expectation over $\mathbb{P}_X$ with the expectation over p_X since one can generate an arbitrarily large number of samples from the generator. Since the KL divergence is always non-negative, we have

$$\begin{aligned}
\mathbb{E}\left[\log p_X(X)\right] \geq &-\frac{1}{\lambda}\left\{\mathbb{E}_{\mathbb{P}_{Z,X}}\left[\ell(\mathbf{x}, \mathbf{G}(\mathbf{z}))\right] - \lambda H\left(\mathbb{P}_{Z,X}\right)\right\} \\
&+ \log C - \log(m) - \frac{r + \log 2\pi}{2}
\end{aligned} \tag{3.13}$$

Moreover, using the data processing inequality, we have $H(\mathbb{P}_{Z,X}) \geq H(\mathbb{P}_{\mathbf{G}(Z),X})$ [47]. Thus,

$$\underbrace{\mathbb{E}\left[\log p_X(X)\right]}_{\text{sample likelihood}} \geq -\frac{1}{\lambda} \underbrace{\left\{\mathbb{E}_{\mathbb{P}_{Z,X}}\left[\ell(\mathbf{x}, \mathbf{G}(\mathbf{z}))\right] - \lambda H\left(\mathbb{P}_{X,\hat{X}}\right)\right\}}_{\text{GAN objective with entropy regularizer}} + \log C - \log(m) - \frac{r + \log 2\pi}{2} \tag{3.14}$$

This inequality is true for every $\mathbb{P}_{Z,X}$ satisfying the marginal conditions. Thus, similar to VAEs, we can pick $\mathbb{P}_{Z,X}$ to maximize the lower bound on average sample log-likelihoods. This leads to the entropic GAN optimization (3.5). This concludes the proof.

□

Theorem. 1 has a similar flavor to that of VAEs [62, 63, 64, 65] where a generative model is computed by maximizing a lower bound on sample likelihoods. Having a shift invariant loss function is critical for Theorem 1 as this makes the normalization term C independent from $\mathbf{G}$ and $\mathbf{x}$ (to see this, one can define $\mathbf{y}' := \mathbf{y} - \mathbf{G}(\mathbf{x})$ in (3.8)). The most standard OT GAN loss functions such as the ℓ_2 for WGAN [36] and the quadratic loss for W2GAN [59] satisfy this property.

One can further simplify this result by considering specific loss functions. For example, we have the following result for the entropic GAN with the quadratic loss function.

Corollary 1. *Let* $\ell(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2/2$. *Then,* $p_{X|Z=\mathbf{z}}(.)$ *of* (3.6) *corresponds to the*

multivariate Gaussian density function and $C = \frac{1}{\sqrt{(2\pi\lambda)^d}}$. *In this case, the constant term in* (3.8) *is equal to* $-\log(m) - d\log(2\pi\lambda)/2 - r/2 - \log(2\pi)/2$.

Let $\mathbf{G}^*$ and $\mathbb{P}^*_{X,Z}$ be optimal solutions of an entropic GAN optimization 3.5 (note that the optimal coupling can be computed efficiently using (3.19)). Let $\mathbf{x}^{\text{test}}$ be a newly observed sample. An important question is what the likelihood of this sample is given the trained generative model. Using the explicit probability model of (3.6) and the result of Theorem 1, we can (approximately) compute sample likelihoods as explained in the following corollary.

Corollary 2. *Let* $\mathbf{G}^*$ *and* $\mathbb{P}^*_{X,\hat{X}}$ *(or, alternatively* $\mathbb{P}^*_{X,Z}$*) be optimal solutions of the entropic GAN* (3.5). *Let* $\mathbf{x}^{test}$ *be a new observed sample. We have*

$$\log p_X(\mathbf{x}^{test}) \geq -\frac{1}{\lambda}\Big\{\mathbb{E}_{\mathbb{P}^*_{Z|X=\mathbf{x}^{test}}}\left[\ell(\mathbf{x}^{test}, \mathbf{G}^*(\mathbf{z}))\right] - \lambda H\left(\mathbb{P}^*_{Z|X=\mathbf{x}^{test}}\right)\Big\} \tag{3.15}$$
$$+ \mathbb{E}_{\mathbb{P}^*_{Z|X=\mathbf{x}^{test}}}\left[-\frac{\|\mathbf{z}\|^2}{2}\right] + constants.$$

The inequality becomes tight iff $KL\left(\mathbb{P}^*_{Z|X=\mathbf{x}^{test}}||p_{Z|X=\mathbf{x}^{test}}\right) = 0$, *where* $KL(.||.)$ *is the Kullback-Leibler divergence between two distributions.*

3.3 Dual of Entropic GANs

In Section. 2.3, we discussed the dual formulation of optimal transport. Similarly, the dual formulation of the entropic GAN (3.5) can be written as the following

optimization [51, 52] [2]:

$$\min_{\mathbf{G}\in\mathcal{G}} \max_{\mathbf{D}_1,\mathbf{D}_2} \mathbb{E}\left[\mathbf{D}_1(X)\right] - \mathbb{E}\left[\mathbf{D}_2(\mathbf{G}(Z))\right] - \lambda \mathbb{E}_{\mathbb{P}_X \times \mathbb{P}_{\hat{X}}}\left[\exp\left(v(\mathbf{x},\hat{\mathbf{x}})/\lambda\right)\right], \tag{3.16}$$

where

$$v(\mathbf{x},\hat{\mathbf{x}}) := \mathbf{D}_1(\mathbf{x}) - \mathbf{D}_2(\hat{\mathbf{x}}) - \ell(\mathbf{x},\hat{\mathbf{x}}). \tag{3.17}$$

Moreover, the optimal primal variables $\mathbb{P}^*_{X,\hat{X}}$ can be computed according to the following lemma [52]:

Lemma 1. *Let* $\mathbf{D}_1^*$ *and* $\mathbf{D}_2^*$ *be the optimal discriminator functions for a given generator function* $\mathbf{G}$ *according to optimization 3.16. Let*

$$v^*(\mathbf{x},\hat{\mathbf{x}}) := \mathbf{D}_1^*(\mathbf{x}) - \mathbf{D}_2^*(\hat{\mathbf{x}}) - \ell(\mathbf{x},\hat{\mathbf{x}}). \tag{3.18}$$

Then,

$$\mathbb{P}^*_{X,\hat{X}}(\mathbf{x},\hat{\mathbf{x}}) = \mathbb{P}_X(\mathbf{x})\mathbb{P}_{\hat{X}}(\hat{\mathbf{x}}) \exp\left(v^*\left(\mathbf{x},\hat{\mathbf{x}}\right)/\lambda\right). \tag{3.19}$$

This lemma is important since it provides an efficient way to compute the optimal coupling between real and generative variables (i.e. $P^*_{X,\hat{X}}$) using the optimal generator ($\mathbf{G}^*$) and discriminators ($\mathbf{D}_1^*$ and $\mathbf{D}_2^*$) of optimization 3.16. It is worth

[2] Note that optimization 3.16 is dual of optimization 3.5 when the terms $\lambda H(\mathbb{P}_X) + \lambda H(\mathbb{P}_{\hat{X}})$ have been added to its objective. Since for a fixed $\mathbf{G}$ (fixed marginals), these terms are constants, they can be ignored from the optimization objective without loss of generality.

noting that without the entropy regularization term, computing the optimal coupling using the optimal generator and discriminator functions is not straightforward in general (unless in some special cases such as W2GAN [49, 59]). This is another additional computational benefit of using entropic GAN.

3.4 Experimental Results

In this section, we supplement our theoretical results with experimental validations. One of the main objectives of our work is to provide a framework to compute sample likelihoods in GANs. Such likelihood statistics can then be used in several statistical inference applications that we discuss in Section 3.5. With a trained entropic WGAN, the likelihood of a test sample can be lower-bounded using Corollary 2. Note that this likelihood estimate requires the discriminators $\mathbf{D}_1$ and $\mathbf{D}_2$ to be solved to optimality. In our implementation, we use the algorithm presented in [60] to train the Entropic GAN. It has been proven in [60] that this algorithm leads to a good approximation of stationary solutions of Entropic GAN. We also discuss an approximate likelihood computation approach for un-regularized GANs in [66].

To obtain the surrogate likelihood estimates using Corollary 2, we need to compute the density $\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}(\mathbf{z})$. As shown in Lemma 1, WGAN with entropy regularization provides a closed-form solution to the conditional density of the latent variable (3.19). When $\mathbf{G}^*$ is injective, $\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}(\mathbf{z})$ can be obtained from (3.19) by change of variables. In general case, $\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}(\mathbf{z})$ is not well defined as multiple $\mathbf{z}$

can produce the same $\mathbf{x}^{\text{test}}$. In this case,

$$\mathbb{P}^*_{\hat{X}|x=\mathbf{x}^{\text{test}}}(\hat{\mathbf{x}}) = \sum_{\mathbf{z}|\mathbf{G}^*(\mathbf{z})=\mathbf{x}} \mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}(\mathbf{z}). \tag{3.20}$$

Also, from Eq. (3.19), we have

$$\mathbb{P}^*_{\hat{X}|X=\mathbf{x}^{\text{test}}}(\hat{\mathbf{x}}) = \sum_{\mathbf{z}|\mathbf{G}^*(\mathbf{z})=\mathbf{x}} \mathbb{P}_Z(\mathbf{z}) \exp\left(v^*\left(\mathbf{x}^{\text{test}}, \mathbf{G}^*(\mathbf{z})\right)/\lambda\right). \tag{3.21}$$

One solution (which may not be unique) that satisfies both (3.20) and 3.21 is

$$\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}(\mathbf{z}) = \mathbb{P}_Z(\mathbf{z}) \exp\left(v^*\left(\mathbf{x}^{\text{test}}, G^*(\mathbf{z})\right)/\lambda\right). \tag{3.22}$$

Ideally, we would like to choose $\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}(\mathbf{z})$, satisfying (3.20) and (3.21) that maximizes the lower bound of Corollary 2. But finding such a solution can be difficult in general. Instead we use (3.22) to evaluate the surrogate likelihoods of Corollary 2 (note that our results still hold in this case). In order to compute our proposed surrogate likelihood, we need to draw samples from the distribution $\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}(\mathbf{z})$. One approach is to use a Markov chain Monte Carlo (MCMC) method to sample from this distribution. In our experiments, however, we found that MCMC demonstrates poor performance owing to the high dimensional nature of X. A similar issue with MCMC has been reported for VAEs in [35]. Thus, we use a different estimator to compute the likelihood surrogate which provides a better exploration of the latent space. We present our sampling procedure in Algorithm 1.

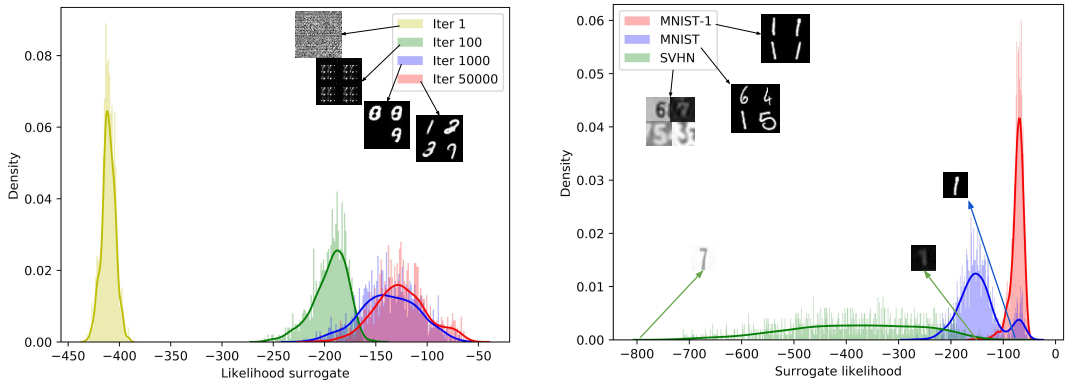
Algorithm 1 Estimating sample likelihoods in GANs

1: Sample n points $\mathbf{z}_i \overset{i.i.d}{\sim} U[-1, 1]$
2: Compute $u_i := p_Z(\mathbf{z}_i) \exp\left(v^*\left(\mathbf{x}^{\text{test}}, G^*(\mathbf{z}_i)\right)/\lambda\right)$
3: Normalize to get probabilities $\tilde{u}_i = \frac{u_i}{\sum_{i=1}^n u_i}$
4: Compute $L = -\frac{1}{\lambda}\left[\sum_{i=1}^n \tilde{u}_i \ell(\mathbf{x}^{\text{test}}, G^*(\mathbf{z}_i)) + \lambda \sum_{i=1}^N \tilde{u}_i \log \tilde{u}_i\right] - \sum_{i=1}^n \tilde{u}_i \frac{\|\mathbf{z}_i\|^2}{2}$
5: Return L

3.4.1 Likelihood Evolution in GAN's Training

In the following experiments, we study how sample likelihoods vary during GAN's training. An entropic WGAN is first trained on MNIST dataset. Then, we randomly choose $1,000$ samples from MNIST test-set to compute the surrogate likelihoods using Algorithm. 1 at different training iterations. Surrogate likelihood computation requires solving $\mathbf{D}_1$ and $\mathbf{D}_2$ to optimality for a given $\mathbf{G}$ (refer to Lemma. 1), which might not be satisfied at the intermediate iterations of the training process. Therefore, before computing the surrogate likelihoods, discriminators $\mathbf{D}_1$ and $\mathbf{D}_2$ are updated for 100 steps for a fixed $\mathbf{G}$. We expect sample likelihoods to increase over training iterations as the quality of the generative model improves.

Fig. 3.2(a) demonstrates the evolution of sample likelihood distributions at different training iterations of the entropic WGAN. At iteration 1, surrogate likelihood values are very low as GAN's generated images are merely random noise. The likelihood distribution shifts towards high values during the training and saturates beyond a point. Details of this experiment are presented in [66].



(a) (b)

Figure 3.2: (a) Distributions of surrogate sample likelihoods at different iterations of entropic WGAN's training using MNIST dataset. (b) Distributions of surrogate sample likelihoods of MNIST, MNIST-1 and SVHN datasets using a GAN trained on MNIST-1.

3.4.2 Likelihood Comparison Across Different Datasets

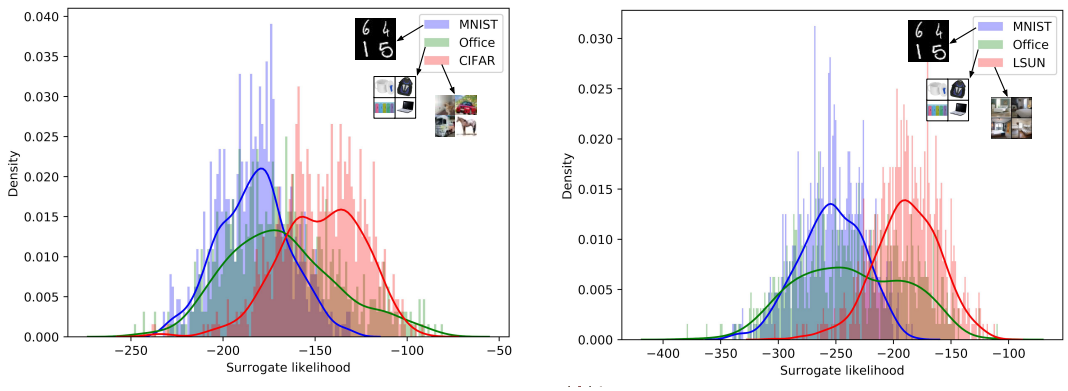
In this section, we perform experiments across different datasets. An entropic WGAN is first trained on a subset of samples from the MNIST dataset containing digit 1 (which we call the MNIST-1 dataset). With this trained model, likelihood estimates are computed for (1) samples from the entire MNIST dataset, and (2) samples from the Street View House Numbers (SVHN) dataset [67] (Fig. 3.2(b)). In each experiment, the likelihood estimates are computed for 1000 samples. We note that highest likelihood estimates are obtained for samples from MNIST-1 dataset, the same dataset on which the GAN was trained. The likelihood distribution for the MNIST dataset is bimodal with one mode peaking inline with the MNIST-1 mode. Samples from this mode correspond to digit 1 in the MNIST dataset. The other mode, which is the dominant one, contains the rest of the digits and has

relatively low likelihood estimates. The SVHN dataset, on the other hand, has much smaller likelihoods as its distribution is significantly different than that of MNIST. Furthermore, we observe that the likelihood distribution of SVHN samples has a large spread (variance). This is because samples of the SVHN dataset is more diverse with varying backgrounds and styles than samples from MNIST. We note that SVHN samples with high likelihood estimates correspond to images that are similar to MNIST digits, while samples with low scores are different than MNIST samples. Details of this experiment are presented in [66].

3.4.3 Approximate Likelihood Computation in Unregularized GANs

Most standard GAN architectures do not have the entropy regularization. Likelihood lower bounds of Theorem. 1 and Corollary. 2 hold even for those GANs as long as we obtain the optimal coupling $\mathbb{P}^*_{X,\hat{X}}$ in addition to the optimal generator $\mathbf{G}^*$ from GAN's training. Computation of optimal coupling $\mathbb{P}^*_{X,\hat{X}}$ from the dual formulation of OT GAN can be done when the loss function is quadratic [59].

For a general GAN architecture, however, the exact computation of optimal coupling $\mathbb{P}^*_{X,\hat{X}}$ may be difficult. One sensible approximation is to couple $X = \mathbf{x}^{\text{test}}$ with a single latent sample $\tilde{\mathbf{z}}$ (we are assuming the conditional distribution $\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}$ is an impulse function). To compute $\tilde{\mathbf{z}}$ corresponding to a $\mathbf{x}^{\text{test}}$, we sample k latent samples $\{\mathbf{z}'_i\}_{i=1}^k$ and select the $\mathbf{z}'_i$ whose $\mathbf{G}^*(\mathbf{z}'_i)$ is closest to $\mathbf{x}^{\text{test}}$. This heuristic takes into account both the likelihood of the latent variable as well as the distance between $\mathbf{x}^{\text{test}}$ and the model (similarly to Eq. (3.19)). We can then use Corollary 2



(a) (b)

Figure 3.3: (a) Sample likelihood estimates of MNIST, Office and CIFAR datasets using a GAN trained on the CIFAR dataset. (b) Sample likelihood estimates of MNIST, Office and LSUN datasets using a GAN trained on the LSUN dataset.

to approximate sample likelihoods for various GAN architectures.

We use this approach to compute likelihood estimates for CIFAR-10 [68] and LSUN-Bedrooms [69] datasets. For CIFAR-10, we train DCGAN while for LSUN, we train WGAN. Fig. 3.3(a) demonstrates sample likelihood estimates of different datasets using a GAN trained on CIFAR-10. Likelihoods assigned to samples from MNIST and Office datasets are lower than that of the CIFAR dataset. Samples from the Office dataset, however, are assigned to higher likelihood values than MNIST samples. We note that the Office dataset is indeed more similar to the CIFAR dataset than MNIST. A similar experiment has been repeated for LSUN-Bedrooms [69] dataset. We observe similar performance trends in this experiment (Fig. 3.3(b)).

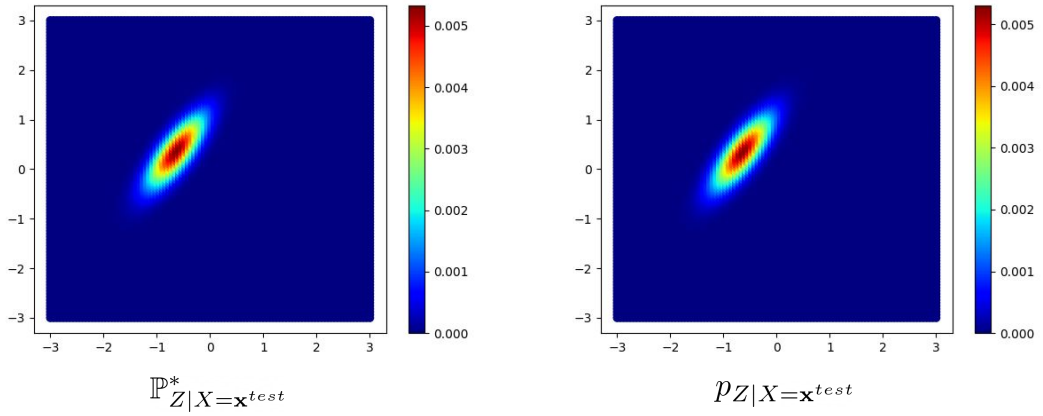


Figure 3.4: A visualization of density functions of $\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}$ and $p_{Z|X=\mathbf{x}^{\text{test}}}$ for a random two-dimensional $\mathbf{x}^{\text{test}}$. Both distributions are very similar to one another making the approximation gap (i.e. $\text{KL}\left(\mathbb{P}^*_{Z|X=\mathbf{x}^{\text{test}}}||p_{Z|X=\mathbf{x}^{\text{test}}}\right)$) very small. Our other experimental results presented in Table 3.1 are consistent with this result.

3.4.4 Tightness of the Variational Bound

In Theorem 1, we have shown that the Entropic GAN objective maximizes a lower-bound on the average sample log-likelihoods. This result has the same flavor as variational lower bounds used in VAEs, thus providing a connection between these two areas. One drawback of VAEs in general is the lack of tightness analysis of the employed variational lower bounds. In this section, we aim to understand the tightness of the entropic GAN's variational lower bound for some generative models.

3.4.4.1 Linear Generators

From corollary 2, we note that the entropic GAN lower bound is tight when $\mathrm{KL}\left(\mathbb{P}_{Z|X=\mathbf{x}}||p_{Z|X=\mathbf{x}}\right)$ approaches 0. Quantifying this term can be useful for assessing the quality of the proposed likelihood surrogate function. We refer to this term as the approximation gap.

Computing the approximation gap can be difficult in general as it requires evaluating $p_{Z|X=\mathbf{x}}$. Here we perform an experiment for linear generative models and a quadratic loss function (same setting of Corrolary 1). Let the real data X be generated from the following underlying model

$$p_{X|Z=\mathbf{z}} \sim \mathcal{N}(\mathbf{G}\mathbf{z}, \lambda\mathbf{I})$$

$$\text{where } Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Using the Bayes rule, we have,

$$p_{Z|X=\mathbf{x}^{\text{test}}} \sim \mathcal{N}(\mathbf{R}\mathbf{y}^{test}, \mathbf{I} - \mathbf{R}\mathbf{G})$$

$$\text{where } \mathbf{R} = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T + \lambda\mathbf{I})^{-1}.$$

Since we have a closed-form expression for $p_{Z|X}$, $\mathrm{KL}\left(\mathbb{P}_{Z|X=\mathbf{x}}||p_{Z|X=\mathbf{x}}\right)$ can be computed efficiently.

The matrix $\mathbf{G}$ to generate X is chosen randomly. Then, an entropic GAN with a linear generator and non-linear discriminators are trained on this dataset.

$\mathbb{P}_{Z|X=\mathbf{x}}$ is then computed using (3.22). Table 3.1 reports the average surrogate log-likelihood values and the average approximation gaps computed over 100 samples drawn from the underlying data distribution. We observe that the approximation gap is orders of magnitudes smaller than the log-likelihood values.

Additionally, in Figure 3.4, we demonstrate the density functions of $\mathbb{P}_{Z|X=\mathbf{x}}$ and $p_{Z|X=\mathbf{x}}$ for a random $\mathbf{x}$ and a two-dimensional case ($r = 2$) . In this figure, one can observe that both distributions are very similar to one another making the approximation gap very small.

Architecture and hyper-parameter details: For the generator network, we used 3 linear layers without any non-linearities ($2 \to 128 \to 128 \to 2$). Thus, it is an over-parameterized linear system. The discriminator architecture (both D_1 and D_2) is a 2-layer MLP with ReLU non-linearities ($2 \to 128 \to 128 \to 1$). $\lambda = 0.1$ was used in all the experiments. Both generator and discriminator were trained using the Adam optimizer with a learning rate 10^{-6} and momentum 0.5. The discriminators were trained for 10 steps per generator iteration. Batch size of 512 was used.

Table 3.1: The tightness of the entropic GAN lower bound. Approximation gaps are orders of magnitudes smaller than the surrogate log-likelihoods. Results are averaged over 100 samples drawn from the underlying data distribution.

Noise dimension	Approximation gap	Surrogate Log-Likelihood
2	9.3×10^{-4}	-4.15
5	4.7×10^{-2}	-15.35
10	6.2×10^{-2}	-46.3

3.4.4.2 Non-linear Generators

In this part, we consider the case of non-linear generators. The approximation gap $\mathrm{KL}\left(\mathbb{P}_{Z|X=\mathbf{x}}||p_{Z|X=\mathbf{x}}\right)$ cannot be computed efficiently for non-linear generators as computing the optimal coupling $\mathbb{P}_{Z|X=\mathbf{x}}$ is intractable. Instead, we demonstrate the tightness of the variational lower bound by comparing the exact data log-likelihood and the estimated lower-bound. As before, a $d-$dimensional Gaussian data distribution is used as the data distribution. The use of Gaussian distribution enables us to compute the exact data likelihood in closed-form. A table showing exact likelihood and the estimated lower-bound is shown in Table 3.2. We observe that the computed likelihood surrogate provides a good estimate to the exact data likelihood.

3.5 Conclusion

In this chapter, we have provided a statistical framework for a family of GANs. Our main result shows that the entropic GAN optimization can be viewed as maximization of a variational lower-bound on average sample log-likelihoods, an approach that VAEs are based upon. This result makes a connection between two most-popular generative models, namely GANs and VAEs. More importantly, our result

Table 3.2: The tightness of the entropic GAN lower bound for non-linear generators.

Noise dimension	Exact Log-Likelihood	Surrogate Log-Likelihood
5	-16.38	-17.94
10	-35.15	-43.6

constructs an explicit probability model for GANs that can be used to compute a lower-bound on sample likelihoods. Our experimental results on various datasets demonstrate that this likelihood surrogate can be a good approximation of the true likelihood function. Although in this chapter we mainly focus on understanding the behavior of the sample likelihood surrogate in different datasets, the proposed statistical framework of GANs can be used in various statistical inference applications. For example, our proposed likelihood surrogate can be used to quantitatively evaluate the performance of different GAN architectures, quantify domain shifts, select a proper generator class by balancing the bias term vs. variance, detect outlier samples, and can be used in statistical tests such as hypothesis testing, etc. We leave exploring these directions for future work.

Part II

Unsupervised Domain Adaptation

The objective of distributionally robust learning is to train models that perform reliably when the test data comes from a different distribution than the training dataset. Unsupervised domain adaptation comprises of the class of techniques to adapt models trained on one distribution (source) to a different distribution (target), with the goal of improving performance on the target distribution.

In general, performing well under any distributional shift is extremely challenging, and often infeasible [70]. So, the key challenge lies in imposing restrictive assumptions on the type of distributional shifts. One popular assumption is *covariate shift*, in which the conditional distribution of labels given inputs is assumed to be fixed across distributions, while the marginal distributions differ. Formally, let $p_{src}(\mathbf{x}, y)$ denote the source distribution and $p_{tgt}(\mathbf{x}, y)$ denote the target distribution. Then, under covariate shift assumption,

$$p_{src}(y|\mathbf{x}) = p_{tgt}(y|\mathbf{x})$$

$$p_{src}(\mathbf{x}) \neq p_{tgt}(\mathbf{x})$$

In this part of the dissertation, we develop algorithms for performing unsupervised domain adaptation under the covariate shift assumption. We focus on deep adaptation, in which we adapt the representations learnt by deep neural networks. In unsupervised domain adaptation, we are given access to a source dataset $\mathcal{D}_{src} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_s}$ and a target dataset $\mathcal{D}_{tgt} = \{\mathbf{x}_i\}_{i=1}^{n_t}$. Note that the source dataset is labeled while the target dataset is unlabeled. The objective is to learn models using the labeled source and unlabeled target dataset so that performance on the target

dataset improves.

In supervised deep learning, we train a model by minimizing the cross-entropy loss on the source dataset (Section. 2.1.1). Due to the presence of domain shift, a model trained solely on the source data performs sub-optimally on the target [71]. The absence of ground-truth labels on the target distribution prohibits training a model using cross-entropy loss on the target. The natural question that arises is how to utilize the unlabeled target distribution in addition to the labeled source to improve performance on the target. This forms the key challenge in unsupervised domain adaptation.

In this dissertation, we develop discriminative and generative approaches for unsupervised domain adaptation problem. The core idea is introduce loss functions that align the feature spaces of the source and target feature distribtions. In Chapter 4, we use a Generative Adversarial Network (GAN) to perform the feature space alignment. In Chapters 5 and 6, we develop computationally-efficient variants of optimal transport distances that can aid the feature space alignment. The developed algorithms are rigorously evaluated on some benchmark adaptation datasets.

Chapter 4: Generate to Adapt: Aligning Domains using Generative Adversarial Networks

4.1 Introduction

In this chapter, we provide an approach for learning a deep feature embedding that is robust to the domain shift between source and target distributions. We achieve this by using unsupervised data from the target distribution to guide the supervised learning procedure that uses data from the source distribution. We propose an adversarial image generation approach to directly learn the shared feature embedding using labeled data from source and unlabeled data from the target. It should be noted that while there have been a few approaches that use an adversarial framework for solving the domain adaptation problem [71, 72], the novelty of the proposed approach is in using a joint generative discriminative method: the embeddings are learned using a combination of classification loss and an image generation procedure that is modeled using a variant of Generative Adversarial Networks (GANs) [34].

Figure 4.1 illustrates the pipeline of the proposed approach. During training, the source images are passed through the feature extraction network (encoder) to

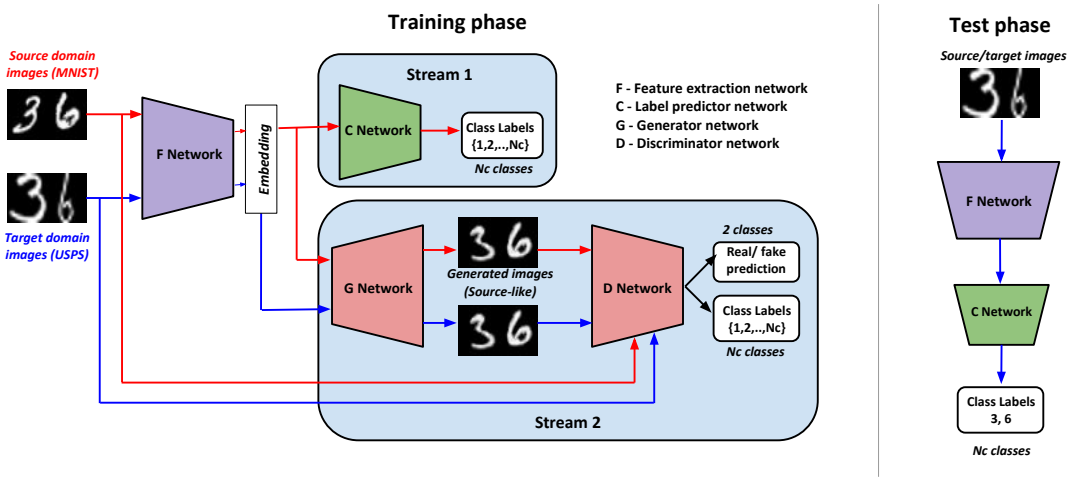


Figure 4.1: Illustration of the proposed approach. In the training phase, our pipeline consists of two parallel streams - (1) Stream 1: classification branch where F-C networks are updated using supervised classification loss and (2) Stream 2: adversarial branch which is a Auxiliary Classifier GAN (ACGAN) framework (G-D pair). F-G-D networks are updated so that both source and target embeddings produce source-like images. Note: The auxiliary classifier in ACGAN uses only the source domain labels, and is needed to ensure that class-consistent images are generated (e.g) embedding of digit 3 generates an image that looks like 3. In the test phase, we remove Stream 2, and classification is performed using the F-C pair.

obtain an embedding which is then used by the label prediction network (classifier) for predicting the source label and also used by the generator to generate a realistic source image. The realistic nature of the images from the generator (G) is controlled by the discriminator (D). The encoder is updated based on the discriminative gradients from the classifier and generative gradients from the adversarial framework. Given unlabeled target images, the encoder is updated using only gradients from the adversarial part, since the labels are unavailable. Thus, the encoder learns to discriminate better even in the target domain using the knowledge imparted by the generator-discriminator pair. By using the discriminator as a multi-class classifier, we ensure that the gradient signals backpropagated by the discriminator for the unlabeled target images belong to the feature space of the respective classes. By sampling from the distribution of the generator after training, we show that the network has indeed learned to bring the source and target distributions closer.

The main contribution of this work is to provide an adversarial image generation approach for unsupervised domain adaptation that directly learns a joint feature space in which the distance between source and target distributions is minimized. Different from contemporary approaches that achieve a similar objective by using a GAN as a data augmenter, our approach achieves superior results even in cases where a standalone image generation process is bound to fail (such as in the OFFICE dataset). This is done by utilizing the GAN framework to address the domain shift directly in the feature space learnt by the encoder. Our experiments show that the proposed approach yields superior results compared to similar approaches which update the embedding based on auto-encoders [73] or disentangling

the domain information from the embedding by learning a separate domain classifier [71].

4.2 Related Work

Earlier approaches to domain adaptation focused on building invariant feature representations using feature re-weighting and selection mechanisms [74, 75], or by learning an explicit manifold-based feature transformations that aligns source distribution to the target [2, 76, 77]. The ability of deep neural networks to learn powerful representations [5, 29] has been harnessed to perform unsupervised domain adaptation in recent works [71, 72, 78, 79, 80]. The underlying idea behind such methods is to minimize a suitable loss function that captures domain discrepancy, in addition to the task being solved.

Deep learning methods for visual domain adaptation can be broadly grouped into few major categories. One line of work uses Maximum Mean Discrepancy(MMD) as a metric to measure the shift across domains. Deep Domain Confusion (DDC) [78] jointly minimizes the classification loss and MMD loss of the last fully connected layer. Deep Adaptation Networks (DAN) [79] extends this idea by embedding all task specific layers in a reproducing kernel Hilbert space and minimizing the MMD in the projected space. In addition to MMD, Residual Transfer Networks (RTN) [80] uses a gated residual layer for classifier adaptation. Joint Adaptation Networks [81] learn a transfer network by aligning the joint distributions of multiple domain-specific layers across domains based on a Joint Maximum Mean Discrepancy

(JMMD) criterion.

Another class of methods uses adversarial losses to perform domain adaptation. [71] employs a domain classification network which aims to discriminate the source and the target embeddings. The goal of the feature extraction network is to produce embeddings that maximize the domain classifier loss, while at the same time minimizing the label prediction loss. This is accomplished by negating the gradients coming from the domain classification network. Adversarial Discriminative Domain Adaptation (ADDA) [72], on the other hand, learns separate feature extraction networks for source and target domains, and trains the target CNN so that a domain classifier cannot distinguish the embeddings produced by the source or target CNNs.

While methods discussed above apply adversarial losses in the embedding space, there has been a lot of interest recently to perform adaptation in the pixel space. Such approaches primarily use generative models such as GANs to perform cross-domain image mapping. Taigman et al. [82] and Bousmalis et al. [83] use adversarial networks to map source images to target and perform adaptation in the transferred space. Coupled GAN (CoGAN) [84], on the other hand, trains a coupled generative model that learns the joint data distribution across the two domains. A domain invariant classifier is learnt by sharing weights with the discriminator of the CoGAN network.

Comparison to other GAN-based DA approaches: While previous approaches such as [82] and [83] use GANs as a data augmentation step, we use a GAN to obtain rich gradient information that makes the learned embeddings do-

main adaptive. Unlike the previous methods, our approach does not completely rely on a successful image generation process. As a result, our method works well in cases where image generation is hard (eg. in the OFFICE dataset where the number of samples per class is limited). We observed that in such cases, the generator network we use performs a mere style transfer, yet this is sufficient for providing good gradient information for successfully aligning the domains, as demonstrated by our superior performance on the OFFICE dataset.

4.3 Approach

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output space. Since, we consider the classification problem, the label space is discrete with N_c labels i.e., $\{1, 2, \dots n_c\}$. Let $\mathcal{D}_{src} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_s}$ and $\mathcal{D}_{tgt} = \{\mathbf{x}_i\}_{i=1}^{n_t}$ denote the source and target datasets, respectively. Here, each $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The objective is to train a deep feature network $F_{\theta_f} : \mathcal{X} \to \mathbb{R}^d$ that maps images to an embedding space, and a classifier $C_{\theta_c} : \mathbb{R}^d \to \mathcal{Y}$. In domain adaptation, we are interested in improving the performance of the model on the unlabeled dataset $\mathcal{D}_{tgt}$.

Several approaches including learning entropy-based metrics [80], learning a domain classifier based on a embedding network [71] or denoising autoencoders [73] have been used to transfer information between source and target distributions. In this work, we propose a GAN-based approach to bridge the gap between source and target domains. We accomplish this by using both generative and a discriminative processes thus ensuring a rich information transfer to the learnt embedding.

We use a variant of GANs, called conditional GANs [85] for modeling the image distribution. Conditional GANs enable conditioning the generator and discriminator mappings on additional data such as a class label or an embedding. They have been shown to generate images of digits and faces corresponding to a given class label or the embedding, respectively [82]. Training a conditional GAN involves optimizing the following minimax objective:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}} \left[\log(D(\mathbf{x}|y)\right] + \mathbb{E}_{z\sim p_z} \left[\log(1 - D(G(\mathbf{z}|y)))\right]$$

Here, y is the conditioning variable.

Proposed Approach: In this work, we employ a variant of the conditional GANs, called Auxiliary Classifier GAN (AC-GAN) [86] where the discriminator is modeled as a multi-class classifier instead of providing conditioning information at the input. We modify the AC-GAN set up for the domain adaptation problem as follows:

(a) Given a real image $\mathbf{x}$ as input, the input to the generator network G is $\mathbf{e} = [F(\mathbf{x}), \mathbf{z}, \mathbf{y}]$, which is a concatenated version of the encoder embedding $F(\mathbf{x})$, a random noise vector $\mathbf{z} \in \mathbb{R}^d$ sampled from $\mathcal{N}(0, 1)$ and a one hot encoding of the class label $y \in \{0, 1\}^{(n_c+1)}$. Here, n_c is the number of real classes in the dataset, and the label $\{n_c + 1\}$ is treated as the fake class. For all target samples, since the class labels are unknown, $\mathbf{y}$ is set as the one hot encoding of the fake class $\{n_c + 1\}$.

(b) We employ a classifier network C that takes as input the embedding generated by F and predicts a multiclass distribution $C(F(\mathbf{x}))$ - the class probability

distribution of the input $\mathbf{x}$, which is modeled as a (n_c)-way classifier.

(c) The discriminator D takes the real image $\mathbf{x}$ or the generated image $G(\mathbf{x}_g)$ as input and outputs two distributions: (1) $D_{gan}(\mathbf{x})$: the probability of the input being real, which is modeled as a binary classifier. (2) $D_{cls}(\mathbf{x})$: the class probability distribution of the input $\mathbf{x}$, which is modeled as a (n_c)-way classifier. We use $D_{cls}(\mathbf{x})_y$ to denote the probability assigned by the classifier mapping D_{cls} for input $\mathbf{x}$ to class y. It should be noted that, for target data, since class labels are unknown, only D_{gan} is used to backpropagate the gradients.

Now, we describe our optimization procedure in detail. To jointly learn the embedding and the generator-discriminator pair, we optimize the D, G, F and C networks in an alternating manner:

1. Given source images as input, D outputs two distributions D_{gan} and D_{cls}. D_{gan} is optimized by minimizing a binary cross entropy loss $L_{gan,src}$, while D_{cls} is optimized by minimizing the cross entropy loss $L_{cls,src}$. In the case of source inputs, the gradients are generated using the following loss functions:

$$\max_{D} \ \mathcal{L}_{gan,src} + \mathcal{L}_{cls,src} = \ \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{src}}\Big[\log(D_{gan}(\mathbf{x})) + \log(1 - D_{gan}(G(\mathbf{e}))) + \log(D_{cls}(\mathbf{x})_y)\Big]$$

2. Using the gradients from D, the generator G is updated using a combination of adversarial loss and classification loss to produce realistic class consistent

Algorithm 2 Iterative training procedure of our approach

Require: Number of training iterations = n_{iter}

1: **for** t in $1 : N$ **do**

2: Sample k images with labels from source dataset $\mathcal{D}_{src}$: $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^k$.

3: Sample k unlabeled images from target dataset $\mathcal{D}_{tgt}$: $\{\mathbf{x}_i^t\}_{i=1}^k$

4: Sample $2k$ random noise vectors $\{\mathbf{z}_i^s\}_{i=1}^k$ and $\{\mathbf{z}_i^t\}_{i=1}^k \sim \mathcal{N}(0, 1)$.

5: Let $\mathbf{e}_i^s = [F(\mathbf{x}_i^s, \mathbf{z}_i^s, y_i)]$ and $\mathbf{e}_i^t = [F(\mathbf{x}_i^t, \mathbf{z}_i^t, (n_c + 1))]$ be the concatenated embeddings to the generator.

6: Update the discriminator using the following loss function:

$$\max_D \ \mathcal{L}_{disc} := \mathcal{L}_{gan,src} + \mathcal{L}_{cls,src} + \mathcal{L}_{gan,tgt}$$

- $\mathcal{L}_{gan,src} = \frac{1}{k}\sum_{i=1}^k \log(D_{gan}(\mathbf{x}_i^s)) + log(1 - D_{gan}(G(\mathbf{e}_i^s)))$
- $\mathcal{L}_{cls,src} = \frac{1}{k}\sum_{i=1}^k \log(D_{cls}(\mathbf{x}_i^s)_{y_i^s})$
- $\mathcal{L}_{gan,tgt} = \frac{1}{k}\sum_{i=1}^k \log(1 - D_{gan}(G(\mathbf{e}_i^t)))$

7: Update the generator, only for source data, through the discriminator gradients computed using real labels.

$$\min_G \ \mathcal{L}_{gen} = \frac{1}{k}\sum_{i=1}^{k} -\log(D_{cls}(G(\mathbf{e}_i^s))_{y_i^s}) + \log(1 - D_{gan}(G(\mathbf{e}_i^s))) \tag{4.1}$$

8: Update the feature network F using a linear combination of the adversarial loss and classification loss, while update the classifier C only using the classification loss.

$$\min_F \ \min_C \ \mathcal{L}_f = \mathcal{L}_{cls} + \alpha\mathcal{L}_{disc,cls} + \beta\,\mathcal{L}_{adv} \tag{4.2}$$

- $\mathcal{L}_{cls} = \frac{1}{k}\sum_{i=1}^k -\log(C(F(\mathbf{x}_i^s))_{y_i^s})$
- $\mathcal{L}_{disc,cls} = \frac{1}{k}\sum_{i=1}^k -\log(D_{cls}(G(\mathbf{e}_i^s))_{y_i^s})$
- $L_{adv} = \frac{1}{k}\sum_{i=1}^k \log(1 - D_{gan}(G(\mathbf{e}_i^t)))$

9: **end for**

source images.

$$\min_{G} \ \mathcal{L}_{gen} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{src}} \Big[-\log(D_{cls}(G(\mathbf{e}))_y) + \log(1 - D_{gan}(G(\mathbf{e}))) \Big] \quad (4.3)$$

3. F and C are updated based on the source images and source labels in a traditional supervised manner. F is also updated using the adversarial gradients from D so that the feature learning and image generation processes co-occur smoothly.

$$\min_{F} \ \min_{C} \ \mathcal{L}_{cls} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{src}} \left[-\log(C(F(\mathbf{x}))_y) \right],$$
$$\min_{F} L_{disc,cls} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{src}} \Big[-\alpha \ \log(D_{cls}(G(\mathbf{e}))_y) \Big]$$

4. In the final step, the real target images are presented as input to F. The target embeddings output by F along with the random noise vector $\mathbf{z}$ and the fake label encoding y are input to G. The generated target images $G(\mathbf{e})$ are then given as input to D. As described above, D outputs two distributions but the loss function is evaluated only for D_{gan} since in the unsupervised case considered here, target labels are not provided during training. Hence, D is updated to classify the generated target images as fake as follows:

$$\max_{D} \ \mathcal{L}_{gan,tgt} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{tgt}} \left[\log(1 - D_{gan}(G(\mathbf{e}))) \right]$$

In order to transfer the knowledge of target distribution to the embedding, F is updated using the gradients from D_{gan} that corresponds to the generated target images being classified as real:

$$\min_{F} \mathcal{L}_{adv} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{tgt}} \left[\beta \log(1 - D_{gan}(G(\mathbf{e})))\right]$$

The proposed iterative optimization procedure is summarized as a pseudocode in Algorithm 2. α and β are the coefficients that trade off between the classification loss and the source and target adversarial losses. Based on our experiments, we find that our approach is not overly sensitive to the cost coefficients α and β. However, the value of the parameter is dependent on the application and size of the dataset. Such specifications are mentioned in the Sankaranarayanan et al. [87].

Use of unlabeled target data: The main strength of our approach is how the target images are used to update the embedding. Given a batch of target images as input, we update the embedding F by using the following binary loss term:

$$\min_{F} \beta \log(1 - D_{gan}(G(\mathbf{e}))) \tag{4.4}$$

where $\mathbf{e}$ is the concatenated input to G as described earlier and β is the weight coefficient for the target adversarial loss. The use of target data is intended to bring the source and target distributions closer in the feature space learned by F. To achieve this, we update the F network to produce class consistent embed-

dings for both source and target data. Performing this update for source data is straightforward since the source labels are available during training. Since labels are unavailable for target data, we use the generative ability of the G-D pair for obtaining the required gradients.

Given source inputs, G is updated to fool D using gradients from Eq. (4.3) which provide the conditioning required for G to produce class consistent fake images. Given target inputs, the update in Eq. (4.4) encourages F to produce target embeddings that are aligned with the source distribution. As training progresses, the class conditioning information learned by G during the source update (Eq. (4.3)) was found to be sufficient for it to produce class consistent images for target embeddings as well. This symbiotic relationship between the embedding and the adversarial framework contributes to the success of the proposed approach.

4.4 Experiments and Results

In this section, we perform a thorough experimental study by conducting experiments across three domain adaptation settings: (1) low domain shift and simple data distribution: DIGITS dataset, (2) moderate domain shift and complex data distribution: OFFICE dataset, (3) high domain shift and complex data distribution: Synthetic to real adaptation. By complex data distribution, we denote datasets containing images with high variability and limited number of samples. Our methods performs well in all three regimes, thus demonstrating the versatility of our approach.

[0]Code available at: `https://goo.gl/zUVeqC`

4.4.1 Digit Experiments

The first set of experiments involve digit classification in three standard DIGITS datasets: MNIST [88], USPS [89] and SVHN [90]. Each dataset contains digits belonging to 10 classes (0-9). MNIST and USPS are large datasets of handwritten

Table 4.1: Accuracy (mean ± std%) values for cross-domain recognition tasks over five independent runs on the digits based datasets. The best numbers are indicated in **bold** and the second best are underlined. − denotes unreported results. MN: MNIST, US: USPS, SV: SVHN. MN→US (p) denotes the MN→US experiment run using the protocol established in [1], while MN→US (f) denotes the experiment run using the entire datasets. (Refer to Digits experiments section for more details)

Method	MN → US (p)	MN → US (f)	US → MN	SV → MN
Source only	75.2 ± 1.6	79.1 ± 0.9	57.1 ± 1.7	60.3 ± 1.5
RevGrad [71]	77.1 ± 1.8	-	73.0 ± 2.0	73.9
DRCN [73]	91.8 ± 0.09	-	73.7 ± 0.04	82.0 ± 0.16
CoGAN [84]	91.2 ± 0.8	-	89.1 ± 0.8	-
ADDA [72]	89.4 ± 0.2	-	90.1 ± 0.8	76.0 ± 1.8
PixelDA [83]	-	**95.9**	-	-
Ours	**92.8 ± 0.9**	95.3 ± 0.7	**90.8** ± 1.3	**92.4** ± 0.9

Table 4.2: Accuracy (mean ± std%) values on the OFFICE dataset for the standard protocol for unsupervised domain adaptation [2]. Results are reported as an average over 5 independent runs. The best numbers are indicated in **bold** and the second best are underlined. − denotes unreported results. We use Resnet-50 model in our experiments. A: Amazon, W: Webcam, D: DSLR

Method	A → W	D → W	W → D	A → D	D → A	W → A	Average
Source only [5]	68.4 ± 0.2	96.7 ± 0.1	99.3 ± 0.1	68.9 ± 0.2	62.5 ± 0.3	60.7 ± 0.3	76.1
TCA [77]	72.7 ± 0.0	96.7 ± 0.0	99.6 ± 0.0	74.1 ± 0.0	61.7 ± 0.0	60.9 ± 0.0	77.6
GFK [2]	72.8 ± 0.0	95.0 ± 0.0	98.2 ± 0.0	74.5 ± 0.0	63.4 ± 0.0	61.0 ± 0.0	77.5
DDC [78]	75.6 ± 0.2	76.0± 0.2	98.2 ± 0.1	76.5 ± 0.3	62.2 ± 0.4	61.5 ± 0.5	78.3
DAN [79]	80.5 ± 0.4	97.1 ± 0.2	99.6 ± 0.1	78.6 ± 0.2	63.6 ± 0.3	62.8 ± 0.2	80.4
RTN [80]	84.5 ± 0.2	96.8 ± 0.1	99.4 ± 0.1	77.5 ± 0.3	66.2 ± 0.2	64.8 ± 0.3	81.6
RevGrad [71]	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.4 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
JAN [81]	85.4 ± 0.3	97.4 ± 0.2	**99.8** ± 0.2	84.7 ± 0.3	68.6 ± 0.3	70.0 ± 0.4	84.3
Ours	**89.5** ± 0.5	**97.9** ± 0.3	**99.8** ± 0.4	**87.7** ± 0.5	**72.8** ± 0.3	**71.4** ± 0.4	**86.5**

digits captured under constrained conditions. SVHN dataset, on the other hand was obtained by cropping house numbers in Google Street View images and hence captures much more diversity. We test the three common domain adaptation settings: SVHN $\rightarrow$ MNIST, MNIST $\rightarrow$ USPS and USPS $\rightarrow$ MNIST. In each setting, we use the label information only from the source domain, thus following the unsupervised protocol.

For all digit experiments, following other recent works [71][72], we use a modified version of Lenet architecture as our F network. For G and D networks, we use architectures similar to those used in DCGAN [91].

(a) MNIST $\leftrightarrow$ USPS

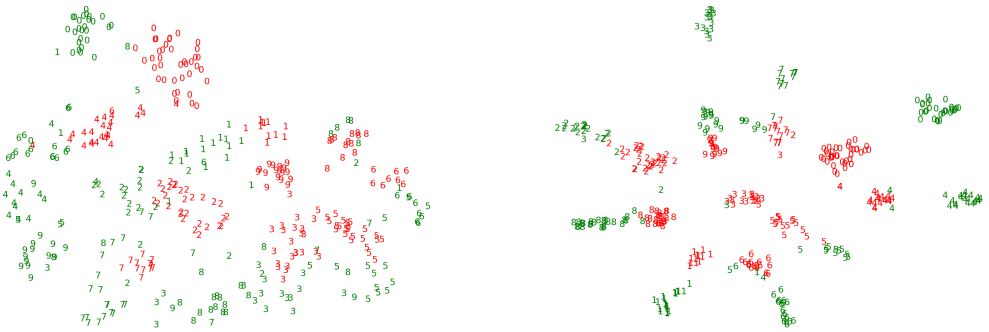
We start with the easy case of adaptation involving MNIST and USPS. The MNIST dataset is split into 60000 training and 10000 test images, while the USPS dataset contains 7291 training and 2007 test images. We run our experiments in two settings: (1) using the entire training set of MNIST and USPS (MNIST $\leftrightarrow$USPS (f)), and (2) using the protocol established in [1], sampling 2000 images from MNIST and 1800 images from USPS (MNIST $\leftrightarrow$USPS (p)). Table. 4.1 presents the results of the proposed approach in comparison with other contemporary approaches. The reported numbers are averaged over 5 independent runs with different random samplings or initializations. We can observe that our approach achieves the best performance in all cases except in the MNIST $\rightarrow$ USPS full protocol case where our accuracy is very close to the best performing method.

(b) SVHN → MNIST

Compared to the previous experiment, SVHN → MNIST presents a harder case of domain adaptation owing to larger domain gap. Following other works [71] [72], we use the entire training set (labeled 73257 SVHN images and unlabeled 60000 MNIST images) to train our model, and evaluate on the training set of the target domain (MNIST dataset). From Table. 4.2, we observe that our method significantly improves the performance of the source-only model from 60.3% to 92.4%, which results in a performance gain of 32.1%. We also outperform other methods by a large margin, obtaining at least 10.4% performance improvement. A visualization of this improvement in performance is done in figure 4.2, where we show a t-SNE plot of the features of the embedding network F for the adapted and non-adapted cases.

4.4.2 OFFICE experiments

The next set of experiments involve the OFFICE dataset, which is a small scale dataset containing images belonging to 31 classes from three domains - Amazon, Webcam and DSLR, each containing 2817, 795 and 498 images respectively. The small dataset size poses a challenge to our approach since we rely on GAN which demands more data for better image generation. Nevertheless, we perform experiments on the OFFICE dataset to demonstrate that though our method does not succeed in generating very realistic images, the approach still results in improved performance by using the generative process to obtain domain invariant feature representations.



(a) Non adapted (b) Adapted

Figure 4.2: TSNE visualization of SVHN → MNIST adaptation. In (a), the source data shown in *red* is classified well into distinct clusters but the target data is clustered poorly. On applying the proposed approach, as shown in (b), both the source and target distributions are brought closer in a class consistent manner.

Training deep networks with randomly initialized weights on small datasets give poor performance. So, an effective technique used in practice is to fine-tune networks trained on a related task having large data [92]. Following this rationale, we initialized the F network using a pre-trained ResNet-50 [5] model trained on Imagenet. For D and G networks, we used architectures similar to the ones used in the Digits experiments. It should be noted that even though the inputs are 224×224, the G network is made to generate a downsampled version of size 64×64. Standard data augmentation steps involving mean normalization, random cropping and mirroring were performed.

In all our experiments, we follow the standard unsupervised protocol - using the entire labeled data in the source domain and unlabeled data in the target domain. Table 4.2 reports the performance of our method in comparison to other methods.

We observe that our method obtains the state-of-the-art performance in all the settings. In particular, we get good performance improvement consistently in all hard transfer cases: $A \rightarrow W$, $A \rightarrow D$, $W \rightarrow A$ and $D \rightarrow A$.

4.4.3 Synthetic to Real experiments

To test the effectiveness of the proposed approach further, we perform experiments in the hardest case of domain adaptation involving adaptation from synthetic to real datasets. This setting is particularly interesting because of its enormous practical implications. In this experiment, we use CAD synthetic dataset [93] and a subset of PASCAL VOC dataset [94] as our source and target sets respectively. The CAD synthetic dataset contains multiple renderings of 3D CAD models of the 20 object categories contained in the PASCAL dataset. To create the datasets, we follow the protocol described in [95]: The CAD dataset contains six subsets with different configurations (i.e. RR-RR, W-RR, W-UG, RR-UG, RG-UG, RG-RR). Of these, we use images with white background (W-UG subset) as our training set. To generate the target set, we crop 14976 patches from 4952 images of the PASCAL VOC 2007 test set using the object bounding boxes provided. The lack of realistic background and texture in the CAD synthetic dataset increases the disparity from the natural image manifold, thus making domain adaptation extremely challenging.

Due to the high domain gap, we observed that models trained on the CAD synthetic dataset with randomly initialized weights performed very poorly on the target dataset. So, similar to the previous set of experiments, we initialized the F

Table 4.3: Accuracy (mean ± std%) values over five independent runs on the Synthetic to real setting. The best numbers are indicated in **bold**.

Method	CAD → PASCAL
VGGNet - Source only	38.1 ± 0.4
RevGrad [71]	48.3 ± 0.7
RTN [80]	43.2 ± 0.5
JAN [81]	46.4 ± 0.8
Ours	**50.4** ± 0.6

network with pretrained models. In particular, we removed the last fully connected layer from the VGG16 model trained on Imagenet and used it as our F network. Note that the same F network is used to train all other methods for fair comparison. Table. 4.3 reports the results of the experiments we ran on the synthetic to real setting. We can observe that our method improves the baseline performance from 38.1% to 50.4% in addition to outperforming all other compared methods.

4.4.4 VISDA challenge

In this section, we present the results on VISDA dataset [95] - a large scale testbed for unsupervised domain adaptation algorithms. The task is to train classification models on synthetic dataset generated from the renderings of 3D CAD models and adapt these models to real images which are drawn from Microsoft COCO [96](validation set) and Youtube Bounding Box dataset [97](test set). We train our models using the same hyper-parameter settings and data augmentation scheme as the previous experiment. Table. 4.4 presents the results on the VISDA classification challenge. We find that our method achieves significant performance gains compared to the baseline model.

Table 4.4: Performance (accuracy) of our approach on VISDA classification dataset.

Model	Visda-C: *Val*		
	Source-only	Adapted	Gain
Resnet-18	35.3	63.1	78.7%
Resnet-50	40.2	69.5	72.8%
Resnet-152	44.5	77.1	73.2%
	Visda-C: *Test*		
Resnet-152	40.9	72.3	76.7%

4.4.5 Ablation Study

4.4.5.1 Loss Function Analysis

In this experiment, we study the effect of each individual component in out method to the overall performance. The embedding network F is updated using a combination of losses from two streams (1) supervised classification stream and (2) adversarial stream, as shown in Figure 4.1. The adversarial stream consists of the G-D pair, with D containing two components - real/fake classifier which we denote as C_1, and auxiliary classifier which we denote as C_2. We report the performance on the following three settings: (1) using only the Stream 1 and only using source data to train - this corresponds to the Source-only setting (2) Using stream 1 + C_1 classifier from stream 2 - this corresponds to the case where source and target embeddings are forced to produce source-like images, but class information is not provided to the discriminator and (3) Using stream 1 + stream2 (C_1 + C_2) - this is our entire system. For settings (2) and (3) we utilized labeled source data and unlabeled target

Table 4.5: Ablation study for OFFICE A→W setting.

Setting	Accuracy(in %)
Stream 1 - Source only	68.4
Stream 1 + Stream 2 (C_1 only)	80.5
Stream 1 + Stream 2 ($C_1 + C_2$)	89.5

data during training. Table 4.5 presents the results of this experiment.

We observe that using only the real/fake classifier C_1 in the discriminator does improve performance, but the auxiliary classifier C_2 is needed to get the full performance benefit. This can be attributed to the mode collapse problem in traditional GANs (we observed that training without C_2 resulted in missing modes and mismatched mappings where embeddings get mapped to images of wrong classes), hence resulting in sub-optimal performance. Use of an auxiliary classifier objective in D stabilizes the GAN training as observed in [86] and significantly improves the performance of our approach.

4.4.5.2 Noise Analysis

The input to the generator network G is $\mathbf{e} = [F(\mathbf{x}), \mathbf{z}, \mathbf{y}]$, a concatenated version of the feature embedding, noise vector $\mathbf{z} \in \mathbb{R}^d$ sampled from $\mathcal{N}(0, 1)$ and $\mathbf{y}$, the one-hot encoding of the class label. In this section, we perform a study of how the dimensionality of the noise vector $\mathbf{z}$ affects the transfer accuracy. In figure 4.3, the transfer accuracy for the task SVHN → MNIST is plotted against the number of training epochs. The dimensionality d is varied over the set: $\{32, 64, 128, 256, 512\}$. The following observations can be made: (1) The approach is not overly sensitive

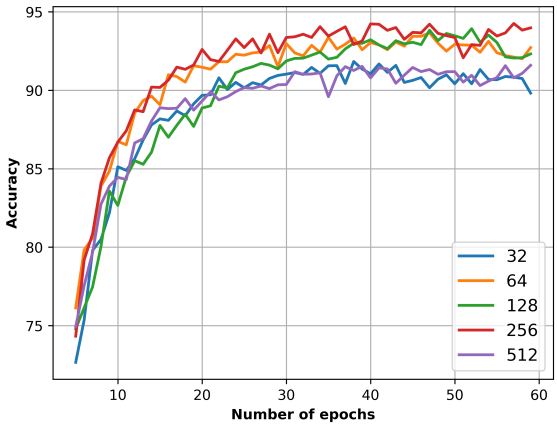


Figure 4.3: Effect of the noise dimension on classification accuracy for the transfer task SVHN → MNIST.

to d since all values of d obtain an average performance of 90.5% or more. (2) The values of dimensionality that is too low (32) or too high (512) result in slightly suboptimal performance.

4.4.5.3 Generation visualization

In Fig. 4.4, we show some sample images generated by the G network in two experimental settings - $SVHN \rightarrow MNIST$ and Office $A \rightarrow W$. The top set of images show the generations when the input to the system are the samples taken from the source dataset, while the bottom set are the generations when inputs are the images from the target dataset. We make the following observations: (1) The quality of image generation is better in the digits experiments compared to the Office experiments (2) The generator is able to produce source-like images for both the source and target inputs in a class-consistent manner (3) There is mode collapse in

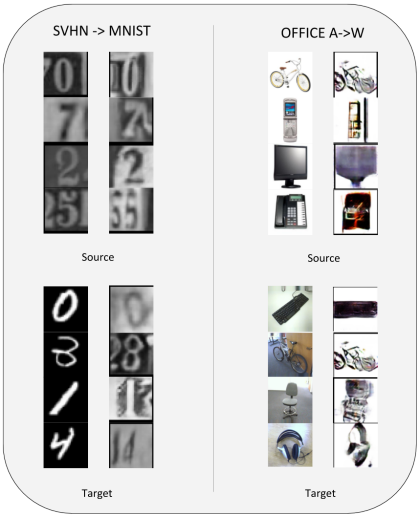


Figure 4.4: Example of images sampled from G after training. In each set, the images on the left indicate the source images and the images on the right indicate the generated images.

the generations produced in the Office experiments.

The difficulty of GANs in generating realistic images in the Office and synthetic to real datasets makes it significantly hard for the methods that use cross-domain image generation as a data augmentation step. Since we rely on the image generation as a mode for deriving rich gradients to the feature extraction network, our method works well even in the presence of severe mode collapse and poor generation quality.

4.4.6 Network Architectures and Hyperparameters

This section describes the details of the network architectures used in our experiments. A detailed description of all the architectures can be found in Fig. 4.5

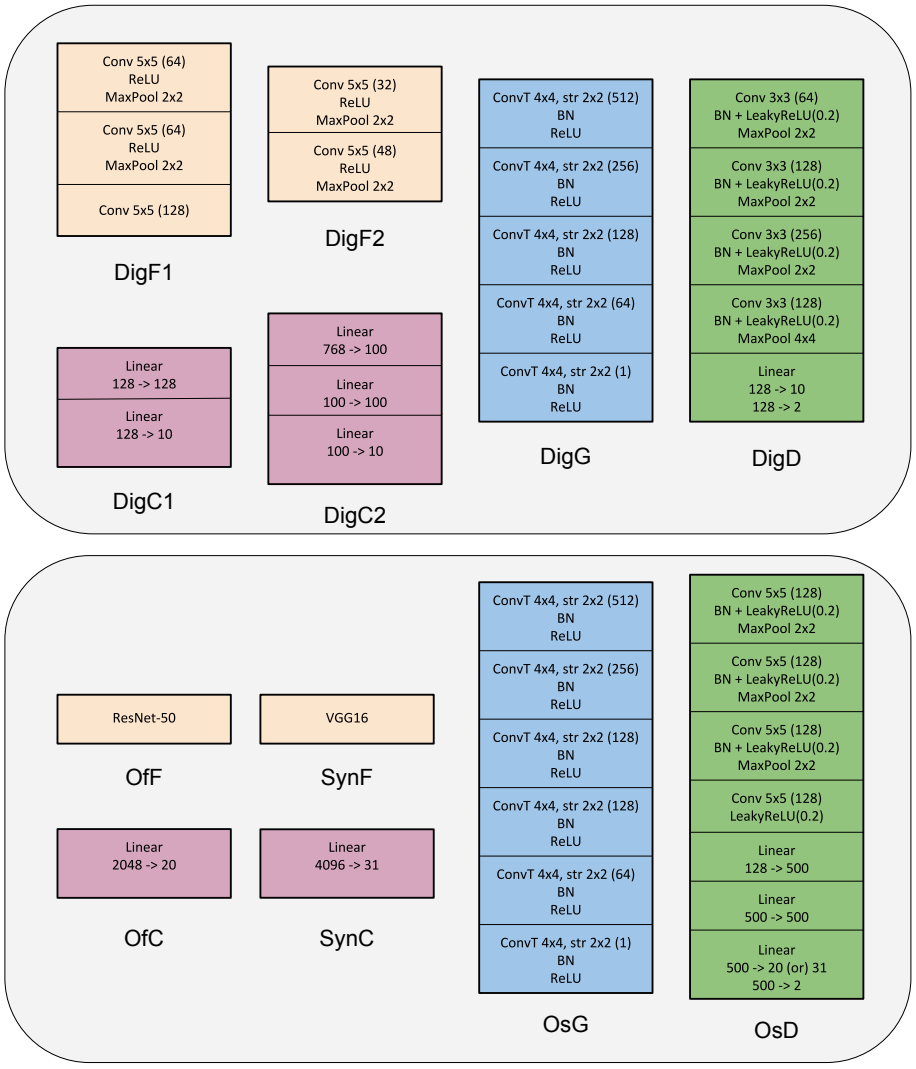


Figure 4.5: Network Architectures. Legend: BN - Batch Normalization, ConvT - Transposed convolution layer.

Digits experiments: For $SVHN \rightarrow MNIST$ experiment, we used $DigF1$, $DigC1$, $DigG$ and $DigD$ architectures mentioned in Fig. 4.5 as our F, C, G and D networks respectively. For all other digit experiments, we use $DigF2$, $DigC2$, $DigG$ and $DigD$. All models were trained from scratch and were initialized using random Gaussian noise with standard deviation 0.01. We used Adam solver with base learning rate of 0.0005 and momentum 0.8 to train our models. The cost coefficients α and β are set as 0.1 and 0.03 respectively based on validation splits. We resize all input images to 32×32 and scale their values to the range $[0, 1]$.

OFFICE experiments: For OFFICE experiments, we used $OfcC$, OsG and OsD architectures mentioned in Fig. 4.5 as our C, G and D networks respectively. The F network is initialized with pretrained Resnet50 model trained on ImageNet, the last layer of which is removed and the resulting 2048 dimensional vector is used as the feature embedding. We use Adam solver for optimization with a base learning rate of 0.0004 and momentum 0.7 for all the experiments. The dimension of the random noise vector is set as 128 and the cost coefficient α and β are both set as 0.01.

Synthetic to Real experiments: Similar to OFFICE experiments, we used $SynC$, OsG and OsD architectures mentioned in Fig. 4.5 as our C, G and D networks respectively. We remove the last layer of the pretrained VGG16 model trained on Imagenet, and initialize it as our F network. The resulting 4096 dimensional vector is used as the feature embedding. For all the experiments, we used the

same hyperparameter settings as those used in the Office experiments.

4.5 Conclusion and Future Work

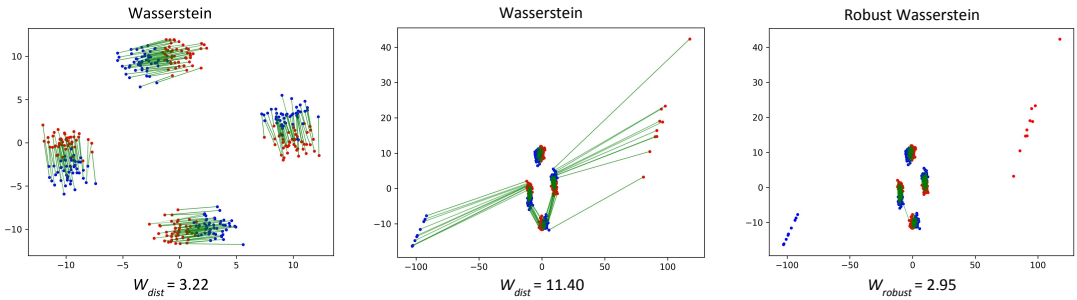
In this chapter, we addressed the problem of unsupervised visual domain adaptation using a joint adversarial-discriminative approach that transfers the information of the target distribution to the learned embedding using a generator-discriminator pair. We demonstrated the superiority of our approach over existing methods that address this problem using experiments on three different tasks, thus making our approach more generally applicable and versatile. Some avenues for future work include using stronger encoder architectures and applications of our approach to more challenging domain adaptation problems such as RGB-D object recognition and medical imaging.

Chapter 5: Robust Optimal Transport

5.1 Introduction

Estimating distances between probability distributions lies at the heart of several problems in machine learning and statistics. A class of distance measures that has gained immense popularity in several machine learning applications is Optimal Transport (OT) [49]. In OT, the distance between two probability distributions is computed as the minimum cost of transporting a source distribution to the target distribution under some transportation cost function. Optimal transport enjoys several nice properties including structure preservation, existence in smooth and non-smooth settings, being well defined for discrete and continuous distributions [49], etc.

As seen in Chapter 4, one of the fundamental issues in domain adaptation is the existence of feature space drift between source and target representations. Optimal transport can be used for minimizing this feature-space drift. In Wasserstein distance based domain adaptation, Wasserstein distance between source and target feature distributions are minimized while training a classifier on source domain using cross-entropy loss [98]. Similar ideas involving distance minimizaton between real and generated image distributions can also be used for training a GAN [36].



(a) Clean dataset (b) Dataset with outliers (c) Dataset with outliers

Figure 5.1: Visualizing couplings of Wasserstein distance computation between two distributions shown in red and blue. In (a), we show the couplings when no outliers are present. In (b), we show the couplings when 5% outliers are added to the data. The Wasserstein distance increases significantly indicating high sensitivity to outliers. In (c), we show the couplings produced by the Robust Wasserstein measure. Our formulation effectively ignores the outliers yielding a Wasserstein estimate that closely approximates the true Wasserstein distance.

One of the fundamental shortcomings of optimal transport is its sensitivity to outlier samples. By outliers, we mean samples with large noise. In OT optimization, to satisfy the marginal constraints between the two input distributions, every sample is weighed equally in the feasible transportation plans. Hence, even a few outlier samples can contribute significantly to the OT objective. This leads to poor estimation of distributional distances when outliers are present. An example is shown in Fig. 5.1, where the distances between distributions shown in red and blue are computed. In the absence of outliers (Fig. 5.1(a)), proper couplings (shown in green) are obtained. However, even in the presence of a very small fraction of outliers (as small as 5%), poor couplings arise leading to a large change in the distance estimate (Fig. 5.1(b)).

The sensitivity of optimal transport to outliers is undesirable, especially when

we deal with large-scale datasets where the noise is inevitable. For instance, the existence of noisy samples in source and target feature distributions can deteriorate the performance of domain adaptation systems. This sensitivity is a consequence of exactly satisfying the marginal constraints in OT's objective. Hence, to boost OT's robustness against outliers, we propose to utilize recent formulations of unbalanced optimal transport [99, 100] which relax OT's marginal constraints. The authors of [99, 100] provide an exact dual form for the unbalanced OT problem. However, we found that using this dual optimization in large-scale deep learning applications such as GANs results in poor convergence and an unstable behaviour (see [101] for details).

To remedy this issue, in this chapter, we derive a computationally efficient dual form for the unbalanced OT optimization that is suited for practical deep learning applications. Our dual simplifies to a weighted OT objective, with low weights assigned to outlier samples. These instance weights can also be useful in interpreting the difficulty of input samples for learning a given task. We develop two solvers for this dual problem based on either a discrete formulation or a continuous stochastic relaxation. These solvers demonstrate high stability in large-scale deep learning applications.

We show that, under mild assumptions, our robust OT measure (which is similar in form to the unbalanced OT) is upper bounded by a constant factor of the true OT distance (OT ignoring outliers) for any outlier distribution. Hence, our robust OT can be used for effectively handling outliers. This is visualized in Figure 5.1(c), where couplings obtained by robust OT effectively ignores outlier samples, yielding a

good estimate of the true OT distance. We demonstrate the effectiveness of the proposed robust OT formulation in two large-scale deep learning applications of domain adaptation and generative modeling. In domain adaptation, we utilize the robust OT framework for the challenging task of synthetic to real adaptation, where our approach improves adversarial adaptation techniques by $\sim 5\%$. In generative modeling, we show how robust Wasserstein GANs can be trained using state-of-the-art GAN architectures to effectively ignore outliers in the generative distrubution.

5.2 Related Work

The use of optimal transport has gained popularity in machine learning [36, 98, 102], computer vision [103, 104] and many other disciplines. Several relaxations of the OT problem have been proposed in the literature. Two popular ones include entropy regularization [51, 105] and marginal relaxation [99, 100, 106, 107, 108]. In this work, we utilize the marginal relaxations of [99, 100] for handling outlier noise in machine learning applications involving OT. To the best of our knowledge, ours is the first work to demonstrate the utility of unbalanced OT in large-scale deep learning applications. Only other work that is similar in spirit to ours is [109]. However, [109] provides a relaxation for the Monge unbalanced OT, which is different from the unbalanced Kantrovich problem we consider in this chapter.

5.3 Robust Optimal Transport

Our objective is to handle outliers in deep learning applications involving OT. For this, we use relaxed OT formulations. In this section, we first formally define the outlier model we use. Then, we discuss the existing marginal relaxation formulations in OT and the issues that arise in deep learning when using these formulations. We then propose a reformulation of the dual that is suited for deep learning.

Outlier Model: We consider outliers as samples with large noise. More specifically, let p_X and p_Y be two distributions whose Wasserstein distance we desire to compute. Let $p_X = \alpha p_X^c + (1-\alpha)p_X^a$; i.e., the clean distribution p_X^c is corrupted with $(1-\alpha)$ fraction of noise p_X^a. Then, p_X^a is considered an oulier distribution if $\mathcal{W}(p_X^c, p_Y) \ll \mathcal{W}(p_X^a, p_Y)$. For an example, refer to Fig. 5.1(b).

5.3.1 Unbalanced Optimal Transport

As seen in Fig. 5.1, sensitivity to outliers arises due to the marginal constraints in OT. If the marginal constraints are relaxed in a way that the transportation plan does not assign large weights to outliers, they can effectively be ignored. [99, 100] have proposed one such relaxation using f-divergence on marginal distributions. This formulation, called *Unbalanced Optimal Transport*, can be written as

$$\mathcal{W}^{ub}(p_X, p_Y) = \min_{\pi \in \Pi(p_{\tilde{X}}, p_{\tilde{Y}})} \int c(x,y)\pi(x,y)dxdy + \mathcal{D}_f(p_{\tilde{X}}||p_X) + \mathcal{D}_f(p_{\tilde{Y}}||p_Y) \quad (5.1)$$

where $\mathcal{D}_f$ is the f-divergence between distributions, defined as $\mathcal{D}_f(P||Q) = \int f(\frac{dP}{dQ})dQ$. Furthermore, [100] derived a dual form for the problem. Let f be a convex lower semi-continuous function. Define $r^*(x) := \sup_{s>0} \frac{x-f(s)}{s}$ where $f'_\infty := \lim_{s\to\infty} \frac{f(s)}{s}$. Then,

$$\mathcal{W}^{ub}(p_X, p_Y) = \max_{\phi,\psi} \int \phi(x)dp_X + \int \psi(y)dp_Y \tag{5.2}$$
$$\text{s.t. } r^*(\phi(x)) + r^*(\psi(y)) \leq c(x, y)$$

Computational issues using this dual form in deep learning: Training neural networks using this dual form is challenging as it involves maximizing over *two* discriminator functions (ϕ and ψ), with constraints connecting these functions. For χ^2 divergence, we derived the GAN objective using this dual and trained a model. However, we were unsuccessful in making the model converge using standard SGD as it showed severe instability. Please refer to Balaji et al. [101] for more details. This limits the utility of this formulation in deep learning applications. In what follows, we present a reformulation of the dual that is scalable and suited for deep learning applications.

5.3.2 Our Duality

We start with a slightly different form than (5.1) where we keep the f-divergence relaxations of marginal distributions as constraints:

$$\mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) := \min_{p_{\tilde{X}}, p_{\tilde{Y}} \in Prob(\mathcal{X})} \min_{\pi \in \Pi(p_{\tilde{X}}, p_{\tilde{Y}})} \int\int c(x,y)\pi(x,y)dxdy \tag{5.3}$$
$$\text{s.t. } \mathcal{D}_f(p_{\tilde{X}}||p_X) \leq \rho_1,\ \mathcal{D}_f(p_{\tilde{Y}}||p_Y) \leq \rho_2.$$

In this formulation, we optimize over the couplings whose marginal constraints are the relaxed distributions $p_{\tilde{X}}$ and $p_{\tilde{Y}}$. To prevent over-relaxation of the marginals, we impose a constraint that the f-divergence between the relaxed and the true marginals are bounded by constants ρ_1 and ρ_2 for distributions $p_{\tilde{X}}$ and $p_{\tilde{Y}}$, respectively. As seen in Fig. 5.1(c), this relaxation effectively ignores the outlier distributions when (ρ_1, ρ_2) are chosen appropriately.

Note that the Lagrangian relaxation of optimization (5.3) takes a similar form to that of the unbalanced OT objective (5.1). Having a hard constraint on f-divergence gives us an explicit control over the extent of the marginal relaxation which is suited for handling outliers. This subtle difference in how the constraints are imposed leads to a dual form of our robust OT that can be computed efficiently for deep learning applications compared to that of the unbalanced OT dual.

We consider the ℓ_2 distance as our choice of cost function in the OT formulation. In this case, the OT distance is also called the *Wasserstein* distance. In that case, we have the following result:

Theorem 2. *Let p_X and p_Y be two distributions defined on a metric space. The robust Wasserstein measure admits the following dual form*

$$\mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) = \min_{p_{\tilde{X}}, p_{\tilde{Y}}} \max_{D(.)\in Lip-1} \int D(x)dp_{\tilde{X}} - \int D(x)dp_{\tilde{Y}} \tag{5.4}$$
$$s.t\ \mathcal{D}_f(p_{\tilde{X}}||p_X) \leq \rho_1,\ \mathcal{D}_f(p_{\tilde{Y}}||p_Y) \leq \rho_2.$$

Proof. We begin with the primal form of the robust optimal transport defined as

$$\mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) = \min_{p_{\tilde{X}}, p_{\tilde{Y}} \in Prob(\mathcal{X})} \min_{\pi} \int\int c(x,y)\pi(x,y)dxdy$$
$$\text{s.t. } \mathcal{D}_f(p_{\tilde{X}}||p_X) \leq \rho_1,\ \mathcal{D}_f(p_{\tilde{Y}}||p_Y) \leq \rho_2$$
$$\int \pi(x,y)dy = p_{\tilde{X}}, \int \pi(x,y)dx = p_{\tilde{Y}}$$

The constraint $p_{\tilde{X}}, p_{\tilde{Y}} \in Prob(\mathcal{X})$ states that $p_{\tilde{X}}$ and $p_{\tilde{Y}}$ are valid probability distributions. For brevity, we shall ignore explicitly stating it in the rest of the proof.

Now, we write the Lagrangian function with respect to marginal constraints.

$$
\begin{aligned}
L = \min_{p_{\tilde{X}}, p_{\tilde{Y}}} \min_{\pi>0} \max_{\phi(x),\psi(y)} & \int\int c(x,y)\pi(x,y)dxdy + \int \phi(x) \left(\int p_{\tilde{X}} - \pi(x,y)dy \right) dx \\
& + \int \psi(y) \left(\int \pi(x,y)dx - p_{\tilde{Y}} \right) dy \\
\text{s.t. } & \mathcal{D}_f(p_{\tilde{X}}||p_X) \leq \rho_1,\ \mathcal{D}_f(p_{\tilde{Y}}||p_Y) \leq \rho_2 \\
= \min_{p_{\tilde{X}}, p_{\tilde{Y}}} \min_{\pi>0} \max_{\phi(x),\psi(y)} & \int\int \left[c(x,y) - \phi(x) + \psi(y) \right] \pi(x,y)dxdy \\
& + \int \phi(x)dp_{\tilde{X}} - \int \psi(y)dp_{\tilde{Y}} \\
\text{s.t. } & \mathcal{D}_f(p_{\tilde{X}}||p_X) \leq \rho_1,\ \mathcal{D}_f(p_{\tilde{Y}}||p_Y) \leq \rho_2
\end{aligned}
$$

Since $\pi > 0$, we observe that

$$
c(x,y) - \phi(x) + \psi(y) = \begin{cases} \infty & \text{if } c(x,y) - \phi(x) + \psi(y) > 0 \\ 0 & \text{otherwise} \end{cases}
$$

Hence, the dual formulation becomes

$$
\begin{aligned}
\mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) = \min_{p_{\tilde{X}}, p_{\tilde{Y}}} \max_{\phi(x),\psi(y)} & \int \phi(x)dp_{\tilde{X}} - \int \psi(y)dp_{\tilde{Y}} \\
\text{s.t. } & \phi(x) - \psi(y) \leq c(x,y) \\
& \mathcal{D}_f(p_{\tilde{X}}||p_X) \leq \rho_1,\ \mathcal{D}_f(p_{\tilde{Y}}||p_Y) \leq \rho_2
\end{aligned} \tag{5.5}
$$

Furthermore, when the distributions lie in a metric space, we can further simplify

this duality. Define

$$k(x) := \inf_y c(x,y) + \psi(y) \tag{5.6}$$

Since the feasible set in the dual problem satisfies $\phi(x) - \psi(y) \leq c(x,y)$, $\phi(x) \leq k(x)$, and by using $y = x$ in Eq (5.6), we obtain, $k(x) \leq \psi(x)$. Hence, $\phi(x) \leq k(x) \leq \psi(x)$.

$$\begin{aligned} |k(x) - k(x')| &= |\inf_y [c(x,y) + \psi(y)] - \inf_y [c(x',y) + \psi(y)]| \\ &\leq |c(x,y) - c(x',y)| \end{aligned}$$

Hence, $k(.)$ is 1-Lipschitz. Using the above inequalities in (5.5), we obtain,

$$\begin{aligned} \mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) \ \leq \min_{p_{\tilde{X}}, p_{\tilde{Y}}} \max_{k \in Lip-1} & \int k(x) dp_{\tilde{X}} - \int k(x) dp_{\tilde{Y}} \\ & \text{s.t. } \mathcal{D}_f(p_{\tilde{X}} || p_X) \leq \rho_1, \ \mathcal{D}_f(p_{\tilde{Y}} || p_Y) \leq \rho_2 \end{aligned}$$

Also, $\phi(x) = k(x)$ and $\psi(x) = k(x)$ is a feasible solution in optimization (5.5). Since (5.5) maximizes over $\phi(.)$ and $\psi(.)$, we obtain

$$\begin{aligned} \mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) \ \geq \min_{p_{\tilde{X}}, p_{\tilde{Y}}} \max_{k \in Lip-1} & \int k(x) dp_{\tilde{X}} - \int k(x) dp_{\tilde{Y}} \\ & \text{s.t } \mathcal{D}_f(p_X, p_{\tilde{X}}) \leq \rho_1, \ \mathcal{D}_f(p_Y, p_{\tilde{Y}}) \leq \rho_2 \end{aligned}$$

Combining these two inequalities, we obtain

$$\mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) = \min_{p_{\tilde{X}}, p_{\tilde{Y}}} \max_{k \in Lip-1} \int k(x) dp_{\tilde{X}} - \int k(x) dp_{\tilde{Y}} \tag{5.7}$$
$$\text{s.t } \mathcal{D}_f(p_X, p_{\tilde{X}}) \leq \rho_1,\ \mathcal{D}_f(p_Y, p_{\tilde{Y}}) \leq \rho_2$$

The above equation is similar in spirit to the *Kantrovich-Rubinstein* duality. An important observation to note is that the above optimization only maximizes over a single discriminator function (as opposed to two functions in optimization (5.5)). Hence, it is easier to train it in large-scale deep learning problems such as GANs.

□

Thus, we can obtain a dual form for robust OT similar to the *Kantrovich-Rubinstein* duality. The key difference of this dual form compared to the unbalanced OT dual (opt. (5.2)) is that we optimize over a single dual function $D(\cdot)$ as opposed to two dual functions in (5.2). This makes our formulation suited for deep learning applications such as GANs and domain adaptation. Note that the integrals in opt. (5.4) are taken with respect to the relaxed distributions $p_{\tilde{X}}$ and $p_{\tilde{Y}}$ which is a non-trivial computation.

In particular, we present two approaches for optimizing the dual problem (5.4):

Discrete Formulation. In practice, we observe empirical distributions $\mathbb{P}_X$ and $\mathbb{P}_Y$ from the population distributions p_X and p_Y. Let $\{\mathbf{x}_i\}_{i=1}^m$, $\{\mathbf{y}_i\}_{i=1}^n$ be the samples corresponding to the empirical distribution $\mathbb{P}_X$ and $\mathbb{P}_Y$, respectively. Following [110], we use weighted empirical distribution for the perturbed distribution $\mathbb{P}_{\tilde{X}}$, i.e.,

$\mathbb{P}_X(\mathbf{x}_i) = 1/m$ and $\mathbb{P}_{\tilde{X}}(\mathbf{x}_i) = w_i^x$.

Let $\mathbf{w}_x = [w_1^x, \dots w_m^x]$. For $\mathbb{P}_{\tilde{X}}$ to be a valid pmf, $\mathbf{w}_x$ should lie in a simplex ($\mathbf{w}^x \in \Delta^m$) i.e., $w_i^x > 0$ and $\sum_i w_i^x = 1$. Then, the robust Wasserstein objective can be written as

$$\mathcal{W}^{rob}_{\rho_1,\rho_2}(\mathbb{P}_X, \mathbb{P}_Y) = \min_{\mathbf{w}_x \in \Delta^m, \mathbf{w}_y \in \Delta^n} \max_{\mathbf{D} \in Lip-1} (\mathbf{w}_x)^t \mathbf{d}_x - (\mathbf{w}_y)^t \mathbf{d}_y$$
$$\text{s.t } \frac{1}{m}\sum_i f(m w_i^x) \leq \rho_1, \quad \frac{1}{n}\sum_i f(n w_i^y) \leq \rho_2$$

where $\mathbf{d}_x = [D(\mathbf{x}_1), D(\mathbf{x}_2) \dots D(\mathbf{x}_m)]$, and $\mathbf{d}_y = [D(\mathbf{y}_1), D(\mathbf{y}_2) \dots D(\mathbf{y}_n)]$. Since $f(.)$ is a convex function, the set of constraints involving $\mathbf{w}_x$ and $\mathbf{w}_y$ are convex w.r.t weights. We use χ^2 as our choice of f-divergence for which $f(t) = (t-1)^2/2$. The optimization then becomes

$$\mathcal{W}^{rob}_{\rho_1,\rho_2}(\mathbb{P}_X, \mathbb{P}_Y) = \min_{\mathbf{w}_x \in \Delta^m, \mathbf{w}_y \in \Delta^n} \max_{\mathbf{D} \in Lip-1} (\mathbf{w}_x)^t \mathbf{d}_x - (\mathbf{w}_y)^t \mathbf{d}_y \tag{5.8}$$
$$\text{s.t } \left\| \mathbf{w}_x - \frac{1}{m} \right\|_2 \leq \sqrt{\frac{2\rho_1}{m}}, \quad \left\| \mathbf{w}_y - \frac{1}{n} \right\|_2 \leq \sqrt{\frac{2\rho_2}{n}}$$

We solve this optimization using an alternating gradient descent between $\mathbf{w}$ and D updates. The above optimization is a second-order cone program with respect to weights $\mathbf{w}$ (for a fixed D). For a fixed $\mathbf{w}$, D is optimized using stochastic gradient descent similar to [36].

Continuous Stochastic Relaxation. In (5.8), weight vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ are optimized by solving a second order cone program. Since the dimension of weight

vectors is the size of the entire dataset, solving this optimization is expensive for large datasets. Hence, we propose a continuous stochastic relaxation for (5.4). Let us assume that supports of $p_{\tilde{X}}$ and p_X match (satisfied in real spaces). We make the following reparameterization: $p_{\tilde{X}}(\mathbf{x}) = W_x(\mathbf{x})p_X(\mathbf{x})$. For $p_{\tilde{X}}$ to be a valid pdf, we require $\int W_x(\mathbf{x})dp_X = 1$, i.e., $\mathbb{E}_{\mathbf{x}\sim p_X}[W_x(\mathbf{x})] = 1$. The constraint on f-divergence becomes $\mathbb{E}_{\mathbf{x}\sim p_X}[f(W_x(\mathbf{x}))] \leq \rho_1$. Using these, the dual of robust Wasserstein measure can be written as

$$\mathcal{W}^{rob}_{\rho_1,\rho_2}(p_X, p_Y) = \min_{W_x,W_y} \max_{D\in Lip-1} \mathbb{E}_{\mathbf{x}\sim p_X}[W_x(\mathbf{x})D(\mathbf{x})] - \mathbb{E}_{\mathbf{y}\sim p_Y}[W_y(\mathbf{y})D(\mathbf{y})] \tag{5.9}$$

$$\text{s.t} \;\; \mathbb{E}_{\mathbf{x}\sim p_X}[f(W_x(\mathbf{x}))] \leq \rho_1, \;\; \mathbb{E}_{\mathbf{y}\sim p_Y}[f(W_y(\mathbf{y}))] \leq \rho_2$$

$$\mathbb{E}_{\mathbf{x}\sim p_X}[W_x(\mathbf{x})] = 1, \;\; \mathbb{E}_{\mathbf{y}\sim p_Y}[W_y(\mathbf{y})] = 1, W_x(\mathbf{x}) \geq 0, W_y(\mathbf{y}) \geq 0$$

$W_x(.)$ and $W_y(.)$ are weight functions which can be implemented using neural networks. One crucial benefit of the above formulation is that it can be easily trained using stochastic GD.

5.3.3 Can robust OT handle outliers?

Theorem 3. *Let $\mathbb{P}_X$ and $\mathbb{P}_Y$ be two empirical distributions such that $\mathbb{P}_X$ is corrupted with γ fraction of outliers i.e., $\mathbb{P}_X = (1-\gamma)\mathbb{P}^c_X + \gamma\mathbb{P}^a_X$, where $\mathbb{P}^c_X$ is the clean distribution and $\mathbb{P}^a_X$ is the outlier distribution. Let $\mathcal{W}(\mathbb{P}^a_X, \mathbb{P}^c_X) = k\mathcal{W}(\mathbb{P}^c_X, \mathbb{P}_Y)$, with*

$k \geq 1$. *Then,*

$$\mathcal{W}^{rob}_{\rho,0}(\mathbb{P}_X, \mathbb{P}_Y) \leq \max\Big(1, 1 + k\gamma - k\sqrt{2\rho\gamma(1-\gamma)}\Big)\mathcal{W}(\mathbb{P}^c_X, \mathbb{P}_Y).$$

Proof. Let $\{\mathbf{x}^a_i\}_{i=1}^{n_a}$ be the samples in the anomaly distribution $\mathbb{P}^a_X$, $\{\mathbf{x}^c_i\}_{i=1}^{n_c}$ be the samples in the clean distribution $\mathbb{P}^c_X$, and $\{\mathbf{y}_i\}_{i=1}^{m}$ be the samples in the distribution $\mathbb{P}_Y$. We also know that $\frac{n_a}{n_a+n_c} = \gamma$.

$\mathcal{W}^{rob}_{\rho,0}(\mathbb{P}_X, \mathbb{P}_Y)$ is defined as

$$\begin{aligned}\mathcal{W}^{rob}_{\rho,0}(\mathbb{P}_X, \mathbb{P}_Y) = &\min_{\mathbb{P}_{\tilde{X}} \in Prob(\mathcal{X})} \min_{\pi} \sum_i \sum_j \pi_{ij} c_{ij} \\ &\text{s.t. } \sum_j \pi_{ij} = \mathbb{P}_{\tilde{X}}, \quad \sum_i \pi_{ij} = \mathbb{P}_Y, \quad \mathcal{D}_{\chi^2}(\mathbb{P}_{\tilde{X}} || \mathbb{P}_X) \leq \rho\end{aligned}$$

Let π^{c*} and π^{a*} be the optimal transport plans for $\mathcal{W}(\mathbb{P}^c_X, \mathbb{P}_Y)$ and $\mathcal{W}(\mathbb{P}^a_X, \mathbb{P}_Y)$ respectively. We consider transport plans of the form $\beta\pi^{a*} + (1-\beta)\pi^{c*}$, for $\beta \in [0, 1]$. The marginal constraints can then be written as

$$\begin{aligned}\int \beta\pi^{a*} + (1-\beta)\pi^{c*} dx &= \beta\mathbb{P}^a_X + (1-\beta)\mathbb{P}^c_X \\ \int \beta\pi^{a*} + (1-\beta)\pi^{c*} dy &= \mathbb{P}_Y\end{aligned}$$

For this to be a feasible solution for $\mathcal{W}^{rob}_{\rho,0}(\mathbb{P}_X, \mathbb{P}_Y)$, we require

$$\mathcal{D}_{\chi^2}(\beta\mathbb{P}^a_X + (1-\beta)\mathbb{P}^c_X || \gamma\mathbb{P}^a_X + (1-\gamma)\mathbb{P}^c_X) \leq \rho$$

The distribution $\beta\mathbb{P}_X^a + (1-\beta)\mathbb{P}_X^c$ can be characterized as

$$[\underbrace{\frac{\beta}{n_a}, \dots}_{n_a \text{ terms}} , \underbrace{\frac{1-\beta}{n_c}, \dots}_{n_c \text{ terms}}].$$

Using this, the above constraint can be written as

$$(\beta - \gamma)^2 \leq 2\rho\gamma(1-\gamma) \tag{5.10}$$

Hence, all transport plans of the form $\beta\pi^{a*} + (1-\beta)\pi^{c*}$ are feasible solutions of $\mathcal{W}_{\rho,0}^{rob}(\mathbb{P}_X, \mathbb{P}_Y)$ if β satisfies $(\beta-\gamma)^2 \leq 2\rho\gamma(1-\gamma)$. Therefore, we have:

$$\begin{aligned}
\mathcal{W}_{\rho,0}^{rob}(\mathbb{P}_X, \mathbb{P}_Y) &\leq \min_{\beta} \sum_i \sum_j c_{i,j}[\beta\pi_{i,j}^{a*} + (1-\beta)\pi_{i,j}^{c*}] \\
&\leq \min_{\beta} \beta\mathcal{W}(\mathbb{P}_X^a, \mathbb{P}_Y) + (1-\beta)\mathcal{W}(\mathbb{P}_X^c, \mathbb{P}_Y) \\
&\quad \text{s.t. } (\beta-\gamma)^2 \leq 2\rho\gamma(1-\gamma)
\end{aligned}$$

By the assumption, we have:

$$\begin{aligned}
\mathcal{W}(\mathbb{P}_X^c, \mathbb{P}_X^a) &= k\mathcal{W}(\mathbb{P}_X^c, \mathbb{P}_Y) \\
\mathcal{W}(\mathbb{P}_X^a, \mathbb{P}_Y) &\leq \mathcal{W}(\mathbb{P}_X^a, \mathbb{P}_X^c) + \mathcal{W}(\mathbb{P}_X^c, \mathbb{P}_Y) \\
&\leq (k+1)\mathcal{W}(\mathbb{P}_X^c, \mathbb{P}_Y)
\end{aligned}$$

Hence,

$$\mathcal{W}^{rob}_{\rho,0}(\mathbb{P}_X, \mathbb{P}_Y) \leq \min_{\beta} (1+\beta k)\mathcal{W}(\mathbb{P}^c_X, \mathbb{P}_Y)$$
$$\text{s.t. } (\beta-\gamma)^2 \leq 2\rho\gamma(1-\gamma)$$

The smallest value β can take is $\gamma - \sqrt{2\rho\gamma(1-\gamma)}$. This gives

$$\mathcal{W}^{rob}_{\rho,0}(\mathbb{P}_X, \mathbb{P}_Y) \leq \max\left(1, 1+k\gamma - k\sqrt{2\rho\gamma(1-\gamma)}\right)\mathcal{W}(\mathbb{P}^c_X, \mathbb{P}_Y)$$

Note: The transport plan $\beta\pi^{a*} + (1-\beta)\pi^{c*}$ is not the the optimal transport plan for the robust OT optimization between corrupted distribution $\mathbb{P}_X$ and $\mathbb{P}_Y$. However, this plan is a "feasible" solution that satisfies the constraints of the robust OT. Hence, the cost obtained by this plan is an upper bound to the true robust OT cost.

□

The above theorem states that robust OT obtains a provably robust distance estimate under our outlier model. That is, the robust OT is upper bounded by a constant factor of the true Wasserstein distance. This constant depends on the hyper-parameter ρ: when ρ is appropriately chosen, robust OT measure obtains a value approximately close to the true distance. Note that we derive this result for one-sided robust OT ($\mathcal{W}^{rob}_{\rho,0}$), which is the robust OT measure when marginals are relaxed only for one of the input distributions. This is the form we use for GANs and DA experiments (Section. 5.4).

Choosing ρ and the tightness of the bound: The constant ρ in our formulation is a hyper-parameter that needs to be estimated. The value of ρ denotes the extent of marginal relaxation. In applications such as GANs or domain adaptation, performance on a validation set can be used for choosing ρ. Or when the outlier fraction γ is known, an appropriate choice of ρ is $\rho = \frac{\gamma}{2(1-\gamma)}$. More details and experiments on tightness of our upper bound are provided in Balaji et al. [101].

5.4 Experiments

For all our experiments, we use one-sided robust Wasserstein ($\mathcal{W}^{rob}_{\rho,0}$) where the marginals are relaxed only for one of the input distributions. Please refer to Balaji et al. [101] for more experimental details. Code for our experiments is available at `https://github.com/yogeshbalaji/robustOT`.

5.4.1 Domain adaptation

Let $\mathcal{D}_{src} = \{(\mathbf{x}^s_i, y^s_i)\}_{i=1}^{n_s}$ and $\mathcal{D}_{tgt} = \{(\mathbf{x}^t_i)\}_{i=1}^{n_t}$ denote the source and target distributions, respectively. Let F denote a feature network, and C denote a classifier. Then, the UDA optimization that minimizes the robust OT distance between source and target feature distributions can be written as

$$\min_{F,C} \frac{1}{n_s} \sum_i \mathcal{L}_{cls}(\mathbf{F}(\mathbf{x}_i), y_i) + \lambda \left[\min_{\mathbf{w} \in \Delta^{n_t}} \max_D \frac{1}{n_s} \sum_i D(F(\mathbf{x}^s_i)) - \frac{1}{n_t} \sum_j w_j D(F(\mathbf{x}^t_j)) \right] \tag{5.11}$$

$$\text{s.t } \|n_t \mathbf{w} - 1\|_2 \leq \sqrt{2\rho n_t}$$

where $\mathbf{w} = [w_1, w_2 \dots w_{n_t}]$. That is, we minimize a combination of two terms - cross-entropy loss in source domain and the robust Wasserstein distance between source and target feature embeddings. Cross entropy loss ensures that we obtain consistent predictions in the source domain, and the robust Wasserstein term minimizes the feature space drift.

Algorithm 3 Domain adaptation training algorithm

Require: n_{iter}: Number of training iterations, n_{critic}: Number of critic iterations, n_{batch}: Batch size, n_{weight}: Number of weight update iterations

1: Intialize weight bank $\mathbf{w}_b = [w_1, w_2, \dots w_{n_t}]$, where each w_i corresponds to weight of target $\mathbf{x}_i^t$
2: **for** t in $1 : n_{iter}$ **do**
3: Sample a batch of labeled source images $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^{n_{batch}} \sim \mathcal{D}_{src}$.
4: Sample a batch of unlabeled target images $\{\mathbf{x}_i^t\}_{i=1}^{n_{batch}} \sim \mathcal{D}_{tdt}$
5: Obtain the weight vectors w_i^t corresponding to the target samples $\mathbf{x}_i^t$ from the weight bank $\mathbf{w}_b$
6: Obtain discriminator loss as

$$\mathcal{L}_{disc} = \frac{1}{n_{batch}} \sum_i D(F(\mathbf{x}_i^s)) - \frac{1}{n_{batch}} \sum_i w_i^t D(F(\mathbf{x}_i^t))$$

7: Obtain source label prediction loss as

$$\mathcal{L}_{cls} = \frac{1}{n_{batch}} \sum_i \mathcal{L}_{ce}(C(F(\mathbf{x}_i^s)), y_i^s)$$

8: Update discriminator $D \leftarrow D - \eta_d \nabla_D \mathcal{L}_{disc}$
9: Update feature network and classifier $F \leftarrow F - \eta_f \nabla_F \mathcal{L}_{cls}$, $C \leftarrow C - \eta_c \nabla_C \mathcal{L}_{cls}$
10: **if** $t \,\%n_{weight} == 0$ **then**
11: Update weight bank $\mathbf{w}_b$ using Algorithm 4
12: **end if**
13: **if** $t \,\%n_{critic} == 0$ **then**
14: Update feature network as $F \leftarrow F + \eta_f \nabla_F \mathcal{L}_{disc}$
15: **end if**
16: **end for**

While we describe this formulation for the Wasserstein distance, similar ideas can be applied to other adversarial losses. For instance, by replacing the second and third terms of (5.11) with binary cross entropy loss, we obtain the non-saturating

Algorithm 4 Algorithm for updating weights

1: Form the discriminator vector $\mathbf{d} = [D(\mathbf{x}_1^t), D(\mathbf{x}_2^t), \dots D(\mathbf{x}_{N_t}^t)]$

2: Obtain $\mathbf{w}_b$ as the solution of the following second-order cone program

$$\begin{aligned} &\min_{\mathbf{w}} (\mathbf{w})^t \mathbf{d} \\ &\text{s.t } \|\mathbf{w} - 1\|_2 \leq \sqrt{2\rho_1 N_t} \\ &\quad \mathbf{w} \geq 0, \ (\mathbf{w})^t \mathbf{1} = N_t \end{aligned}$$

3: Return $\mathbf{w}_b$

Table 5.1: Cross-domain recognition accuracy on VISDA-17 dataset using Resnet-18 model averaged over 3 runs.

Method	Accuracy (in %)
Source only	44.7
Adversarial (*no ent*)	55.4
Robust adversarial (*no ent*)	62.9
Adversarial (*with ent*)	59.5
Robust adversarial (*with ent*)	**63.9**

objective. Note that we use the discrete formulation of dual objective (Section 5.3.2) instead of the continuous one (Section 5.3.2). This is because in our experiments, small batch sizes (~ 28) were used due to GPU limitations. With small batch sizes, continuous relaxation gives sub-optimal performance. The training algorithm we use is provided in Alg. 3

For experiments, we use VISDA-17 dataset [114], which is a large scale benchmark dataset for UDA. The task is to perform 12- class classification by adapting models from synthetic to real dataset. In our experiments, we use non-saturating loss instead of Wasserstein to enable fair comparison with other adversarial approaches such as DANN. In addition to the adversarial alignment, we use an entropy regularizer on target logits, which is a standard technique used in UDA [115]. The

Table 5.2: Adaptation accuracy on VISDA-17 using Resnet-50 model averaged over 3 runs.

	Method	Accuracy (in %)
	Source Only	50.7
	DAN [79]	53.0
	RTN [80]	53.6
	DANN [71]	55.0
	JAN-A [81]	61.6
	GTA [87]	69.5
	SimNet [111]	69.6
	CDAN-E [112]	70.0
Ours	Adversarial (*no ent*)	62.9
Ours	Robust adversarial (*no ent*)	68.6
Ours	Adversarial (*with ent*)	65.5
Ours	Robust adversarial (*with ent*)	**71.5**

adaptation results using Resnet-18, Resnet-50 and Resnet-101 models are shown in Tables 5.1, 5.2 and 5.3, respectively. Our robust adversarial objective gives consistent performance improvement of $\sim \mathbf{5}\%$ over the standard adversarial objective in all experiments. By using a weighted adversarial loss, our approach assigns low weights to samples that are hard to adapt and high weights to target samples that look more similar to source, thereby promoting improved adaptation. Also, with the use of entropy regularization, our generic robust adversarial objective reaches performance on par with other competing approaches that are tuned specifically for the UDA problem. This demonstrates the effectiveness of our approach.

Ablation: Sensitivity of ρ In Table. 5.4, we report the sensitivity of ρ for both GANs and domain adaptation experiments. In the case of GANs, performance is relatively low only for very low values of ρ and stable for higher values. For DA, sensititivity is low in general. For all DA experiments, we used $\rho = 0.2$ without

Table 5.3: Adaptation accuracy on VISDA-17 using Resnet-101 model averaged over 3 runs.

	Method	Accuracy (in %)
	Source only	55.3
	DAN [79]	61.1
	DANN [71]	57.4
	MCD [113]	71.9
Ours	Adversarial (*no ent*)	65.5
Ours	Robust adversarial (*no ent*)	69.3
Ours	Adversarial (*with ent*)	69.3
Ours	Robust adversarial (*with ent*)	**72.7**

Table 5.4: Sensitivity Analysis of ρ

GAN exp	ρ	0	0.01	0.05	0.1	0.15
CIFAR + MNIST	FID	37.5	34.7	31.9	**29.9**	30.2
DA exp	ρ	0.0	0.05	0.1	0.2	0.4
Resnet-18	Acc	59.5	62.8	63.1	**63.9**	63.6

tuning it individually for each setting.

5.4.2 Generative modeling

In this section, we show how our robust Wasserstein formulation can be used to train GANs that are insensitive to outliers. The core idea is to train a GAN by minimizing the robust Wasserstein measure (in dual form) between real and generative data distributions. Let $\mathbf{G}$ denote a generative model which maps samples from random noise vectors to real data distribution. Using the one-directional version of the dual form of robust Wasserstein measure (5.9), we obtain the following

optimization problem

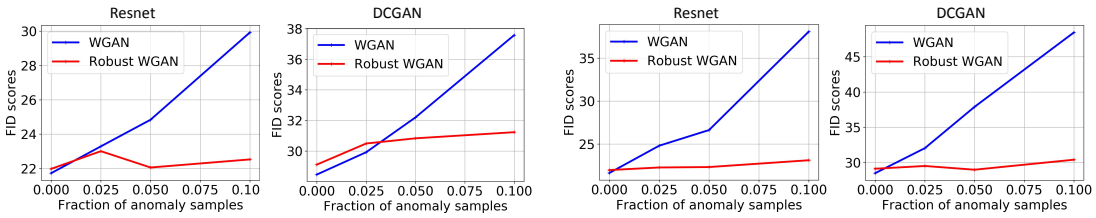
$$\min_{W,G} \max_{D \in Lip-1} \mathbb{E}_{\mathbf{x} \sim p_{data}}[W(\mathbf{x})D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}}[D(G(\mathbf{z}))]$$
$$\text{s.t} \quad \mathbb{E}_{\mathbf{x} \sim p_{data}}[(W(\mathbf{x}) - 1)^2] \leq 2\rho, \quad \mathbb{E}_{\mathbf{x} \sim p_{data}}[W(\mathbf{x})] = 1, \quad W(\mathbf{x}) \geq 0$$

The first constraint is imposed using a Lagrangian term in the objective function. To impose the second constraint, we use ReLU as the final layer of $W(\cdot)$ network and normalize the weights by the sum of weight vectors in a batch. This leads to the following optimization

$$\min_{W,G} \max_{D \in Lip-1} \mathbb{E}_{\mathbf{x}}[W(\mathbf{x})D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}}[D(G(\mathbf{z}))] + \lambda \max\left(\mathbb{E}_{\mathbf{x}}[(W(\mathbf{x}) - 1)^2] - 2\rho, 0\right) \tag{5.12}$$

We set λ to a large value (typically $\lambda = 1000$) to enforce the constraint on χ^2-divergence. A detailed algorithm can be found in Balaji et al. [101]. Our robust Wasserstein formulation can easily be extended to other GAN objective functions such as non-saturating loss and hinge loss, as discussed in Balaji et al. [101].

Datasets with outliers: First, we train the robust Wasserstein GAN on datasets corrputed with outlier samples. For the ease of quantitative evaluation, the outlier corrupted dataset is constructed as follows: We artificially add outlier samples to the CIFAR-10 dataset such they occupy γ fraction of the samples. MNIST and uniform noise are used as two choices of outlier distributions. Samples generated by Wasserstein GAN and robust Wasserstein GAN on this dataset are shown in



(a) Clean dataset (b) Dataset with outliers

Figure 5.2: FID scores of GAN models trained on CIFAR-10 corrupted with outlier noise. In (a), samples from MNIST dataset are used as the outliers, while in (b), uniform noise is used. FID scores of WGAN increase with the increase in outlier fraction, while robust WGAN maintains FID scores.

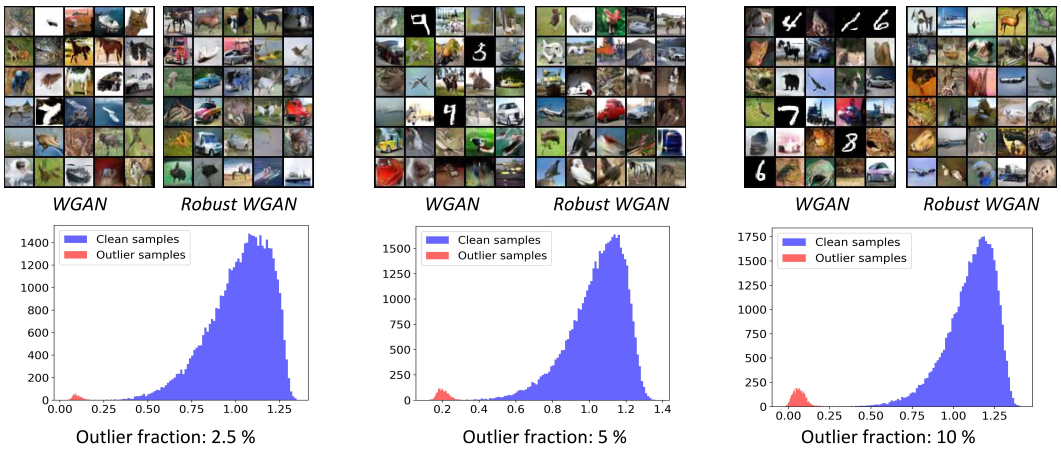


Figure 5.3: Visualizing samples and weight histograms. In the top panel, we show samples generated by WGAN and robust WGAN trained on the CIFAR-10 dataset corrupted with MNIST samples as outliers. WGAN fits both CIFAR and MNIST samples, while the robust WGAN ignores the outliers. In the bottom panel, we visualize the weights (output of the $W(.)$ function) for in-distribution and outlier samples. The outlier samples are assinged low weights while in-distribution samples get large weights.

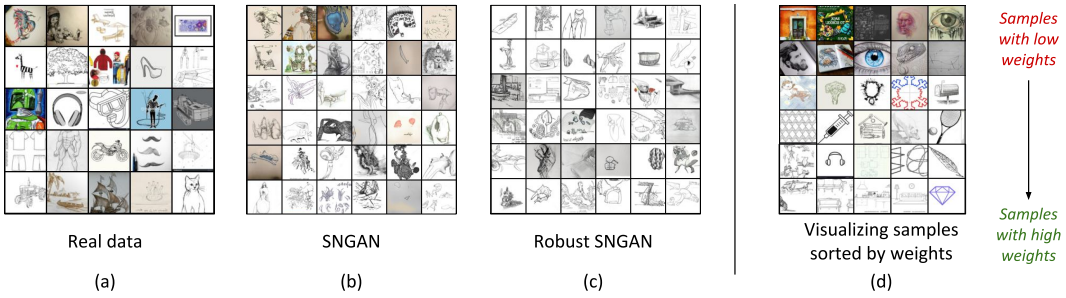


Figure 5.4: Visualizing samples generated on Domainnet sketch dataset. In panels (a), (b) and (c), we show the real data, samples generated by SNGAN and robust SNGAN, respectively. Robust SNGAN only generates images of sketches ignoring outliers. In panel (d), we visualize real samples sorted by weights. Low weights are assigned to outliers, while sketch images get large weights.

Fig. 5.3. While Wasserstein GAN fits outliers in addition to the CIFAR samples, the robust Wasserstein GAN effectively ignores outliers and generates samples only from the CIFAR-10 dataset.

For a quantitative evaluation, we report the FID scores of the generated samples with respect to the clean CIFAR-10 distribution (Figure 5.2). Since Wasserstein GAN generates outlier samples in addition to the CIFAR-10 samples, the FID scores get worse as the outlier fraction increases. Robust Wasserstein GAN, on the other hand, obtains good FID even for large fraction of outliers. This trend is consistent for both outlier distributions MNIST and uniform noise.

Next, we train our robust GAN model on a dataset where outliers are naturally present. We use *Sketch* domain of DomainNet dataset [116] for this purpose. As shown in Figure 5.4(a), the dataset contains many outlier samples (non-sketch images). Samples generated by spectral normalization GAN and robust spectral

Table 5.5: Quantitative evaluation of robust WGAN on clean datasets. In each cell, the top row corresponds to the Inception score and the bottom row corresponds to the FID score.

Dataset	**Arch**	WGAN	RWGAN	
			$\rho = 0$	$\rho = 0.3$
CIFAR-10	DCGAN	6.86 28.46	6.84 29.11	6.91 29.45
CIFAR-10	Resnet	7.49 21.73	7.35 21.98	7.36 21.57
CIFAR-100	Resnet	9.01 15.60	8.79 15.61	8.93 15.32

normalization GAN (both using Resnet) are shown in Figure 5.4(b, c). We observe that the SNGAN model generates some non-sketch images in addition to sketch images. Robust SNGAN, on the other hand, ignores outliers and only generates samples that look like sketches.

Clean datasets: In the previous section, we demonstrated how robust Wasserstein GAN effectively ignores outliers in the data distributions. A natural question that may arise is what would happen if one uses the robust WGAN on a clean dataset (dataset without outliers). To understand this, we train the robust Wasserstein GAN on CIFAR-10 and CIFAR-100 datasets. The Inception and FID scores of generated samples are reported in Table. 5.5. We observe no drop in FID scores, which suggest that no modes are dropped in the generated distribution.

Usefulness of sample weights: In the optimization of the robust GAN, each sample is assigned a weight indicating the difficulty of that sample to be generated by the model. In this section, we visualize the weights learnt by our robust GAN. In

Figure 5.3, we plot the histogram of weights assigned to in-distribution and outlier samples for robust WGAN trained on CIFAR-10 dataset corrupted with MNIST outliers. Outliers are assigned smaller weights compared to the in-distribution samples, and there is a clear separation between their corresponding histograms. For the GAN model trained on the Sketch dataset, we show a visualization of randomly chosen input samples sorted by their assigned weights in Figure 5.4(d). We observe that non-sketch images are assigned low weights while the true sketch images obtain larger weights. Hence, the weights learnt by our robust GAN can be a useful indicator for assessing how difficult it is to generate a given sample.

5.5 Conclusion

In this chapter, we proposed the robust optimal transport formulation which is insensitive to outliers (samples with large noise) in the data. The applications of previous formulations of robust OT are limited in practical deep learning problems such as GANs and domain adaptation due to the instability of their optimization solvers. We derive a computationally efficient dual form of the robust OT objective that is suited for deep learning applications. We demonstrate the effectiveness of the proposed method in two applications of domain adaptation and GANs, where our approach is shown to effectively handle outliers and achieve good performance improvements.

Chapter 6: Normalized Wasserstein for Mixture Distributions

6.1 Introduction

Quantifying distances between probability distributions is a fundamental problem in machine learning and statistics with applications in domain adaptation and generative modeling. Popular probability distance measures include *optimal transport* measures such as the Wasserstein distance [49] and *divergence* measures such as the Kullback-Leibler (KL) divergence [47]. Classical distance measures, however, can lead to some issues for mixture distributions. A *mixture distribution* is the probability distribution of a random variable X where $X = X_i$ with probability π_i for $1 \leq i \leq k$. k is the number of mixture components and $\pi = [\pi_1, ..., \pi_k]^T$ is the vector of *mixture (or mode) proportions*. The probability distribution of each X_i is referred to as a *mixture component* (or, a mode). Mixture distributions arise naturally in different applications where the data contains two or more sub-populations. For example, image datasets with different labels can be viewed as a mixture (or, multi-modal) distribution where samples with the same label characterize a specific mixture component.

If two mixture distributions have exactly same mixture components (i.e. same X_i's) with different mixture proportions (i.e. different π's), classical distance mea-

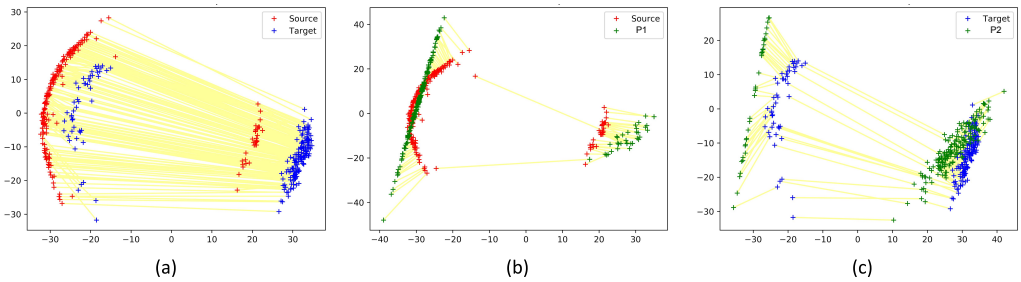


Figure 6.1: An illustration of the effectiveness of the proposed Normalized Wasserstein measure in domain adaptation. The source domain (shown in red) and the target domain (shown in blue) have two modes with different mode proportions. (a) The couplings computed by estimating Wasserstein distance between source and target distributions (shown in yellow lines) match several samples from incorrect and distant mode components. (b,c) Our proposed normalized Wasserstein measure (6.3) constructs intermediate mixture distributions $\mathbb{P}_1$ and $\mathbb{P}_2$ (shown in green) with similar mixture components to source and target distributions, respectively, but with optimized mixture proportions. This significantly reduces the number of couplings between samples from incorrect modes and leads to 42% decrease in target loss in domain adaptation compared to the baseline.

sures between the two will be large. This can lead to undesired results in several distance-based machine learning methods. To illustrate this issue, consider the *Wasserstein distance* between two distributions p_X and p_Y, defined as

$$\mathcal{W}(p_X, p_Y) := \min_{p_{X,Y}} \ \mathbb{E}\left[\|X - Y\|\right], \tag{6.1}$$

$$\text{marginal}_X(p_{X,Y}) = p_X, \ \text{marginal}_Y(p_{X,Y}) = p_Y$$

where $p_{X,Y}$ is the joint distribution (or coupling) whose marginal distributions are equal to p_X and p_Y. When no confusion arises and to simplify notation, in some equations, we use $\mathcal{W}(X, Y)$ notation instead of $\mathcal{W}(p_X, p_Y)$.

The Wasserstein distance optimization is over all joint distributions (couplings) $p_{X,Y}$ whose marginal distributions *match* exactly with input distributions p_X and p_Y. This requirement can cause issues when p_X and p_Y are mixture distributions with different mixture proportions. In this case, due to the marginal constraints, samples belonging to very different mixture components will have to be coupled together in $p_{X,Y}$ (e.g. Figure 6.1(a)). Thus, using this distance measure can then lead to undesirable outcomes in problems such as domain adaptation. This motivates the need for developing a new distance measure to take into account mode imbalances in mixture distributions.

In this chapter, we propose a new distance measure that resolves the issue of imbalanced mixture proportions for multi-modal distributions. Our developments focus on a class of optimal transport measures, namely the Wasserstein distance Eq (6.1). However, our ideas can be extended naturally to other distance measures (eg. adversarial distances [71]) as well.

Let $\mathbf{G}$ be an array of generator functions with k components defined as $\mathbf{G} := [\mathbf{G}_1, ..., \mathbf{G}_k]$. Let $p_{\mathbf{G},\pi}$ be a mixture probability distribution for a random variable X where $X = \mathbf{G}_i(Z)$ with probability π_i for $1 \leq i \leq k$. Throughout the chapter, we assume that Z has a normal distribution.

By relaxing the marginal constraints of the classical Wasserstein distance (6.1), we introduce the *Normalized Wasserstein measure* (*NW measure*) as follows:

$$\mathcal{W}_N(p_X, p_Y) := \min_{\mathbf{G},\pi^{(1)},\pi^{(2)}} \mathcal{W}(p_X, p_{\mathbf{G},\pi^{(1)}}) + \mathcal{W}(p_Y, p_{\mathbf{G},\pi^{(2)}}).$$

There are two key ideas in this definition that help resolve mode imbalance issues for mixture distributions. First, instead of directly measuring the Wasserstein distance between p_X and p_Y, we construct two intermediate (and potentially mixture) distributions, namely $p_{\mathbf{G},\pi^{(1)}}$ and $p_{\mathbf{G},\pi^{(2)}}$. These two distributions have the same mixture components (i.e. same $\mathbf{G}$) but can have different mixture proportions (i.e. $\pi^{(1)}$ and $\pi^{(2)}$ can be different). Second, mixture proportions, $\pi^{(1)}$ and $\pi^{(2)}$, are considered as optimization variables. This effectively *normalizes* mixture proportions before Wasserstein distance computations. See an example in Figure 6.1 (b, c) for a visualization of $p_{\mathbf{G},\pi^{(1)}}$ and $p_{\mathbf{G},\pi^{(2)}}$, and the re-normalization step.

In this chapter, we show the effectiveness of the proposed Normalized Wasserstein measure in domain adaptation and generative modeling. In each case, the performance of our proposed method significantly improves against baselines when input datasets are mixture distributions with imbalanced mixture proportions. Below, we briefly highlight these results:

Domain Adaptation: In Section 6.4, we formulate the problem of domain adaptation as minimizing the normalized Wasserstein measure between source and target feature distributions. On classification tasks with imbalanced datasets, our method significantly outperforms baselines (e.g. $\sim 20\%$ gain in synthetic to real adaptation on VISDA-3 dataset).

GANs: In Section 6.5, we use the normalized Wasserstein measure in GAN's formulation to train mixture models with varying mode proportions. We show that such a generative model can help capture rare modes, decrease the complexity of the generator, and re-normalize an imbalanced dataset.

6.2 Normalized Wasserstein Measure

In this section, we introduce the *normalized Wasserstein* measure and discuss its properties. Recall that $\mathbf{G}$ is an array of generator functions defined as $\mathbf{G} := [\mathbf{G}_1, ..., \mathbf{G}_k]$ where $\mathbf{G}_i : \mathbb{R}^r \to \mathbb{R}^d$. Let $\mathcal{G}$ be the set of all possible $\mathbf{G}$ function arrays. Let π be a discrete probability mass function with k elements, i.e. $\pi = [\pi_1, \pi_2, \cdots, \pi_k]$ where $\pi_i \geq 0$ and $\sum_i \pi_i = 1$. Let Π be the set of all possible π's.

Let $p_{\mathbf{G},\pi}$ be a mixture distribution, i.e. it is the probability distribution of a random variable X such that $X = \mathbf{G}_i(Z)$ with probability π_i for $1 \leq i \leq k$. We assume that Z has a normal density, i.e. $Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We refer to $\mathbf{G}$ and π as mixture components and proportions, respectively. The set of all such mixture distributions is defined as:

$$\mathcal{P}_{\mathbf{G},k} := \{p_{\mathbf{G},\pi} : \mathbf{G} \in \mathcal{G}, \pi \in \Pi\} \tag{6.2}$$

where k is the number of mixture components. Given two distributions p_X and p_Y belonging to the family of mixture distributions $\mathcal{P}_{\mathbf{G},k}$, we are interested in defining a distance measure agnostic to differences in mode proportions, but sensitive to shifts in mode components, i.e., the distance function should have high values only when mode components of p_X and p_Y differ. If p_X and p_Y have the same mode components but differ only in mode proportions, the distance should be low.

The main idea is to introduce mixture proportions as optimization variables in the Wasserstein distance formulation (6.1). This leads to the following distance

measure which we refer to as the *Normalized Wasserstein measure* (NW measure), $\mathcal{W}_N(p_X, p_Y)$, defined as:

$$\begin{aligned} \min_{\mathbf{G},\pi^{(1)},\pi^{(2)}} \quad & \mathcal{W}(p_X, p_{\mathbf{G},\pi^{(1)}}) + \mathcal{W}(p_Y, p_{\mathbf{G},\pi^{(2)}}) \\ & \sum_{j=1}^{k} \pi_j^{(i)} = 1 \qquad i = 1, 2, \\ & \pi_j^{(i)} \geq 0 \qquad\quad 1 \leq j \leq k, \quad i = 1, 2. \end{aligned} \tag{6.3}$$

Since the normalized Wasserstein's optimization (6.3) includes mixture proportions $\pi^{(1)}$ and $\pi^{(2)}$ as optimization variables, if two mixture distributions have similar mixture components with different mixture proportions (i.e. $p_X = p_{\mathbf{G},\pi^{(1)}}$ and $p_Y = p_{\mathbf{G},\pi^{(2)}}$), although the Wasserstein distance between the two can be large, the introduced normalized Wasserstein measure between the two will be zero. Note that W_N is defined with respect to a set of generator functions $\mathbf{G} = [\mathbf{G}_1, ..., \mathbf{G}_k]$. However, to simplify the notation, we make this dependency implicit. We would like to point our that our proposed NW measure is a *semi-distance* measure (and not a distance) since it does not satisfy all properties of a distance measure. Please refer to Balaji et al. [117] for more details.

To compute the NW measure, we use an alternating gradient descent approach similar to the dual computation of the Wasserstein distance [36]. Moreover, we impose the π constraints using a soft-max function. Please refer to Balaji et al. [117] for more details.

To illustrate how NW measure is agnostic to mode imbalances between dis-

tributions , consider an unsupervised *domain adaptation* problem with MNIST-2 (i.e. a dataset with two classes: digits 1 and 2 from MNIST) as the source dataset, and noisy MNIST-2 (i.e. a noisy version of it) as the target dataset (details of this example is presented in Section 6.4.2). The source dataset has $4/5^{th}$ fraction as digit 1 and $1/5^{th}$ as digits 2, while the target dataset has $1/5^{th}$ as noisy digit 1 and $4/5^{th}$ as noisy digits 2. The couplings produced by estimating the Wasserstein distance between the two distributions is shown in yellow lines in Figure 6.1-a. We observe that there are many couplings between samples from incorrect mixture components. The normalized Wasserstein measure, on the other hand, constructs intermediate mode-normalized distributions $\mathbb{P}_1$ and $\mathbb{P}_2$, which get coupled to the correct modes of source and target distributions, respectively (see panels (b) and (c) in Figure 6.1)).

6.3 Theoretical Results

For NW measure to work effectively, the number of modes k in NW formulation (Eq. (6.3)) must be chosen appropriately. For instance, given two mixture distributions with k components each, Normalized Wasserstein measure with $2k$ modes would always give 0 value. In this section, we provide some theoretical conditions under which the number of modes can be estimated accurately. We begin by making the following assumptions for two mixture distributions p_X and p_Y whose NW distance we wish to compute. We use X and p_X interchangeably for brevity.

- (A1) If mode i in distribution X and mode j in distribution Y belong to the same mixture component, then their Wasserstein distance is $\leq \epsilon$ i.e., if X_i and

Y_j correspond to the same component, $\mathcal{W}(p_{X_i}, p_{Y_j}) < \epsilon$.

- (A2) The minimum Wasserstein distance between any two modes of one mixture distribution is at least δ i.e., $\mathcal{W}(p_{X_i}, p_{X_j}) > \delta$ and $\mathcal{W}(p_{Y_i}, p_{Y_j}) > \delta \; \forall i \neq j$. Also, non-overlapping modes between X and Y are separated by δ i.e., for non-overlapping modes X_i and Y_j, $\mathcal{W}(p_{X_i}, p_{Y_j}) > \delta$. This ensures that modes are well-separated.

- (A3) We assume that each mode X_i and Y_i have density at least η i.e., $p_{X_i} \geq \eta \quad \forall i$, $p_{Y_i} \geq \eta \quad \forall i$. This ensures that every mode proportion is at least η.

- (A4) Each generator $\mathbf{G}_i$ is powerful enough to capture exactly one mode of distribution p_X or p_Y.

Let $\mathcal{W}_N(p_X, p_Y; k)$ denote the normalized wasserstein measure computed using k intermediate components between distribution p_X and p_Y. For the rest of this section, let us denote $\mathcal{W}_N(p_X, p_Y; k)$ by $\mathcal{W}_N(k)$ for brevity.

Lemma 2. *$\mathcal{W}_N(k)$ is a monotonically decreasing function with respect to k.*

Proof. This is because in $\mathcal{W}_N(k+1)$, we add one additional mode compared to $\mathcal{W}_N(k)$. If we have the mixture weights $\pi^{(1)}$, $\pi^{(2)}$ for this new mode to be 0, and have the same assignments as $\mathcal{W}_N(k)$ for the rest of the modes of $\mathcal{W}_N(k+1)$, we will obtain $\mathcal{W}_N(k+1) = \mathcal{W}_N(k)$. Since computing $\mathcal{W}_N(k)$ contains a minimization over mode assignments, the $\mathcal{W}_N(k+1) \leq \mathcal{W}_N(k) \; \forall k$. Hence, it is monotonically decreasing. □

Lemma 3. *Let $k^* = n_1 + n_2 - r$. Then, $\mathcal{W}_N(k^*) \leq \epsilon$.*

Proof. This is because at $k = k^*$, we can make the following mode assignments.

- Assign $n_1 + n_2 - r$ modes of the generator distribution to each of $n_1 + n_2 - r$ non-overlapping modes in $\mathbb{P}_X$ and $\mathbb{P}_Y$ with the same mixture.
- Assign the remaining r modes of the generator distribution to the overlapping modes of either p_X or p_Y. WLOG, let us assume we assign them to r overlapping modes of p_X.
- Choose $\pi^{(1)}$ to be same as π for p_X, and assign 0 to non-overlapping components of p_Y.
- Choose $\pi^{(2)}$ to be same as π for p_Y, with 0 to non-overlapping components of p_X.

Let us denote $nov(X)$ to be non-overlapping modes of X, $ov(X)$ to be overlapping modes of X, $nov(Y)$ to be non-overlapping modes of Y, and $ov(Y)$ to be overlapping modes of Y. Then, under the mode assignments given above, $\mathcal{W}_N(k^*)$ can be evaluated as,

$$\begin{aligned}
\mathcal{W}_N(p_X, p_Y) :&= \min_{\mathbf{G},\pi^{(1)},\pi^{(2)}} \mathcal{W}(p_X, p_{\mathbf{G},\pi^{(1)}}) + \mathcal{W}(p_Y, p_{\mathbf{G},\pi^{(2)}}). \\
&= \sum_{i \in nov(X)} \pi_i^X \mathcal{W}(p_{X_i}, p_{X_i}) + \sum_{i \in ov(X)} \pi_i^X \mathcal{W}(p_{X_i}, p_{X_i}) + \\
&\sum_{i \in nov(Y)} \pi_i^Y \mathcal{W}(p_{Y_i}, p_{Y_i}) + \sum_{i \in ov(Y)} \pi_i^Y \mathcal{W}(p_{Y_i}, p_{X_i}) \\
&= 0 + 0 + 0 + \sum_{i \in ov(Y)} \pi_i^Y \mathcal{W}(p_{Y_i}, p_{X_i}) \\
&\leq \epsilon
\end{aligned}$$

The last step follows from (A1) i.e., overlapping modes are separated by a Wasserstein distance of ϵ. $\square$

Lemma 4. $\mathcal{W}_N(k^* - 1) \geq \frac{\delta}{2}\eta$

Proof. By assumption (A2), we know that any two modes have separation of at least δ. In the distribution $p_X + p_Y$, there are $n_1 + n_2 - r$ unique cluster centers, each pair of clusters at a Wasserstein distance δ distance apart. In $\mathcal{W}_N(k^* - 1)$, the generator distribution has $n_1 + n_2 - r - 1$ modes, which is 1 less than the number of modes in $\mathbb{P}_X + \mathbb{P}_Y$. Now, let us assume that $\mathcal{W}_N(k^* - 1) < \frac{\delta}{2}\eta$. Then,

$$\mathcal{W}(p_X, p_{\mathbf{G},\pi^{(1)}}) + \mathcal{W}(p_Y, p_{\mathbf{G},\pi^{(2)}}) < \frac{\delta}{2}\eta$$

Since each mode of p_X and p_Y has density at least η (by (A3)), the above condition can be satisfied only if

$$\forall i \in [n_1], \exists j \in [k^* - 1] \text{ s.t. } \mathcal{W}(p_{X_i}, \mathbb{P}_{\mathbf{G}_j}) < \frac{\delta}{2} \tag{6.4}$$

$$\forall i \in [n_2], \exists j \in [k^* - 1] \text{ s.t. } \mathcal{W}(p_{Y_i}, \mathbb{P}_{\mathbf{G}_j}) < \frac{\delta}{2} \tag{6.5}$$

Accounting for r mode overlap between X and Y, there will be $n_1 + n_2 - r$ unique constraints in Eq. (6.4) and Eq. (6.5). Since, the generator distribution has only $k^* - 1$ modes, by Pigeonhole principle, there should be at least one pair (i, j) that is matched to the same $\mathbf{G}_j$. WLOG, let us consider both i and j to belong to $\mathbb{P}_X$,

although each can either belong to $\mathbb{P}_X$ or $\mathbb{P}_Y$. Then,

$$\mathcal{W}(p_{X_i}, \mathbf{G}_k) < \frac{\delta}{2}$$
$$\mathcal{W}(p_{X_j}, \mathbf{G}_k) < \frac{\delta}{2}$$

Then, by triangle inequality, $\mathcal{W}(p_{X_i}, p_{X_j}) < \delta$. This contradicts assumption (A2). Hence $\mathcal{W}_N(k^* - 1) \geq \frac{\delta}{2}\eta$. □

Theorem 4. *Let p_X and p_Y be two mixture distributions satisfying (A1)-(A4) with n_1 and n_2 mixture components, respectively, where r of them are overlapping. Let $k^* = n_1 + n_2 - r$. Then, k^* is smallest k for which $\mathcal{W}_N(k)$ is small ($O(\epsilon)$) and $\mathcal{W}_N(k) - \mathcal{W}_N(k-1)$ is relatively large (in the $O(\delta\eta)$)*

Proof. From Lemma 3 and Lemma 2, we know that $\mathcal{W}_N(k) \leq \epsilon \; \forall k \geq k^*$. Similarly, from Lemma 4 and Lemma 2, we $\mathcal{W}_N(k) \geq \frac{\delta}{2}\eta \; \forall k < k^*$. Hence, k^* is the smallest k for which $NW(k)$ is small ($O(\epsilon)$) and $NW(k) - NW(k-1)$ is relatively large (in the $O(\delta\eta)$). □

Note: All assumptions made are reasonable: (A1)-(A3) enforces that non-overlapping modes in mixture distributions are separated, and overlapping modes are close in Wasserstein distance. To enforce (A4), we need to prevent multi-mode generation in one mode of $\mathbf{G}$. This can be satisfied by using a regularization function. In the above theorem, k^* is the optimal k that should be used in the Normalized Wasserstein formulation. The theorem presents a way to estimate k^*. Please refer to Section 6.6 for experimental results. In many applications like domain adaption,

however, the number of components k is known beforehand, and this step can be skipped.

6.4 Normalized Wasserstein in Domain Adaptation

In this section, we demonstrate the effectiveness of the NW measure in Unsupervised Domain Adaptation (UDA) both for supervised (e.g. classification) and unsupervised (e.g. denoising) tasks. Note that the term *unsupervised* in UDA means that the label information in the target domain is unknown while *unsupervised* tasks mean that the label information in the source domain is unknown.

First, we consider domain adaptation for a classification task. Let (X^s, Y^s) be the random variable corresponding to the source data and labels, while X^t denote the random variable corresponding to the target data. Let $\mathcal{D}_{src} = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ represent the source dataset while $\mathcal{D}_{tgt}\{\mathbf{x}_i^t\}_{i=1}^{n_t}$ denote the target dataset ($\mathbf{x}_i^s, y_i^s, \mathbf{x}_i^t$ are the realizations of the random variables X^s, Y^s, X^t, respectively). In domain adaptation, we learn a feature representation to embed the source and the target samples to a common feature space where the *distance* between the source and target feature distributions is sufficiently small, while a good classifier can be computed for the source domain [71]. In this case, one solves the following optimization:

$$\min_{F} \quad \mathbb{E}_{\mathcal{D}_{src}} \left[\mathcal{L}_{cls}\left(F(X^s), Y^s\right)\right] + \lambda \text{ dist}\left(F(X^s), F(X^t)\right) \tag{6.6}$$

where λ is an adaptation parameter and $\mathcal{L}_{cls}$ is the classification loss function (e.g. the cross-entropy loss). The distance function between distributions can be adversar-

ial distances [71, 72], the Wasserstein distance [98], or MMD-based distances [79, 80].

When X^s and X^t are mixture distributions (which is often the case as each label corresponds to one mixture component) with different mixture proportions, the use of these classical distance measures can lead to the computation of inappropriate transformation and classification functions. In this case, we propose to use the NW measure as the distance function. Computing the NW measure requires training mixture components $\mathbf{G}$ and mode proportions $\pi^{(1)}, \pi^{(2)}$. To simplify the computation, we make use of the fact that labels for the source domain (i.e. Y^s) are known, thus source mixture components can be identified using these labels. Using this information, we can avoid the need for computing $\mathbf{G}$ directly and use the conditional source feature distributions as a proxy for the mixture components as follows:

$$\begin{aligned} &\mathbf{G}_i(Z) \overset{\text{dist}}{=} F(X_i^s), \\ &X_i^s = \{X^s | Y^s = i\}, \quad \forall 1 \leq i \leq k, \end{aligned} \tag{6.7}$$

where $\overset{\text{dist}}{=}$ means matching distributions. Using (6.7), the formulation for domain adaptation can be written as

$$\min_F \min_\pi \mathcal{L}_{cls}\left(X^s, Y^s\right) + \lambda \mathcal{W}\left(\sum_i \pi^{(i)} F(X_i^s), F(X^t)\right). \tag{6.8}$$

The above formulation can be seen as a version of instance weighting as source samples in X_i^s are weighted by π_i. Instance weighting mechanisms have been well

studied for domain adaptation [118, 119]. However, different from these approaches, we train the mode proportion vector π in an end-to-end fashion using neural networks and integrate the instance weighting in a Wasserstein optimization. Of more relevance to our work is the method proposed in [120], where the instance weighting is trained end-to-end in a neural network. However, in [120], instance weights are maximized with respect to the Wasserstein loss, while we show that the mixture proportions need to minimized to normalize mode mismatches. Moreover, our NW measure formulation can handle the case when mode assignments for source embeddings are unknown (as we discuss in Section 6.4.2). This case cannot be handled by the approach presented in [120].

For unsupervised tasks when mode assignments for source samples are unknown, we cannot use the simplified formulation of (6.7). In that case, we use a domain adaptation method solving the following optimization:

$$\min_{F} \quad \mathcal{L}_{unsup}\left(X_s\right) + \lambda \mathcal{W}_N\left(F(X_s), F(X_t)\right), \tag{6.9}$$

where $\mathcal{L}_{unsup}(X_s)$ is the loss corresponding to the desired *unsupervised* learning task on the source domain data.

6.4.1 Experiments: UDA for supervised tasks

6.4.1.1 MNIST → MNIST-M

In the first set of experiments[1], we consider adaptation between MNIST→ MNIST-M datasets. We consider three settings with imbalanced class proportions in source and target datasets: 3 modes, 5 modes, and 10 modes. More details can be found in Balaji et al. [117].

We use the same architecture as [71] for feature network and discriminator. We compare our method with the following approaches: (1) Source-only which is a baseline model trained only on source domain with no domain adaptation performed, (2) DANN [71], a method where adversarial distance between source and target distributions is minimized, and (3) Wasserstein [98] where Wasserstein distance between source and target distributions is minimized. Table 6.1 summarizes our results of this experiment. We observe that performing domain adaptation using adversarial distance and Wasserstein distance leads to decrease in performance compared to the baseline model. This is an outcome of not accounting for mode imbalances, thus resulting in negative transfer, i.e., samples belonging to incorrect classes are coupled and getting pushed to be close in the embedding space. Our proposed NW measure, however, accounts for mode imbalances and leads to a significant boost in performance in all three settings.

[1]Code available at `https://github.com/yogeshbalaji/Normalized-Wasserstein`

Table 6.1: Mean classification accuracies (in %) averaged over 5 runs on imbalanced MNIST→MNIST-M adaptation.

Method	3 modes	5 modes	10 modes
Source only	66.63	67.44	63.17
DANN	62.34	57.56	59.31
Wasserstein	61.75	60.56	58.22
Normalized Wasserstein	**75.06**	**76.16**	**68.57**

6.4.1.2 VISDA

In the experiment of Section 6.4.1.1 on digits dataset, models have been trained from scratch. However, a common practice used in domain adaptation is to transfer knowledge from a pretrained network (eg. models trained on ImageNet) and fine-tune on the desired task. To evaluate the performance of our approach in such settings, we consider adaptation on the VISDA dataset [114]; a recently proposed benchmark for adapting from synthetic to real images.

We consider a subset of the entire VISDA dataset containing the following three classes: *aeroplane*, *horse* and *truck*. The source domain contains $(0.55, 0.33, 0.12)$ fraction of samples per class, while that of the target domain is $(0.12, 0.33, 0.55)$. We use a Resnet-18 model pre-trained on ImageNet as our feature network. As shown in Table 6.2, our approach significantly improves the domain adaptation performance over the baseline and other compared methods.

Table 6.2: Mean classification accuracies (in %) averaged over 5 runs on synthetic to real adaptation on mode imbalanced VISDA dataset (3 classes).

Method	Accuracy (in %)
Source only	53.19
DANN	68.06
Wasserstein	64.84
Normalized Wasserstein	**73.23**

Table 6.3: Mean classification accuracies (in %) averaged over 5 runs on synthetic to real adaptation on mode balanced VISDA dataset (3 classes).

Method	Accuracy (in %)
Source only	60.22
DANN	85.24
Wasserstein	83.47
Normalized Wasserstein	84.16

6.4.1.3 Mode balanced datasets

The previous two experiments demonstrated the effectiveness of our method when datasets are imbalanced. In this section, we study the case where source and target domains have mode-balanced datasets – the standard setting considered in the most domain adaptation methods. We perform experiment on MNIST→MNIST-M adaptation using the entire dataset. Table 6.3 reports the results obtained. We observe that our approach performs on-par with the standard wasserstein distance minimization.

6.4.2 Experiments: UDA for unsupervised tasks

For unsupervised tasks on mixture datasets, we use the formulation of Eq (6.9) to perform domain adaptation. To empirically validate this formulation, we consider the image denoising problem. The source domain consists of digits $\{1, 2\}$ from MNIST dataset as shown in Fig 6.2(a). Note that the color of digit 2 is inverted. The target domain is a noisy version of the source, i.e. source images are perturbed with random $i.i.d$ Gaussian noise $\mathcal{N}(0.4, 0.7)$ to obtain target images. Our dataset contains $5,000$ samples of digit 1 and $1,000$ samples of digit 2 in the source domain, and $1,000$ samples of noisy digit 1 and $5,000$ samples of noisy digit 2 in the target. The task is to perform image denoising by dimensionaly reduction, i.e., given a target domain image, we need to reconstruct the corresponding clean image that looks like the source. We assume that no (source, target) correspondence is available in the dataset.

To perform denoising when the (source, target) correspondence is unavailable, a natural choice would be to minimize the reconstruction loss in source while minimizing the distance between source and target embedding distributions. We use the NW measure as our choice of distance measure. This results in the following optimization:

$$\min_{F,G} \quad \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{src}}\|G(F(\mathbf{x})) - \mathbf{x}\|_2^2 + \lambda\mathcal{W}_N\left(F(X^s), F(X^t)\right)$$

where $F(\cdot)$ is the encoder and $G(\cdot)$ is the decoder.

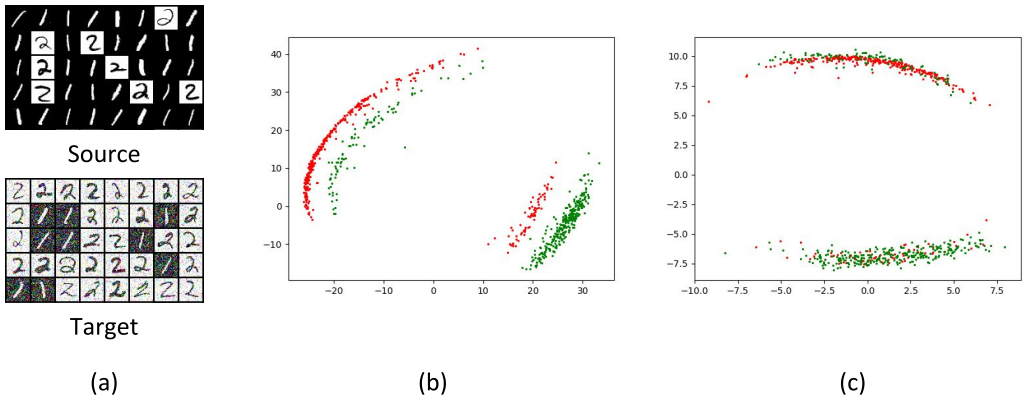


Figure 6.2: Domain adaptation for image denoising. (a) Samples from source and target domains. (b) Source and target embeddings learnt by the baseline model. (c) Source and target embeddings learnt by minimizing the proposed NW measure. In (b) and (c), red and green points indicate source and target samples, respectively.

As our baseline, we consider a model trained only on source using a quadratic reconstruction loss. Fig 6.2(b) shows source and target embeddings produced by this baseline. In this case, the source and the target embeddings are distant from each other. However, as shown in Fig 6.2(c), using the NW formulation, the distributions of source and target embeddings match closely (with estimated mode proportions) . We measure the ℓ_2 reconstruction loss of the target domain, $\epsilon_{rec,tgt} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{tgt}}\|G(F(\mathbf{x})) - \mathbf{x}\|_2^2$, as a quantitative evaluation measure. This value for different approaches is shown in Table 6.4. We observe that our method outperforms the compared approaches.

Table 6.4: $\epsilon_{rec,tgt}$ for an image denoising task.

Method	$\epsilon_{rec,tgt}$
Source only	0.31
Wasserstein	0.52
Normalized Wasserstein	0.18
Training on target (Oracle)	0.08

6.5 Normalized Wasserstein GAN

Recall from Sec. 2.2.1 that the goal of GANs is to learn a generative model that can synthesize samples that resemble the real data distribution. If the real distribution p_X is a mixture one, the proposed normalized Wasserstein measure (6.3) can be used for learning an effective generative model. Instead of estimating a single generator $\mathbf{G}$ as done in standard GANs, we estimate a mixture distribution $p_{\mathbf{G},\pi}$ using the proposed NW measure. We refer to this GAN as the *Normalized Wasserstein GAN* (or NWGAN) formulated as the following optimization:

$$\min_{\mathbf{G},\pi} \quad \mathcal{W}_N(p_{data}, p_{\mathbf{G},\pi}). \tag{6.10}$$

In this case, the NW distance simplifies as

$$\begin{aligned}
&\min_{\mathbf{G},\pi} \mathcal{W}_N(p_{data}, p_{\mathbf{G},\pi}) \\
&= \min_{\mathbf{G},\pi} \min_{\mathbf{G}',\pi^{(1)},\pi^{(2)}} \mathcal{W}(p_{data}, \mathbb{P}_{\mathbf{G}',\pi^{(1)}}) + \mathcal{W}(p_{\mathbf{G},\pi}, p_{\mathbf{G}',\pi^{(2)}}) \\
&= \min_{\mathbf{G},\pi} \mathcal{W}(p_{data}, p_{\mathbf{G},\pi}).
\end{aligned} \tag{6.11}$$

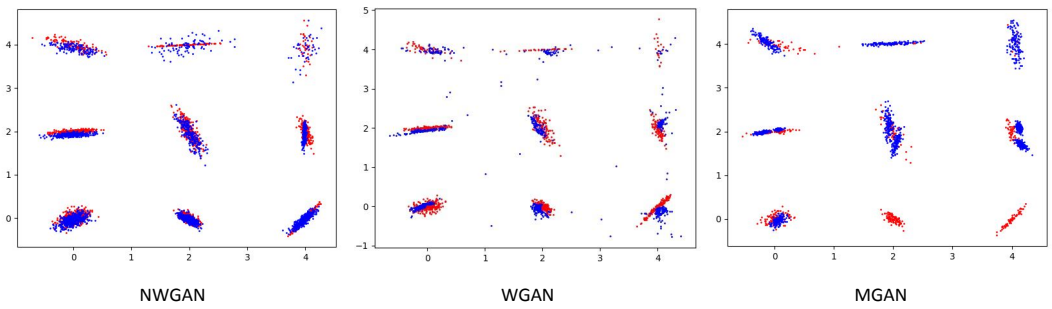


Figure 6.3: Mixture of Gaussian experiments. In all figures, red points indicate samples from the real data distribution while blue points indicate samples from the generated distribution. NWGAN is able to capture rare modes in the data and produces a significantly better generative model than other methods.

There are a few differences between the proposed NWGAN and the existing GAN architecures. The generator in the proposed NWGAN is a mixture of k models, each producing π_i fraction of generated samples. We select k a priori based on the application domain while π is computed within the NW distance optimization. Modeling the generator as a mixture of k neural networks has also been investigated in some recent works [121, 122]. However, these methods assume that the mixture proportions π are known beforehand, and are held fixed during the training. In contrast, our approach is more general as the mixture proportions are also optimized. Estimating mode proportions have several important advantages: (1) we can estimate rare modes, (2) an imbalanced dataset can be re-normalized, (3) by allowing each $\mathbf{G}_i$ to focus only on one part of the distribution, the quality of the generative model can be improved while the complexity of the generator can be reduced. In the following, we highlight these properties of NWGAN on different datasets.

6.5.1 Mixture of Gaussians

First, we present the results of training the NWGAN on a two dimensional mixture of Gaussians. The input data is a mixure of 9 Gaussians, each centered at a vertex of a 3×3 grid as shown in Figure 6.3. The mean and the covariance matrix for each mode are randomly chosen. The mode proportion for mode i is chosen as $\pi_i = \frac{i}{45}$ for $1 \leq i \leq 9$.

Generations produced by NWGAN using $k = 9$ affine generator models on this dataset is shown in Figure 6.3. We also compare our method with WGAN [36] and MGAN [121]. Since MGAN does not optimize over π, we assume uniform mode proportions ($\pi_i = 1/9$ for all i). To train WGAN, a non-linear generator function is used since a single affine function cannot model a mixture of Gaussian distribution.

To evaluate the generative models, we report the following quantitative scores: (1) the average mean error which is the mean-squared error (MSE) between the mean vectors of real and generated samples per mode averaged over all modes, (2) the average covariance error which is the MSE between the covariance matrices of real and generated samples per mode averaged over all modes, and (3) the π estimation error which is the normalized MSE between the π vector of real and generated samples. Note that computing these metrics require mode assignments for generated samples. This is done based on the closeness of generative samples to the ground-truth means.

We report these error terms for different GANs in Table 6.5. We observe that the proposed NWGAN achieves best scores compared to the other two approaches.

Table 6.5: Quantitative Evaluation on Mixture of Gaussians.

Method	Avg. μ error	Avg. Σ error	π error
WGAN	0.007	0.0003	0.0036
MGAN	0.007	0.0002	0.7157
NWGAN	**0.002**	**0.0001**	**0.0001**

Also, from Figure 6.3, we observe that the generative model trained by MGAN misses some of the rare modes in the data. This is because of the error induced by assuming fixed mixture proportions when the ground-truth π is non-uniform. Since the proposed NWGAN estimates π in the optimization, even rare modes in the data are not missed. This shows the importance of estimating mixture proportions specially when the input dataset has imbalanced modes.

6.5.2 A Mixture of CIFAR-10 and CelebA

One application of learning mixture generative models is to disentangle the data distribution into multiple components where each component represents one mode of the input distribution. Such disentanglement is useful in many tasks such as clustering. To test the effectiveness of NWGAN in performing such disentanglement, we consider a mixture of $50,000$ images from CIFAR-10 and $100,000$ images from CelebA [123] datasets as our input distribution. All images are reshaped to be 32×32.

To highlight the importance of optimizing mixture proportion to produce disentangled generative models, we compare the performance of NWGAN with a variation of NWGAN where the mode proportion π is held fixed as $\pi_i = \frac{1}{k}$ (the uniform

Figure 6.4: Sample generations of NWGAN with $k = 2$ on a mixture of CIFAR-10 and CelebA datasets for fixed and optimized π's. When π is fixed, one of the generators produces a mix of CIFAR and CelebA generative images (boxes in red highlight some of the CelebA generations in the model producing CIFAR+CelebA). However, when π is optimized, the model produces disentangled representations.

distribution). Sample generations produced by both models are shown in Figure 6.4. When π is held fixed, the model does not produce disentangled representations (in the second mode, we observe a mix of CIFAR and CelebA generative images.) However, when we optimize π, each generator produces distinct modes.

6.6 Ablation: Choosing the number of modes

As discused in Section 6.3, choosing the number of modes (k) is crucial for computing NW measure. While this information is available for tasks such as domain adaptation, it is unknown for others like generative modeling. In this section, we experimentally validate our theoretically justified algorithm for estimating k. Consider the mixture of Gaussian dataset with $k = 9$ modes presented in Section 6.5.1. On this dataset, the NWGAN model (with same architecture as that used in Section 6.5.1) was trained with varying number of modes k. For each setting, the NW

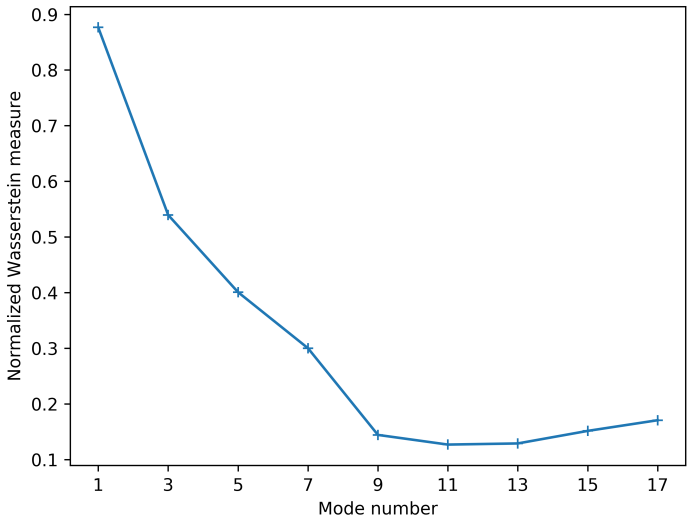


Figure 6.5: Choosing k: Plot of NW measure vs number of modes.

measure between the generated and real data distribution is computed and plotted in Fig 6.5. We observe that $k = 9$ satisfies the condition discussed in Theorem 4: optimal k^* is the smallest k for which $\mathcal{W}_N(k)$ is small, $\mathcal{W}_N(k-1) - \mathcal{W}_N(k)$ is large, and $\mathcal{W}_N(k)$ saturates after k^*.

6.7 Conclusion

In this chapter, we showed that Wasserstein distance, due to its marginal constraints, can lead to undesired results when when applied on imbalanced mixture distributions. To resolve this issue, we proposed a new distance measure called the Normalized Wasserstein. The key idea is to optimize mixture proportions in the distance computation, effectively normalizing mixture imbalance. We demonstrated the usefulness of NW measure in domain adaptation and generative model-

ing. Strong empirical results on all three problems highlight the effectiveness of the proposed distance measure.

Part III

Domain Generalization

Chapter 7: MetaReg: Towards Domain Generalization using Meta-Regularization

7.1 Introduction

Existing machine learning algorithms including deep neural networks achieve good performance in cases where the training and the test data are sampled from the same distribution, but fail to perform well on out-of-distribution inputs. As discussed in the previous part of the dissertation, one approach for improving performance on such out-of-distribution inputs is *domain adaptation*, in which we utilize unsupervised target domain data to adapt our models. However, access to unlabeled target data might not be available in many real-world settings. Hence, it is critical to design systems that can generalize to unseen variations in data. This problem, also called *domain generalization*, is the focus of this chapter.

In domain generalization, we are provided with multiple labeled source datasets. The task is to train a robust model using variations in the source domains so that it generalizes well to novel target domains. Domain generalization is a much harder problem than domain adaptation as we assume no access to the target information. Instead, the variations in multiple source domains are utilized to generalize to novel

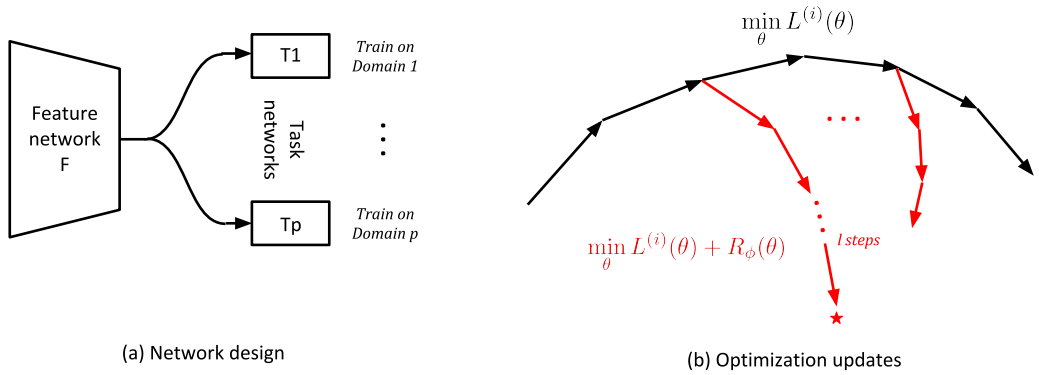


Figure 7.1: Illustration of the proposed approach. Figure (a) depicts the network design - We employ a shared feature network F and p task networks $\{T_i\}_{i=1}^{p}$. Each task network T_i is trained only on the data from domain i, and the shared network F is trained on all p source domains. The figure on the right illustrates the optimization updates. At each iteration we sample a pair of domains (i, j) from the training set. The black arrows are the SGD updates of the task network T_i trained on domain i. From each point in the black path, we take l gradient steps using the regularized loss and the samples from domain i to reach a new point $*$. We then compute the loss on domain j at $*$. The regularizer parameters ϕ are updated so that this meta-loss is minimized. This ensures that the task network T_i trained with the proposed regularizer generalizes to domain j.

test distributions.

One popular approach to improve the generalization of a parametric model is to introduce regularization in the loss function [124]. Several regularization schemes have been proposed for neural networks including weight decay [125], Dropout [126], DropConnect [127], batch normalization [128], etc. While these schemes have been shown to reduce test error on samples drawn from the same training distribution, they do not generalize when there is a training-test distribution mismatch. In this chapter, we investigate if a regularization function can be learnt for neural networks

with the objective of generalizing to out-of-distribution shifts.

Designing such regularizers for achieving cross-domain generalization is a challenging problem. The difficulty in mathematically modeling domain shifts makes it hard to design hand-crafted regularizers. Instead, we take a data-driven approach where we aim to learn the regularization function using the variability in the source domains. We cast the problem of learning regularizers in a *learning to learn*, or *meta-learning* framework, which has received a resurgence in interest recently with applications including few-shot learning [129, 130] and learning optimizers [131, 132]. Similar to [129], we follow an episodic paradigm where at each iteration, we sample an episode comprising meta-train and meta-test data such that the domains contained in meta-train and meta-test sets are disjoint. The objective is then to train the regularizer such that k steps of gradient descent using the meta-train data results in a decrease in meta-test loss. This procedure is repeated for multiple episodes sampled from the source dataset. After the regularizer is trained, we fine-tune a new model on the entire source dataset using the trained regularizer.

The primary contribution of this work is that we propose a scheme for learning regularization functions that enable domain generalization. We show how the objective of domain generalization can be explicitly encoded in a regularization function, which can then be used to train models that are more robust to domain shifts. This framework is also scalable as the same regularizer can later be used to fine-tune on a larger dataset. Experiments indicate that our approach can learn regularizers that achieve good cross-domain generalization on benchmark domain generalization datasets.

7.2 Related work

Meta-learning: The concept of meta-learning (or) learning to learn has a long standing history, some of the earlier works include [133, 134]. Recently, there has been a lot of interest in applying such strategies for deep neural networks. One interesting application is the problem of learning the optimization updates of neural networks by casting it as a policy learning problem in a Markov decision process [131, 132]. Few-shot learning is another problem where meta-learning strategies have been widely explored. Ravi and Larochelle [130] proposes an LSTM-based meta learner for learning the optimization updates of a few-shot classifier. Instead of learning the updates, Finn et al. [129] learns transferable weight representations that quickly adapts to a new task using only a few samples. Other recent applications that use meta learning include imitation learning [135], visual question answering [136], etc.

Domain Generalization: Unlike domain adaptation, domain generalization is a relatively less explored area of research. Muandet et al. [137] proposes domain invariant component analysis, a kernel-based algorithm for minimizing the differences in the marginal distributions of multiple domains. Ghifary et al. [138] attempts to learn a domain-invariant feature representation by using multi-view autoencoders to perform cross domain reconstructions. The method in Khosla et al. [139] decomposes the parameters of a model (SVM classifier) into domain specific and domain invariant components, and uses the domain invariant parameters to make predic-

tions on the unseen domain. Li et al. [140] extends this idea to decompose the weights of deep neural networks using multi-linear model and tensor decomposition.

Finn et al. [129] recently proposed a model agnostic meta-learning procedure for the few shot learning problems. The objective of their approach (MAML) is to find a good initialization θ such that few gradient steps from θ results in a good task specific network. The focus of the MAML is to adapt quickly in few shot settings. Recently, [3] proposed a meta learning based approach (MLDG) extending MAML to the domain generalization problem. This approach has the following limitations - the objective function of MAML is more suited for fast task adaptation for which it was originally proposed. In domain generalization however, we do not have access to samples from a new domain, and so a MAML-like objective might not be effective. The second issue is scalability - it is hard to scale MLDG to deep architectures like Resnet [5]. Our approach attempts to tackle both these problems - (1) We explicitly encode the objective of domain generalization in our episodic training procedure by using a regularizer to go from a task specific representation to a task agnostic representation at each episode. (2) We make our approach scalable by freezing the feature network and performing meta learning only on the task network. This enables us to use our approach to train deeper models like Resnet-50. A similar approach for training meta-learning algorithms in feature space has been explored in a recent work of Zhou et al. [141].

7.3 Method

7.3.1 Problem Setup

Let $\mathcal{X}$ denote the instance space (which can be images, text, etc.) and $\mathcal{Y}$ denote the label space. Domain generalization involves data sampled from p source distributions and q target distributions, each containing data for performing the same task. Classification tasks are considered in this work. Hence, $\mathcal{Y}$ is the discrete set $\{1, 2, \ldots n_c\}$, where n_c denotes the number of classes. Let $\mathcal{D}_i = \{(\mathbf{x}_j^{(i)}, y_j^{(i)})\}_{j=1}^{n_i}$ represent the dataset corresponding to the i^{th} distribution. In the rest of the chapter, $\mathcal{D}_i$ is also referred to as the i^{th} *domain*. Note that every $\mathcal{D}_i$ shares the same data and label space i.e., each $\mathbf{x}_j^{(i)} \in \mathcal{X}$ and $y_j^{(i)} \in \mathcal{Y}$. However, each of the $p+q$ domains contain varied domain statistics. The objective is to train models on the p source domains so that they generalize well to the q novel target domains.

We are interested in training a parametric model $M_\Theta : \mathcal{X} \to \mathcal{Y}$ using data only from the p source domains. In this work, we consider M_Θ to be a deep neural network. We decompose the network M into a feature network F and a task network T (i.e) $M_\Theta(\mathbf{x}) = (T_\theta \circ F_\psi)(\mathbf{x})$, where $\Theta = \{\psi, \theta\}$. Here, ψ denotes the weights of the feature network F, and θ denotes the weights of the task network. The output of $M_\Theta(\mathbf{x})$ is a vector of dimension n_c with i^{th} entry denoting the probability that the instance $\mathbf{x}$ belongs to the class i. Standard neural network training involves

minimizing the cross entropy loss function given by Eq (7.1)

$$\mathcal{L}(\psi, \theta) = \mathbb{E}_{(\mathbf{x},y)\sim D}[-\mathbf{y}.\log(M_\Theta(\mathbf{x}))] = \sum_{i=1}^{p}\sum_{j=1}^{N_i} -\mathbf{y}_j^{(i)}.\log(M_\Theta(\mathbf{x}_j^{(i)})) \tag{7.1}$$

Here, $\mathbf{y}_j^{(i)}$ is the one-hot representation of the label $y_j^{(i)}$ and '.' denotes the dot product between two vectors. The above loss function does not take into account any factor that models domain shifts, so generalization to a new domain is not expected. To accomplish this, we propose using a regularizer $R(\psi, \theta)$. The new loss function then becomes $L_{reg}(\psi, \theta) = \mathcal{L}(\psi, \theta) + R(\psi, \theta)$. The regularizer $R(\psi, \theta)$ should capture the notion of domain generalization (i.e) it should enable generalization to a new distribution with varied domain statistics. Designing such regularizers is hard in general, so we propose to learn it using meta learning.

7.3.2 Learning the regularizer

In this work, we model the regularizer R as a neural network parametrized by weights ϕ. Moreover, the regularization is applied only on the parameters θ of the task network to enable scalable meta-learning. So, the regularizer is denoted as $R_\phi(\theta)$ in the rest of the chapter. We now discuss how the parameters of the regularizer $R_\phi(\theta)$ are estimated. In this stage of the training pipeline, the neural network architecture consists of a feature network F and p task networks $\{T_i\}_{i=1}^p$ (with parameters of T_i denoted by θ_i) as shown in Fig. 7.1. Each T_i is trained only on the samples from domain i and F is the shared network trained on all p source domains. The reason for using p task networks is to enforce domain-specificity in

the models so that the regularizer can be trained to make them domain-invariant.

We now describe the procedure for learning the regularizer:

- **Base model updates:** We begin by training the shared network F and p task networks $\{T_i\}_{i=1}^{p}$ using supervised classification loss $\mathcal{L}(\psi, \theta)$ given by Eq (7.1). Note that there is no regularization in this step. Let the network parameters at the k^{th} step of this optimization be denoted as $[\psi^{(k)}, \theta_1^{(k)}, \ldots \theta_p^{(k)}]$.

- **Episode creation:** To train $R_\phi(\theta)$, we follow an episodic training procedure similar to Li et al. [3]. Let a, b be two randomly chosen domains from the training set. Each episode contains data partitioned into two subsets - (1) m_1 labeled samples from domain a denoted as *metatrain* set and (2) m_2 labeled samples from domain b denoted as *metatest* set. The domains contained in both the sets are disjoint (i.e) $a \neq b$, and the data is sampled only from the source distributions (i.e) $a, b \in \{1, 2, \ldots p\}$.

- **Regularizer updates** At iteration k, a new task network T_{new} is initialized with $\theta_a^{(k)}$ - the base model's task network parameters of the a^{th} domain at iteration k. Using the samples from the *metatrain* set (which contains domain a), l steps of gradient descent is performed with the regularized loss function $\mathcal{L}_{reg}(\psi, \theta)$ on T_{new}. Let $\hat{\theta}_a^{(k)}$ denote the parameters of T_{new} after these l gradient steps. We treat each update of the network T_{new} as a separate variable in the computational graph. $\hat{\theta}_a^{(k)}$ then depends on ϕ through these l gradient steps. The unregularized loss on the *metatest* set computed using T_{new} (with parameters $\hat{\theta}_a^{(k)}$) is then minimized with respect to the regularizer parameters

ϕ. Each regularizer update unrolls through the l gradient steps as $\hat{\theta}_a^{(k)}$ depends on ϕ through the l gradient steps. This entire procedure can be expressed by the following set of equations:

$$\begin{aligned}
\beta^1 &\leftarrow \theta_a^{(k)} \\
\beta^t &= \beta^{t-1} - \alpha \nabla_{\beta^{t-1}} \left[\mathcal{L}^{(a)}(\psi^{(k)}, \beta^{t-1}) + R_\phi(\beta^{t-1}) \right] \qquad \forall t \in \{2, \dots l\} \quad (7.2) \\
\hat{\theta}_a^{(k)} &= \beta^l \\
\phi^{(k+1)} &= \phi^{(k)} - \alpha \nabla_\phi \mathcal{L}^{(b)}(\psi^{(k)}, \hat{\theta}_a^{(k)})|_{\phi=\phi^{(k)}} \quad (7.3)
\end{aligned}$$

Here, $\mathcal{L}^{(i)}(\psi, \theta_{new}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D_i}[-\mathbf{y}.\log(T_{\theta_{new}}(F_\psi(\mathbf{x})))]$ (i.e) the loss of task network T_{new} on samples from domain i, and α is the learning rate. Eq (7.2) represents l steps of gradient descent from the initial point $\theta_a^{(k)}$ using samples from *metatrain* set, with β_t denoting the output at the t^{th} step. Eq (7.3) is the meta-update step for updating the parameters of the regularizer. This update ensures that l steps of gradient descent using the regularized loss on samples from domain a results in task network a performing well on domain b. It is important to note that the dependence of ϕ on $\hat{\theta}_a^{(k)}$ comes from the l gradient steps performed in Eq. 7.2. So, the gradients of ϕ propagates through these l unrolled gradient steps.

Since the same regularizer $R_\phi(\theta)$ is trained on every (a, b) pair, the resulting regularizer we learn captures the notion of domain generalization. Please refer to Fig. 7.1 for a pictoral description of the meta-update step. The entire algorithm is

given in Algorithm 5

7.3.3 Training the final model

Once the regularizer is learnt, the regularization parameters ϕ are frozen and the final task network initialized from scratch is trained on all p source domains using the regularized loss function $\mathcal{L}_{reg}(\psi, \theta)$. The network architectures consists of just one $F - T$ pair. In this chapter, we use weighted ℓ_1 loss as our regularization function, (i.e) $R_\phi(\theta) = \sum_i \phi_i |\theta_i|$. The weights of this regularizer are estimated using the meta-learning procedure discussed above. However, our approach is general and can be extended to any class of regularizers (refer to Section. 7.5). The use of weighted ℓ_1 loss can be interpreted as a learnable weight decay mechanism - Weights θ_i for which ϕ_i is positive will be decayed to 0 and those for which ϕ_i is negative will be boosted. By using our meta-learning procedure, we select a common set of weights that achieve good cross-domain generalization across every pair of source domains (a, b).

7.3.4 Summary of the training pipeline

The feature network is first trained using combined data from all source domains, and is kept frozen in the rest of training. The regularizer parameters are then estimated using the meta-learning procedure described in the previous section. As the individual task networks are updated on their respective source domain data, the regularizer updates are derived from each point of this SGD path with the objective

Algorithm 5 MetaReg training algorithm

Require: n_{iter}: number of training iterations
Require: α_1, α_2: Learning rate hyperparameters

for t in $1 : n_{iter}$ **do**
  for i in $1 : p$ **do**
    Sample n_b labeled images $\{(x_j^{(i)}, y_j^{(i)}) \sim D_i\}_{j=1}^{n_b}$
    Perform supervised classification updates:
    $\psi^{(t)} \leftarrow \psi^{(t-1)} - \alpha_1 \nabla_{\psi} L^{(i)}(\psi^{(t-1)}, \theta_i^{(t-1)})$
    $\theta_i^{(t)} \leftarrow \theta_i^{(t-1)} - \alpha_1 \nabla_{\theta_i} L^{(i)}(\psi^{(t-1)}, \theta_i^{(t-1)})$
  end for
  Choose $a, b \in \{1, 2, \ldots p\}$ randomly such that $a \neq b$
  $\beta^1 \leftarrow \theta_a^{(t)}$
  for $i = 2 : l$ **do**
    Sample *metatrain* set $\{(x_j^{(a)}, y_j^{(a)}) \sim D_a\}_{j=1}^{n_b}$
    $\beta^i = \beta^{i-1} - \alpha_2 \nabla_{\beta^{i-1}} [L^{(a)}(\psi^{(t)}, \beta^{i-1}) + R_{\phi}(\beta^{i-1})]$
  end for
  $\hat{\theta}_a^{(t)} = \beta_l$
  Sample *metatest* set $\{(x_j^{(b)}, y_j^{(b)}) \sim D_b\}_{j=1}^{n_b}$
  Perform meta-update for regularizer $\phi^{(t)} = \phi^{(t-1)} - \alpha_2 \nabla_{\phi} L^{(b)}(\psi^{(t)}, \hat{\theta}_a^{(t)})|_{\phi=\phi^{(t)}}$
end for

of cross-domain generalization (refer Alg. 5). To learn the regularizer effectively at the early stages of the task network updates, replay memory is used where the regularizer updates are periodically derived from the early stages of the task networks' SGD paths. The learnt regularizer is used in the final step of the training process where a single $F - T$ network is trained using the regularized cross-entropy loss.

7.4 Experiments

In this section, we describe the experimental validation of our proposed approach. We perform experiments on two benchmark domain generalization datasets - Multi-domain image recognition using PACS dataset [140] and sentiment classification using Amazon Reviews dataset [142]. More details can be found in [143].

7.4.1 PACS dataset

PACS dataset is a recently proposed benchmark dataset for domain generalization. This dataset contains images from four domains - *Photo*, *Art painting*, *Cartoon* and *Sketch*. Following [3], we perform experiments on four settings: In each setting, one of the four domains is treated as the unseen target domain, and the model is trained on the other three source domains.

Alexnet The first set of experiments is based on the Alexnet [29] model pretrained on Imagenet. The feature network F comprises of the top layers of Alexnet model till $pool5$ layer, while the task network T contains $fc6$, $fc7$ and $fc8$ layers. For the regularizer network, we used weighted ℓ_1 loss (i.e) $R_\phi(\theta) = \sum_i \phi_i |\theta_i|$, where ϕ_i are the parameters estimated using meta-learning. In all our experiments, *Baseline* setting denotes training a neural network (Alexnet in this case) on all of the source domains without performing any domain generalization. Other comparison methods include Multi-task Autoencoders (MTAE) [138], Domain Separation Networks (DSN) [144], Artier Domain Generalization (DBA-DG) [140] and MLDG [3]. While some of these methods were originally proposed for domain adaptation, they were adapted to the domain generalization problem as done in Li et al. [3].

All our models are trained using the SGD optimizer with learning rate $5e-4$ and a batch size of 64. This is in accordance with the setup used in Li et al. [3]. Table 7.1 presents the results of our approach along with other comparison methods. We observe that our method obtains a performance improvement of 3.34% over the

Table 7.1: Cross-domain recognition accuracy (in %) averaged over 5 runs on PACS dataset using Alexnet architecture. For the baseline setting, the numbers on the parenthesis indicate the baseline performance as reported by Li et al. [3]

Method	*Art painting*	*Cartoon*	*Photo*	*Sketch*	Average
Baseline	67.21 ± 0.72 (64.91)	66.12 ± 0.51 (64.28)	88.47 ± 0.63 (86.67)	55.32 ± 0.44 (53.08)	69.28 (67.24)
D-MTAE ([138])	60.27	58.65	**91.12**	47.68	64.48
DSN ([144]	61.13	66.54	83.25	58.58	67.37
DBA-DG ([140])	62.86	66.97	89.50	57.51	69.21
MLDG ([3])	66.23	66.88	88.0	58.96	70.01
MetaReg (*Ours*)	**69.82** ± 0.76	**70.35** ± 0.63	91.07 ± 0.41	**59.26** ± 0.31	**72.62**

baseline, thus achieving the state-of-the-art performance on this dataset.

textbfResnet One disadvantage with approaches like MLDG [3] is that it requires differentiating through k steps of optimization updates, and this might not be scalable to deeper architectures like Resnet. Even our approach requires a similar optimization process. However, unlike Li et al. [3], we perform meta-learning only on the task network. Since the task network is much shallower than the feature network, our approach is scalable even to some of the contemporary deep architectures. In this section, we show experiments using two such architectures - Resnet18 and Resnet 50.

We use the Resnet-18 and Resnet-50 models pretrained on ImageNet as our feature network, and the last fully connected layer as our task network. Similar to the previous experiment, we used weighted ℓ_1 loss as our class of regularizers. All models were trained using SGD optimizer with a learning rate of 0.001 and momentum 0.9. The hyper-parameters α_1 and α_2 are both set as 0.001. The results of our experiments are reported in Table. 7.2. Our method performs better than baseline in both settings. It is important to note that the baseline numbers for

Table 7.2: Cross-domain recognition accuracy (in %) averaged over 5 runs on PACS dataset using Resnet architectures

Method	*Art painting*	*Cartoon*	*Photo*	*Sketch*	Average
Resnet-18					
Baseline	79.9 ± 0.22	75.1 ± 0.35	95.2 ± 0.18	69.5 ± 0.37	79.9
Metareg (*Ours*)	**83.7** ± 0.19	**77.2** ± 0.31	**95.5** ± 0.24	**70.3** ± 0.28	**81.7**
Resnet-50					
Baseline	85.4 ± 0.24	77.7 ± 0.31	**97.8** ± 0.17	69.5 ± 0.42	82.6
Metareg (*Ours*)	**87.2** ± 0.13	**79.2** ± 0.27	97.6 ± 0.31	**70.3** ± 0.18	**83.6**

Resnet architectures are much higher than that of Alexnet. Even on such stronger baselines, our method gives performance improvement.

7.4.2 Sentiment Classification

In this section, we perform experiments on the task of sentiment classification on *Amazon reviews* dataset as pre-processed by [145]. The dataset contains reviews of products belonging to four domains - *books*, *DVD*, *electronics* and *kitchen appliances*. The differences in textual description of the reviews each of these product categories manifests as domain shift. Following [71], we use unigrams and bigrams as features resulting in 5000 dimensional vector representations. The reviews are assigned binary labels - 0 if the rating of the product is upto 3 stars, and 1 if the rating is 4 or 5 stars.

We conduct 4 cross-domain experiments - in each setting one of the four domains is treated as the unseen test domain, and the other three domains are used as source domains. Similar to [71], we used a neural network with one hidden layer

Table 7.3: Cross domain classification accuracy (x %) averaged over 10 runs on Amazon Reviews dataset.

Method	*Books*	*DVD*	*Electronics*	*Kitchen*	Average
Baseline	75.5 ± 0.52	79.0 ± 0.37	83.7 ± 0.44	84.7 ± 0.63	80.7
Metareg (*Ours*)	**76.1** ± 0.41	**79.6** ± 0.32	**83.9** ± 0.28	**85.1** ± 0.43	**81.2**

(with 100 neurons) as our task network. All models were trained using an SGD optimizer with learning rate 0.01 and momentum 0.9 for 5000 iterations. The results of our experiments are reported in the Table. 7.3. Since there is significant variation in performance over runs, each experiment was repeated 10 times with different random weight initialization and averages of these 10 runs are reported. We observe that our method performs better than the baseline in all of the settings. However, the performance improvement is less compared to the previous experiments. This is because of the nature of the problem and the architectural choice. We would like to point out that even domain adaptation methods that make use of unlabeled target data achieve similar gains in performance [71] in this dataset.

7.5 Ablation Study

For all the ablation experiments except 7.5.3, we use the Resnet-18 model as our neural network architecture, and Art-painting setting in PACS dataset as our experimental setting, (i.e) we use *Art painting* domain as the test domain, and *Cartoon*, *Photo* and *Sketch* as source domains.

Table 7.4: Effect of different classes of regularization functions.

Baseline	DropConnect [127]	Default ℓ_1	Weighted ℓ_1	Weighted ℓ_2	2 layer NN
79.9	80.1	79.7	83.7	83.2	83.3

7.5.1 Class of Regularizers

In this experiment, we study the effect of different regularizers on the performance of our approach. We experimented on the following class of regularizers: (1) Weighted ℓ_1 loss: $R_\phi(\theta) = \sum_i \phi_i|\theta_i|$, (2) Weighted ℓ_2 loss: $R_\phi(\theta) = \sum_i \phi_i\theta_i^2$, and (3) Two layer neural network: $R_\phi(\theta) = \phi^{(2)T}(ReLU(\phi^{(1)T}\theta))$. The performance of these regularizers are reported in Table. 7.4. We observe that the Weighted ℓ_1 regularizer performs the best among the three. Also, we observed that training networks with the weighted ℓ_1 regularizer lead to better convergence and stability in performance compared to the other two. We also compare our approach with two other schemes: (1) DropConnect [127] and (2) Default ℓ_1 regularization, which is Weighted ℓ_1 regularization where the weights $\phi_i = 1$. We observe that both these schemes do not improve the baseline performance.

7.5.2 Delayed Data Acquisition

In all of the previous experiments, we assumed that the entire training data is available from the start of the training process. But consider a more general setting where we train our model on some initial data, but more data gets available over time. Is it possible make use of the newly available data to improve our models

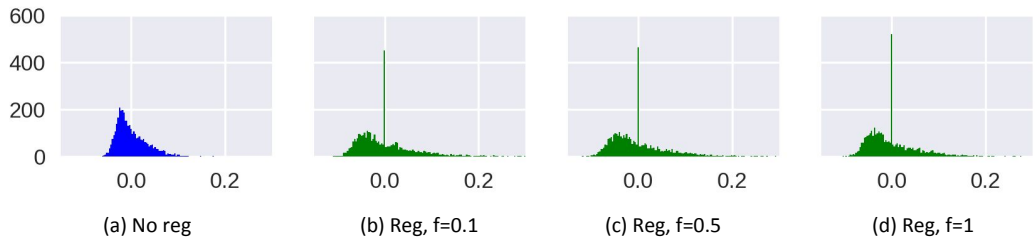


Figure 7.2: Histogram of the weights learnt by the task network. "No reg" corresponds to the network without regularizaton, and "Reg, f=x" corresponds to the regularized network, where the regualrizer R is trained only on $x\%$ of the data.

without having to perform meta-learning again? We propose the following solution: Train the feature network, task network and regularizer on the initial dataset. On the new data, finetune the task network and feature network using the regularizer trained on the initial data. Note that, we do not perform meta learning again on the new data, and so this is computationally efficient since meta-learning procedure incurs significant overhead over a regular finetuning process. With approaches like MLDG [3], meta-learning has to be performed even on the new data.

We simulate these experimental conditions as follows: In each setting, we consider a fraction f of the PACS dataset as our intial dataset on which our model and the regularizer are trained. We then finetune our model on the remaining data using the trained regularizer. The performance of these models on the test set are shown in Table. 7.5. We observe that there is little drop in performance for all f values. Our approach is able to learn good regularizers even with 10% of the entire dataset.

Table 7.5: Experiments for training models on less data.

Data fraction f	0.1	0.2	0.3	0.4	0.5	1.0
Accuracy (in %)	82.86	83.11	83.42	83.62	83.60	83.71

Table 7.6: Effect of cross-domain generalization with varying number of layers regularized on PACS dataset using Alexnet model. Cartoon is used as the test domain.

Layers regularized	None	fc_8	$fc_7 + fc_8$	$fc_6 + fc_7 + fc_8$
Accuracy (in %)	66.12	67.31	70.10	70.35

7.5.3 Effect of the number of layers regularized

In our training paradigm, the neural network is decomposed into feature and task network, and the meta-regularization is performed only on the task network. Deciding this feature/task network split is a design choice which needs to be understood. The effect of domain generalization performance on varying the number of layers is reported in Table 7.6. This experiment is performed using the Alexnet architecture on PACS dataset with *Cartoon* as the target domain. We observe that as the number of regularization layers increases, the generalization performance increases and saturates beyond a point.

7.5.4 Effect of the number of unrolling steps

In this experiment, we examine the effect of number of unrolling SGD updates in the meta learning process (effect of l) on the cross-domain generalization performance. We performed experiments with $l = 1, 2, 3, 4, 5$. The results are reported in

Table 7.7: Effect of number of unrolling steps in Metareg updates.

Number of inner steps l	*Accuracy (in %)*
1	83.41
2	83.57
3	83.71
4	83.66
5	83.72

Table 7.7. Even with a 1- step update, our method achieves good performance improvement compared to the baseline. Performance keeps increasing with increasing l and saturates after $l = 3$.

7.5.5 When does MetaReg work?

Understanding failure cases is important as it provides better insight on the workings of our approach. We study this on Rotated-MNIST dataset – dataset with MNIST digits rotated by $0°, 10°, 20°, 30°$ and $60°$, each of which correponds to one domain. The benefit of using this controlled dataset is that it is easier to quantify domain shifts. For instance, $10°$ rotations are closer to $0°$ than $75°$. In our experimets, the datasets corresponding to $0°$, $10°$ and $30°$ were used as source domains, and $20°$ and $60°$ rotations are used as target domains. A 2-layer MLP ($784 \rightarrow 128 \rightarrow 128 \rightarrow 10$) is used as the task network, no feature network was used. So, the entire network is regularized. Table 7.8 presents the results of the cross-domain generalization on these two target domains.

We observe that there is an improvement in the performance on the $20°$ domain and a drop in performance on the $60°$ domain. This is because the $60°$ domain

Table 7.8: Effect of cross-domain generalization on the extent of domain shift.

Method	*Accuracy (in %)* on 20° domain	*Accuracy (in %)* on 60° domain
Baseline	95.9	**57.3**
MetaReg	**96.7**	56.8

presents much larger domain shift than the variations represented in the training set. This suggests that MetaReg works as long as the shifts encountered in the test set is similar to the variations captured in the training domains.

7.5.6 Visualizing the weights

We plot the histogram of the weights learnt by the task network with and without the use of our regularizer in Fig. 7.2. The following observations can be made: (1) For the network with regularization, there is a sharp peak at 0. This is because the weights θ_i for which ϕ_i are positive are decayed to 0. (2) The weights of the network with regularization has wider spread compared to the network without regularization. This is because the weights θ_i for which ϕ_i are negative are boosted, due to which certain weights have high values.

7.6 Conclusion and Future Work

In this chapter, we addressed the problem of domain generalization by using regularization. The task of finding the desired regularizer that captures the notion of domain generalization is modeled as a meta-learning problem. Experiments indicate that the learnt regularizers achieve good cross-domain generalization on the

benchmark domain generalization datasets. Some avenues for future work include scalable meta-learning approaches for learning regularization functions over convolutional layers while preserving the spatial dependency between the channels, and extending our approach to deep reinforcement learning problems.

Part IV

Robustness to Adversarial Shifts

Chapter 8: Instance Adaptive Adversarial Training

8.1 Introduction

Distributional shifts in machine learning can exist in several forms. The previous sections of this dissertation focused on natural shifts, which are distributional shifts that occur due to natural factors such as variations in environmental conditions, physics of image acquisition, etc. In this chapter, we focus on a different type of distributional shift, called *adversarial shifts*. In adversarial shifts, a malicious adversary creates noisy samples with the intention of breaking the machine learning system. The objective of the adversary is to create imperceptible noisy images that fool the machine learning model.

Neural networks are shown to be extremely sensitive to adversarial noise. Extremely tiny perturbations to network inputs may be imperceptible to the human eye, and yet cause major changes to outputs. Several papers have demonstrated the vulnerability of neural networks in white-box settings [146, 147, 148], black-box settings [149, 150], physical attacks [151], etc. This sensitivity is undesirable, especially as we deploy the models in safety-critical applications.

One of the most effective and widely used methods for hardening networks to small perturbations is "adversarial training" [148], in which a network is trained

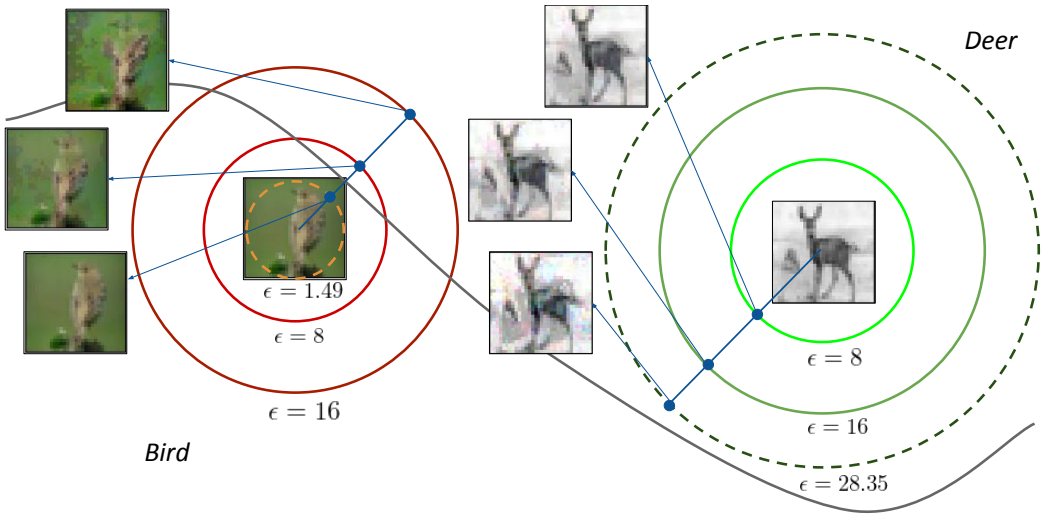


Figure 8.1: Overview of instance adaptive adversarial training. Samples close to the decision boundary (bird on the left) have nearby samples from a different class (deer) within a small L_p ball, making the constraints imposed by PGD-8 / PGD-16 adversarial training infeasible. Samples far from the decision boundary (deer on the right) can withstand large perturbations well beyond $\epsilon = 8$. Our adaptive adversarial training correctly assigns the perturbation radius (shown in dotted line) so that samples within each L_p ball maintain the same class.

using adversarially perturbed samples with a fixed perturbation size. By doing so, adversarial training typically tries to enforce that the output of a neural network remains nearly constant within an ℓ_p ball of every training input.

Despite its ability to increase robustness, adversarial training suffers from poor accuracy on clean (natural) test inputs. The drop in clean accuracy can be as high as 10% on CIFAR-10, and 15% on Imagenet [148, 152], making robust models undesirable in some industrial settings. The consistently poor performance of robust models on clean data has lead to the line of thought that there may be a fundamental trade-off between robustness and accuracy [153, 154], and recent theoretical results characterized this tradeoff [155, 156, 157].

In this work, we aim to understand and optimize the tradeoff between robustness and clean accuracy. More concretely, our objective is to improve the clean accuracy of adversarial training for a chosen level of adversarial robustness. Our method is inspired by the observation that the constraints enforced by adversarial training are *infeasible*; for commonly used values of ϵ, it is not possible to achieve label consistency within an ϵ-ball of each input image because the balls around images of different classes overlap. This is illustrated on the left of Figure 8.1, which shows that the ϵ-ball around a "bird" (from the CIFAR-10 training set) contains images of class "deer" (that do not appear in the training set). If adversarial training were successful at enforcing label stability in an $\epsilon = 8$ ball around the "bird" training image, doing so would come at the *unavoidable* cost of misclassifying the nearby "deer" images that come along at test time. At the same time, when training images lie far from the decision boundary (eg., the deer image on the right in Fig 8.1),

it is possible to enforce stability with large ϵ with no compromise in clean accuracy. When adversarial training on CIFAR-10, we see that $\epsilon = 8$ is too large for some images, causing accuracy loss, while being unnecessarily small for others, leading to sub-optimal robustness.

The above observation naturally motivates adversarial training with *instance adaptive* perturbation radii that are customized to each training image. By choosing larger robustness radii at locations where class manifolds are far apart, and smaller radii at locations where class manifolds are close together, we get high adversarial robustness where possible while minimizing the clean accuracy loss that comes from enforcing overly-stringent constraints on images that lie near class boundaries. As a result, instance adaptive training significantly improves the tradeoff between accuracy and robustness, breaking through the pareto frontier achieved by standard adversarial training. Additionally, we show that the learned instance-specific perturbation radii are interpretable; samples with small radii are often ambiguous and have nearby images of another class, while images with large radii have unambiguous class labels that are difficult to manipulate.

Parallel to our work, we found that Ding et al. [158] uses adaptive margins in a max-margin framework for adversarial training. Their work focuses on improving the adversarial robustness, which differs from our goal of understanding and improving the robustness-accuracy tradeoff. Moreover, our algorithm for choosing adaptive margins significantly differs from that of Ding et al. [158].

8.2 Background

Adversarial attacks are data items containing small perturbations that cause misclassification in neural network classifiers [146]. Popular methods for crafting attacks include the fast gradient sign method (FGSM) [159] which is a one-step gradient attack, projected gradient descent (PGD) [148] which is a multi-step extension of FGSM, the C/W attack [160], DeepFool [161], and many more. All these methods use the gradient of the loss function with respect to inputs to construct additive perturbations with a norm-constraint. Alternative attack metrics include spatial transformer attacks [162], attacks based on Wasserstein distance in pixel space [163], etc.

Defending against adversarial attacks is a crucial problem in machine learning. Many early defenses [164, 165, 166], were broken by strong attacks. Fortunately, *adversarially training* is one defense strategy that remains fairly resistant to most existing attacks.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denote the set of training samples in the input dataset. In this chapter, we focus on classification problems, hence, $y_i \in \{1, 2, \dots n_c\}$, where n_c denotes the number of classes. Let $F_\theta(\mathbf{x}) : \mathbb{R}^{c \times m \times n} \to \mathbb{R}^{n_c}$ denote a neural network model parameterized by θ. Classifiers are often trained by minimizing the cross entropy loss given by

$$\min_\theta \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \sim \mathcal{D}} -\mathbf{y}_i \big[\log(F_\theta(\mathbf{x}_i)) \big]$$

where $\mathbf{y}_i$ is the one-hot vector corresponding to the label y_i. In adversarial training, instead of optimizing the neural network over the clean training set, we use the adversarially perturbed training set. Mathematically, this can be written as the following *min-max* problem

$$\min_{\theta} \max_{\|\delta_i\|_\infty \leq \epsilon} \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \sim \mathcal{D}} -\mathbf{y}_i \big[\log(F_\theta(\mathbf{x}_i + \delta_i)) \big] \tag{8.1}$$

This problem is solved by an alternating stochastic method that takes minimization steps for θ, followed by maximization steps that approximately solve the inner problem using k steps of PGD. For more details, refer to Madry et al. [148].

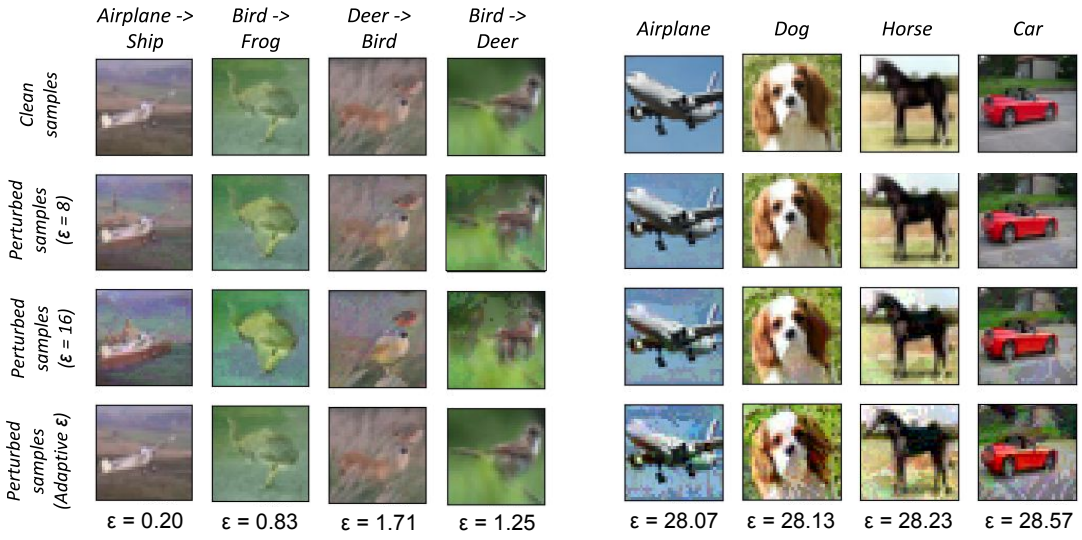
8.3 Instance Adaptive Adversarial Training

To remedy the shortcomings of uniform perturbation radius in adversarial training (Section 8.1), we propose *Instance Adaptive Adversarial Training* (IAAT), which solves the following optimization:

$$\min_{\theta} \max_{\|\delta_i\|_\infty < \epsilon_i} \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \sim \mathcal{D}} -\mathbf{y}_i \big[\log(F_\theta(\mathbf{x}_i + \delta_i)) \big] \tag{8.2}$$

Like vanilla adversarial training, we solve this by sampling mini-batches of images $\{\mathbf{x}_i\}$, crafting adversarial perturbations $\{\delta_i\}$ of size at most $\{\epsilon_i\}$, and then updating the network model using the perturbed images.

The proposed algorithm is distinctive in that it uses a different ϵ_i for each image $\mathbf{x}_i$. Ideally, we would choose each ϵ_i to be as large as possible without finding



(a) Samples from bottom 1% ϵ

(b) Samples from top 1% ϵ

Figure 8.2: Visualizing training samples and their perturbations. The left panel shows samples that are assigned small ϵ (displayed below images) during adaptive training. These images are close to class boundaries, and change class when perturbed with $\epsilon \geq 8$. The right panel show images that are assigned large ϵ. These lie far from the decision boundary, and retain class information even with very large perturbations. All ϵ live in the range $[0, 255]$.

Algorithm 6 Adaptive adversarial training algorithm

Require: n_{iter}: Number of training iterations, n_{warm}: Warmup period

Require: $PGD_k(\mathbf{x}, y, \epsilon)$: Function to generate $PGD-k$ adversarial samples with ϵ norm-bound

Require: ϵ_w: ϵ used in warmup

for t in $1 : n_{iter}$ **do**
  Sample a batch of training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n_{batch}} \sim \mathcal{D}$
  if $t < n_{warm}$ **then**
    $\epsilon_i = \epsilon_w$
  else
    Choose ϵ_i using Alg 7
  end if
  $\mathbf{x}_i^{adv} = PGD(\mathbf{x}_i, y_i, \epsilon_i)$
  $S_+ = \{i \mid F(\mathbf{x}_i)$ is correctly classified as $y_i\}$
  $S_- = \{i \mid F(\mathbf{x}_i)$ is incorrectly classified as $y_i\}$
  $\min_\theta \frac{1}{N_{batch}} \left[\sum_{i \in S_+} \mathcal{L}_{cls}(\mathbf{x}_i^{adv}, y^i) + \sum_{i \in S_-} \mathcal{L}_{cls}(\mathbf{x}_i, y_i) \right]$
end for

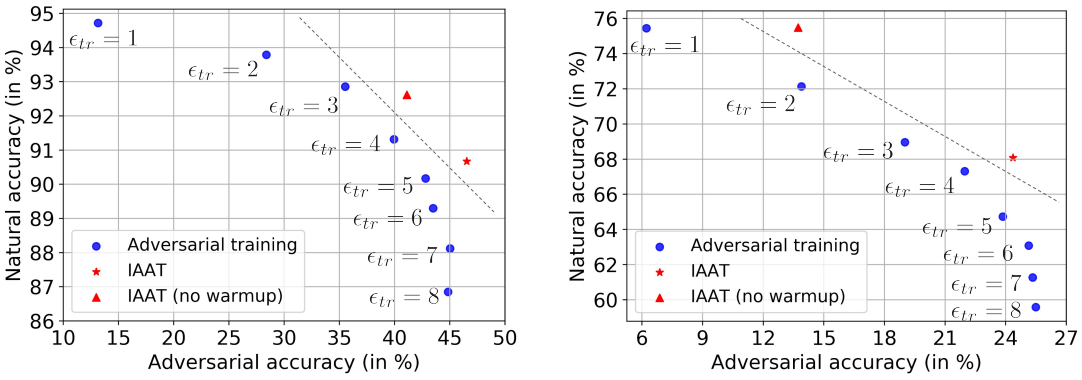
Algorithm 7 ϵ selection algorithm

Require: i: Sample index, j: Epoch index
Require: β: Smoothing constant, γ: Discretization for ϵ search.
Set $\epsilon_1 = \boldsymbol{\epsilon}_{mem}[j-1, i] + \gamma$
Set $\epsilon_2 = \boldsymbol{\epsilon}_{mem}[j-1, i]$
Set $\epsilon_3 = \boldsymbol{\epsilon}_{mem}[j-1, i] - \gamma$
if $F_\theta(PGD_k(\mathbf{x}_i, y_i, \epsilon_1))$ predicts as y_i **then**
    Set $\epsilon_i = \epsilon_1$
else if $F_\theta(PGD_k(\mathbf{x}_i, y_i, \epsilon_2))$ predicts as y_i **then**
    Set $\epsilon_i = \epsilon_2$
else
    Set $\epsilon_i = \epsilon_3$
end if
$\epsilon_i \leftarrow (1-\beta)\boldsymbol{\epsilon}_{mem}[j-1, i] + \beta\epsilon_i$
Update $\boldsymbol{\epsilon}_{mem}[j, i] \leftarrow \epsilon_i$
Return ϵ_i

images of a different class within the ϵ_i-ball around $\mathbf{x}_i$. Since we have no a-priori knowledge of what this radius is, we use a simple heuristic to update ϵ_i after each epoch. After crafting a perturbation for $\mathbf{x}_i$, we check if the perturbed image was a successful adversarial example. If PGD succeeded in finding an image with a different class label, then ϵ_i is too big, so we replace $\epsilon_i \leftarrow \epsilon_i - \gamma$. If PGD failed, then we set $\epsilon_i \leftarrow \epsilon_i + \gamma$.

Since the network is randomly initialized at the start of training, random predictions are made, and this causes $\{\epsilon_i\}$ to shrink rapidly. For this reason, we begin with a warmup period of a few (usually 10 epochs for CIFAR-10/100) epochs where adversarial training is performed using uniform ϵ for every sample. After the warmup period ends, we perform instance adaptive adversarial training.

A detailed training algorithm is provided in Alg. 6.



(a) CIFAR-10 (b) CIFAR-100

Figure 8.3: **Improving the accuracy-robustness trade-off.** Our method (shown in red) which adapts a unique ϵ_{tr} value for each training instance, outperforms standard adversarial training with any fixed ϵ_{tr} value (shown in blue). We report natural accuracy and adversarial accuracy in response to a PGD-1000 adversarial attack with $\epsilon_{te} = 8$ for each method. Adaptive training (IAAT) breaks through the Pareto frontier achieved by standard adversarial training with a fixed ϵ_{tr}.

8.4 Experiments

We evaluate the robustness and generalization of our models using the following metrics: (1) test accuracy of unperturbed (natural) test samples, (2) adversarial accuracy of white-box PGD attacks, (3) adversarial accuracy of transfer attacks and (4) accuracy of test samples under common image corruptions [4]. Following the protocol introduced in [4], we do not train our models on any image corruptions.

8.4.1 CIFAR

On CIFAR-10 and CIFAR-100 datasets, we perform experiments using Resnet-18 and WideRenset-32-10 models following [148, 153]. All models are trained using

Table 8.1: **Improving Robustness-Accuracy Trade-off (CIFAR-10):** PGD attacks are generated with $\epsilon_{te} = 8$. PGD_{10} and PGD_{100} attacks are generated with 5 random restarts, while PGD_{1000} uses 2 random restarts. Our approach *significantly* improves natural accuracy with a minor drop in adversarial robustness compared to adversarial training. Clean performance shown for reference.

Method	**Natural** acc. (in %)	**Whitebox acc.** (in %) PGD_{10}	PGD_{100}	PGD_{1000}	**Transfer** (in %) acc. (PGD_{1000})	**Corruption** acc. (in %)
			Resnet-18			
Clean	94.21	0.02	0.00	0.00	3.03	72.71
Adv ($\epsilon_{tr} = 8$)	83.20	**43.79**	**42.30**	**42.36**	59.80	73.73
IAAT	**87.26**	43.08	41.16	41.16	**59.87**	**78.82**
			WideResnet 32-10			
Clean	95.50	0.05	0.00	0.00	5.02	78.35
Adv ($\epsilon_{tr} = 8$)	86.85	46.86	44.82	44.84	**62.77**	77.99
IAAT	**91.34**	**48.53**	**46.50**	**46.54**	58.20	**83.13**

PGD-10 attacks i.e., 10 steps of PGD iterations to craft adversarial attacks during training. In the whitebox setting, models are evaluated on: (1) PGD-10 attacks with 5 random restarts, (2) PGD-100 attacks with 5 random restarts, and (3) PGD-1000 attacks with 2 random restarts. For transfer attacks, an independent copy of the model is trained using the same training algorithm and hyper-parameter settings, and PGD-1000 adversarial attacks with 2 random restarts are crafted on the surrogate model. Additionally, we report accuracy over the 19 corruptions proposed in CIFAR10-C [4].

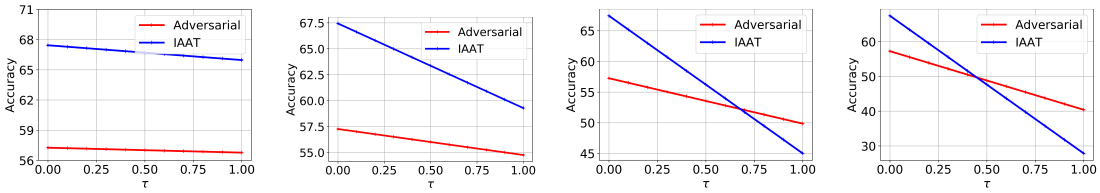
Beating the robustness-accuracy tradeoff: In adversarial training, the perturbation radius ϵ is a hyper-parameter. Training models with varying ϵ produces a robustness-accuracy tradeoff curve - models with small training ϵ achieve better natural accuracy and poor adversarial robustness, while models trained on large ϵ

Table 8.2: **Improving Robustness-Accuracy Trade-off (CIFAR-100):** PGD attacks are generated with $\epsilon = 8$. PGD_{10} and PGD_{100} attacks are generated with 5 random restarts, while PGD_{1000} uses 2 random restarts. Our approach *significantly* improves natural accuracy with a minor drop in adversarial robustness compared to fixed adversarial training. Clean performance shown for reference.

Method	**Natural** acc. (in %)	**Whitebox acc.** (in %) PGD_{10}	PGD_{100}	PGD_{1000}	**Transfer acc.** (in %) PGD_{1000}
		Resnet-18			
Clean	74.88	0.02	0.00	0.01	1.81
Adv($\epsilon_{tr} = 8$)	55.11	**20.69**	**19.68**	**19.91**	35.57
IAAT	**63.90**	18.50	17.10	17.11	**35.74**
		WideResnet 32-10			
Clean	79.91	0.01	0.00	0.00	1.20
Adv($\epsilon_{tr} = 8$)	59.58	**26.24**	**25.47**	**25.49**	**38.10**
IAAT	**68.80**	26.17	24.22	24.36	35.18

have improved robustness and poor natural accuracy. To generate this tradeoff, we perform adversarial training with ϵ in the range $\{1, 2, \ldots 8\}$. Instance adaptive adversarial training is then compared with respect to this tradeoff curve in Fig. 8.3(a), 8.3(b). Two versions of IAAT are reported – with and without a warmup phase. In both versions, we clearly achieve an improvement over the accuracy-robustness tradeoff. Use of the warmup phase helps retain robustness with a drop in natural accuracy compared to its no-warmup counterpart.

Clean accuracy improves for a fixed level of robustness: On CIFAR-10, as shown in Table. 8.1, we observe that our instance adaptive adversarial training algorithm achieves similar adversarial robustness as the adversarial training baseline. However, the accuracy on clean test samples increases by 4.06% for Resnet-18 and 4.49% for WideResnet-32-10. We also observe that the adaptive training algorithm improves robustness to unseen image corruptions. This points to an improvement

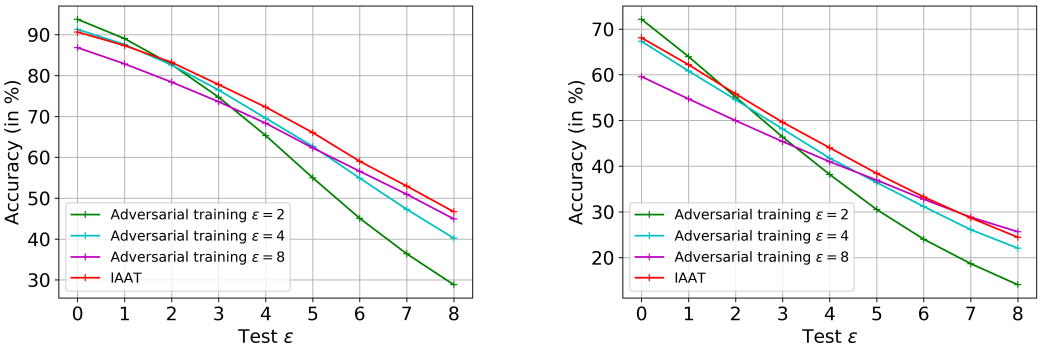


(a) CIFAR-10 (b) CIFAR-100 (c) CIFAR-100 (d) CIFAR-100

Figure 8.4: **Evaluating robustness-accuracy tradeoff in Imagenet**. Models are evaluated on test set containing a mixture of natural and adversarial samples, with τ fraction of adversarial samples. We plot τ Adversarial accuracy + $(1-\tau)$ Natural accuracy as τ is varied. Architecture used is Renset-152. Our approach achieves better tradeoff curve for most test settings.

in overall generalization ability. On CIFAR-100 (Table. 8.2), the performance gain in natural test accuracy further increases - 8.79% for Resnet-18, and 9.22% for Wideresnet-32-10. The adversarial robustness drop is marginal.

Maintaining performance over a range of test ϵ_{te}: Next, we analyze the adversarial robustness over a sweep of test-time ϵ values for PGD-1000. Fig. 8.5(a), 8.5(b) shows an adversarial training baseline where $\epsilon_{tr} = 8$ performs well at high ϵ_{te} regimes and poorly at low ϵ_{te} regimes. On the other hand, adversarial training with $\epsilon_{tr} = 2$ has a reverse effect, performing well at low ϵ_{te} and poorly at high ϵ_{te} regimes. Our instance adaptive training algorithm maintains good performance over all ϵ_{te} regimes, achieving slightly lower performance than the $\epsilon_{tr} = 2$ model for small test ϵ_{te} and dominating all models for larger test ϵ_{te}. Thus, we reiterate that the benefit of instance adaptative adversarial training is to improve the robustness-accuracy trade-off.



(a) CIFAR-10 (b) CIFAR-100

Figure 8.5: **Varying attack strengths.** Plot of adversarial robustness over a sweep of test ϵ. IAAT maintains good performance over the entire range of test ϵ.

Interpretability of ϵ: We find that the values of ϵ_i chosen by our adaptive algorithm correlate well with our own human concept of class ambiguity. Figure 8.2 shows that a sampling of images that receive small ϵ_i contains many ambiguous images, and these images are perturbed into a (visually) different class using $\epsilon = 16$. In contrast, images that receive a large ϵ_i have a visually definite class, and are not substantially altered by an $\epsilon = 16$ perturbation.

Robustness to other attacks: For all experiments, our instance adaptive algorithm is trained using only PGD attacks. Table. 8.3 evaluates IAAT and adversarial training (fixed $\epsilon_{tr} = 8$) on additional adversarial attacks at test time. IAAT achieves a minor robustness improvement, on average, against other gradient-based attacks, while significantly improving the natural accuracy.

Table 8.3: **Robustness Across Different Adversarial Attacks (CIFAR-10, WideResnet 32-10).** We report the robustness across 4 different adversarial attacks for both our instance adaptive approach (IAAT) and standard adversarial training using PGD-10 with fixed $\epsilon_{tr} = 8$. IAAT outperforms standard adversarial training both on clean data and new test-time attacks on which models were not trained on. Results in accuracy (%).

	Training Algorithm	
Test Time Attack	Adversarial training	IAAT
Natural acc.	86.85	**91.34**
PGD-1000 [148]	44.84	**46.54**
DeepFool [161]	65.28	**66.58**
MIFGSM [167]	**54.66**	53.99
CW40 [160]	55.62	**56.80**

Table 8.4: **Improving Robustness-Accuracy Trade-off (ImageNet):** We report robustness against PGD-1000 attacks whitebox attacks for varying test perturbation strengths (ϵ_{te}) - results are in accuracy (%). Additionally, we report robustness to common image corruptions (ImageNet-C) using the proposed mCE metric [4]. (↑) indicates higher numbers are better, while (↓) indicates lower numbers are better. Our approach, IAAT, improves natural accuracy, adversarial robustness on lower perturbation regimens, and robustness to corruptions.

Method	**Natural** acc. (in %) (↑)	**Whitebox acc.** (in %) (↑) $\epsilon_{te} = 4$	$\epsilon_{te} = 8$	$\epsilon_{te} = 12$	$\epsilon_{te} = 16$	**Corruption** mCE (↓)
Resnet-50						
Clean	75.80	0.64	0.18	0.00	0.00	76.69
Adv ($\epsilon_{tr} = 16$)	50.99	50.89	49.11	**44.71**	**35.82**	95.48
IAAT	**62.71**	**61.52**	**54.63**	39.90	22.72	**85.21**
Resnet-101						
Clean	77.10	0.83	0.12	0.00	0.00	70.37
Adv ($\epsilon_{tr} = 16$)	55.42	55.11	53.07	**48.35**	**39.08**	91.45
IAAT	**65.29**	**63.83**	**56.62**	41.51	23.91	**79.52**
Resnet-152						
Clean	77.60	0.57	0.08	0.00	0.00	69.27
Adv ($\epsilon_{tr} = 16$)	57.26	56.77	54.75	**49.86**	**40.40**	89.31
IAAT	**67.44**	**65.97**	**59.28**	45.01	27.85	**78.53**

Table 8.5: **Ablation: Effect of warm-up on CIFAR-10** We find that training IAAT with warmup is important as it increases the adversarial robustness significantly with minor drops in clean performance.

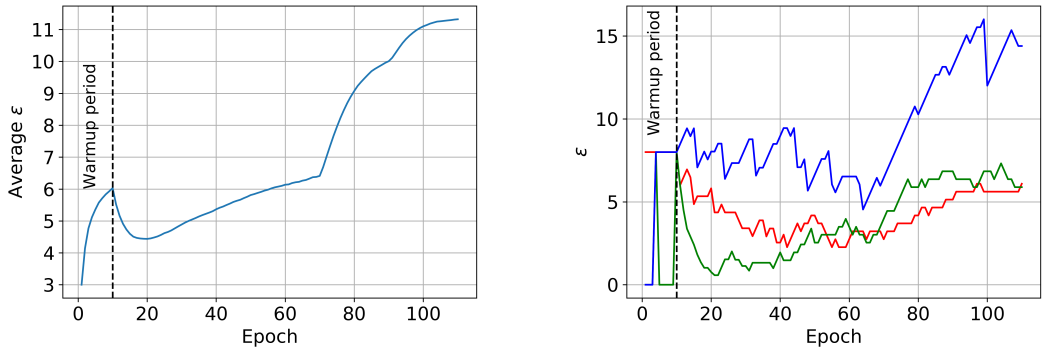
Method	**Natural** acc. (%)	**Whitebox acc.** (in %) PGD_{10}	PGD_{100}	PGD_{1000}	**Transfer acc.**(%) PGD_{1000}	**Corruption** acc. (in %)
			Resnet-18			
IAAT (no warm)	**89.62**	40.55	38.15	38.08	58.89	**81.10**
IAAT (warm)	87.26	**43.08**	**41.16**	**41.16**	**59.87**	78.82
			WideResnet 32-10			
IAAT (no warm)	**92.62**	45.12	41.08	41.11	53.08	**84.92**
IAAT (warm)	90.67	**48.53**	**46.50**	**46.54**	**58.20**	83.13

Table 8.6: **Ablation: Effect of warmup (CIFAR-100)**. Warm-up is important to maintain adversarial robustness of IAAT.

Method	**Natural** acc. (in %)	**Whitebox acc.** (in %) PGD_{10}	PGD_{100}	PGD_{1000}	**Transfer acc.**(%) PGD_{1000}
		Resnet-18			
Adaptive (no warm)	**68.34**	14.76	13.29	13.30	32.39
Adaptive (warm)	63.90	**18.50**	**17.10**	**17.11**	**35.74**
		WideResnet 32-10			
Adaptive (no warm)	**75.48**	18.14	13.78	13.71	24.00
Adaptive (warm)	68.80	**26.17**	**24.22**	**24.36**	**35.18**

Table 8.7: **Ablation: IAAT vs exact line search.** IAAT achieves comparable accuracies (%) to exhaustive line search for selecting ϵ (CIFAR-10, Resnet-18).

Algorithm	Natural acc.	PGD-10	PGD-1000
Full line search	88.67	43.26	41.37
IAAT	87.26	43.08	41.16



(a) CIFAR-10 (b) CIFAR-100

Figure 8.6: **Instance specific ϵ chosen vs training epoch.** ϵ values increase during training as the model becomes more robust.

8.4.2 Imagenet

Following the protocol introduced in [152], we attack Imagenet models using random targeted attacks instead of untargeted attacks as done in previous experiments. During training, adversarial attacks are generated using 30 steps of PGD. As a baseline, we use adversarial training with a fixed ϵ of 16/255. This is the setting used in [152]. Adversarial training on Imagenet is computationally intensive. To make training practical, we use distributed training with synchronized SGD on 64/128 GPUs. More implementation details can be found in Balaji et al. [168].

At test time, we evaluate models on clean test samples and on whitebox attacks (PGD-1000) with $\epsilon = \{4, 8, 12, 16\}$. Additionally, we also report normalized mean corruption error (mCE), an evaluation metric introduced in [4] to test the robustness of neural networks to image corruptions. This metric reports mean classification error of different image corruptions averaged over varying levels of degradation.

Note that while accuracies are reported for natural and adversarial robustness, mCE denotes classification errors, so lower numbers are better.

To measure the tradeoff between robustness and accuracy, we evaluate performance on test set containing τ fraction of adversarial samples for a given test ϵ_{te}. The fraction τ is varied from 0 to 1. When $\tau = 0$, the test set contains only the natural samples and when $\tau = 1$ it contains only the adversarial samples. Evaluating over a sweep of τ thus gives a complete characterization of robustness-accuracy tradeoff. A plot showing this performance curve for Resnet-152 architecture is shown in Figure. 8.4. For $\epsilon_{te} = \{4, 8\}$, IAAT consistently outperforms adversarial training for all values of τ. For $\epsilon_{te} = 12$ and $\epsilon_{te} = 16$, IAAT outperforms adversarial training for $\sim 75\%$ and $\sim 50\%$ of the curve. Thus, IAAT achieves better tradeoff than adversarial training. In $\epsilon_{te} = 16$, IAAT achieves comparable tradeoff to adversarial training. Note that $\epsilon_{te} = 16$ is generally considered a very large perturbation radius (CIFAR experiments are all evaluated with the standard value $\epsilon_{te} = 8$).

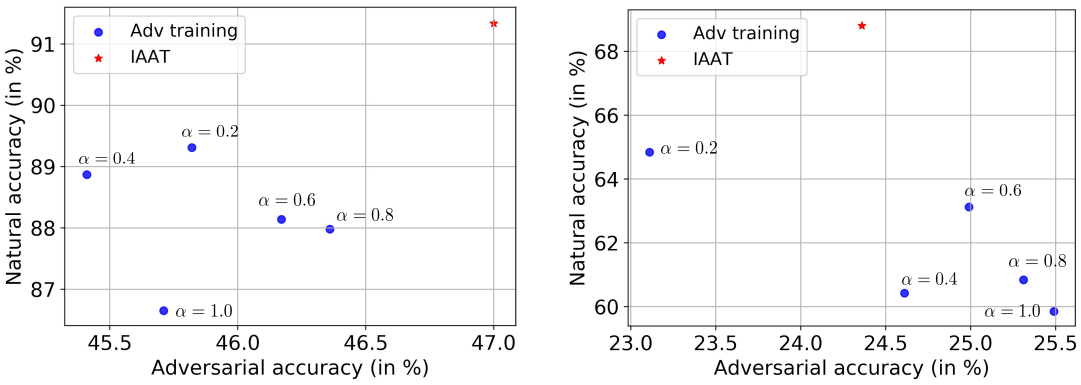
A complete set of results showing natural and adversarial accuracies for various architectures and test ϵ is shown in Table. 8.4. We observe a huge drop in natural accuracy for adversarial training (25%, 22% and 20% drop for Resnet-50, 101 and 152 respectively). Adaptive adversarial training significantly improves the natural accuracy – we obtain a consistent performance gain of $10 + \%$ on all three models over the adversarial training baseline. On whitebox attacks, IAAT outperforms the adversarial training baseline on low ϵ regimes, however a drop of 13% is observed at high ϵ's ($\epsilon = 16$). On the corruption dataset, our model consistently outperforms adversarial training.

8.5 Ablation experiments

Effect of warmup. Recall from Section 8.3 that during warmup, adversarial training is performed with uniform norm-bound constraints. Once the warmup phase ends, we switch to instance adaptive training. From Table 8.5 and 8.6, we observe that when warmup is used, adversarial robustness improves with a small drop in natural accuracy, with more improvements observed in CIFAR-100. However, both these settings improve the accuracy-robustness tradeoff (Fig. 8.3(a), 8.3(b)).

Instance Specific ϵ vs Epoch. Next, we visualize the evolution of ϵ over epochs in adaptive adversarial training (CIFAR-10). Fig 8.6(a) shows how the average ϵ chosen changes during learning while Fig 8.6(b) show how the ϵ for three randomly sampled images changes during learning. We observe that average ϵ converges to around 11, which is higher than the default CIFAR-10 adversarial training setting of $\epsilon_{tr} = 8$. In addition, some samples converge to high ϵ values while others remain low, showcasing the adaptability of our approach.

Natural sample exposure. Instance adaptive adversarial training uses natural samples during training as opposed to just using adversarial samples as done in adversarial training. In this experiment, we intend to study if the improvements in natural accuracy is an outcome of natural sample exposure during IAAT training. To do this, we train models using a variation of adversarial training, called *mixed adversarial training*, where the loss function used is a linear combination of natural loss (cross entropy on unperturbed samples) and adversarial loss. That is, models are trained using α Natural loss + $(1-\alpha)$ Adversarial loss, for a given constant α. A plot



(a) CIFAR-10 (b) CIFAR-100

Figure 8.7: **Effect of incorporating clean data during training.** Comparison of IAAT with mixed adversarial training trained using α Adversarial loss + $(1-\alpha)$ Clean loss. IAAT clearly improves the robustness-accuracy tradeoff compared to mixed adversarial trianing for all values of τ.

comparing robustness-accuracy tradeoff for models trained using mixed adversarial training and IAAT is shown in Figure 8.7(a), 8.7(b). IAAT achives significant improvement in natural accuracy and the robustness-accuracy tradeoff over mixed adversarial training. Hence, just exposing natural samples during training does not help adversarial training generalize better. This shows that improvements achieved by IAAT is due to the instance adaptive algorithm.

Other heuristics. We are interested in estimating instance-specific perturbation radius ϵ_i such that predictions are consistent within the chosen ϵ_i-ball. To obtain an exact estimate of such an ϵ_i, we can perform a line search as follows: Given a discretization η and a maximum perturbation radius ϵ_{max}, generate PGD attacks with radii $\{i\eta\}_{i=1}^{\epsilon_{max}/\eta}$. Choose the desired ϵ_i as the maximum $i\eta$ for which the prediction remains consistent as that of the ground-truth label. We compare the performance of

Table 8.8: Comparison with Mixup.

Method	Natural acc. (in %)	Whitebox acc. (in %) PGD_{10}	PGD_{100}	PGD_{1000}	Transfer attack (in %) PGD_{1000}
		Resnet-18			
Mixup	89.47	42.60	38.42	38.49	59.48
IAAT	87.26	43.08	41.16	41.16	59.87
		WideResnet 32-10			
Mixup	92.57	45.01	36.6	36.44	63.57
IAAT	90.67	48.53	46.50	46.54	58.20

exact line search with that of IAAT in Table 8.7. We observe that exact line search marginally improves compared to IAAT. However, exact line search is computationally expensive as it requires performing ϵ_{max}/η additional PGD computations, whereas IAAT requires only 2.

Comparison with Mixup A recent paper that addresses the problem of improving natural accuracy in adversarial training is [169], where adversarially trained models are optimized using mixup loss instead of the standard cross-entropy loss. In this paper, natural accuracy was shown to improve with no drop in adversarial robustness. However, the robustness experiments were not evaluated on strong attacks (experiments were reported only on PGD-20). We compare our implementation of mixup adversarial training with IAAT on stronger attacks in Table. 8.8. We observe that while natural accuracy improves for mixup, drop in adversarial accuracy is much higher than IAAT.

8.6 Conclusion

In this chapter, we present *instance adaptive adversarial training* (IAAT), a method to improve the robustness-accuracy tradeoff for adversarial training. We show that realizable robustness is a sample-specific attribute: samples close to the decision boundary can only achieve robustness within a small ϵ ball, as they contain samples from a different class beyond this radius. On the other hand samples far from the decision boundary can be robust on a relatively large perturbation radius. Motivated by this observation we estimate sample-specific perturbation radii within which to enforce label consistency. Our proposed algorithm has empirically been shown to improve the robustness-accuracy tradeoff in CIFAR-10, CIFAR-100 and Imagenet datasets.

Part V

Conclusions and Future Research Directions

Chapter 9: Conclusions and Future Research Directions

9.1 Summary

In this dissertation, we presented several techniques for handling out-of-distribution shifts in deep learning systems. We presented three broad classes of approaches involving *detection*, *adaptation* and *generalization* to out-of-distribution shifts. In the first part, we looked at probabilistic modeling of the data using deep generative models. We showed how GANs can be used for likelihood estimation, which was then used for *detecting* out-of-distribution samples as outliers. In the second part, we looked at *adaptation* algorithms, where the goal was to adapt the neural networks to an unlabeled target domain so that the performance on the target distribution improves. We presented three algorithms - Generate to Adapt, Robust Optimal Transport and Normalized Wasserstein measure to perform domain adaptation under various settings, and studied their properties.

In the last two parts of the dissertation, we focused on robust training algorithms for *generalization* to out-of-distribution shifts. In the third part, we proposed Metareg, an algorithm for learning data-dependent regularization functions using meta-learning. The regularization function was used for training models that give better out-of-distribution performance on novel test distributions. Finally, in the

last part of the dissertation, we proposed Instance Adaptative Adversarial Training, a robust training algorithm for improving robustness-accuracy tradeoff with respect to adversarial shifts.

9.2 Future Directions

Online Adaptation: The adaptation algorithms discussed in this dissertation work on a static setting, one in which both the source and the target domains are stationary. In real world, however, data distributions change continually with time. Using the static adaptation algorithms on such dynamic environments can be undesirable since static algorithms have high sample complexity and can lead to catastrophic forgetting. Hence, it is desirable to develop scalable adaptation algorithms that can seamlessly adapt to evolving distributions.

Improved Models for OOD Generalization: Models trained on Imagenet, while being extremely accurate on the in-distribution validation set, perform poorly on out-of-distribution datasets such as corruptions [170], paintings, Imagenet renditions [171], etc. Hence, any downstream visual recognition system that uses these model weights for fine-tuning will have poor OOD generalization as well. Data augmentation seems to be the most effective technique for improving OOD generalization, but there still exist a huge performance gap between the in-distribution and out-of-distribution accuracies [171]. Hence, it is of utmost interest to develop better training algorithms (such as Arjovsky et al. [172]) that can yield robust and generalizable models to these unseen out-of-distribution shifts.

Large Scale Likelihood Estimation: The likelihood estimation framework we developed in this dissertation fails to work effectively on large-scale datasets. Other likelihood-based models such as normalizing flows [173] that have been developed recently can work on large-scale datasets. But, they have several failure modes such as assigning high likelihood scores to out-of-distribution samples [174]. Hence, it is of interest to develop likelihood-based generative models that are both scalable, have good sample generation quality and provide good estimates of sample likelihood scores for tasks such as out-of-distribution detection.

Bibliography

[1] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S. Yu. Transfer feature learning with joint distribution adaptation. In *IEEE International Conference on Computer Vision, ICCV 2013*, 2013.

[2] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[3] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. *CoRR*, abs/1710.03463, 2017.

[4] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJz6tiCqYm`.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[8] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *arXiv*, 2016. URL `https://arxiv.org/abs/1609.03499`.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[11] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[13] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[14] Sébastien Racanière, Théophane Weber, David P Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5694–5705, 2017.

[15] Adam Stooke and Pieter Abbeel. Accelerated methods for deep reinforcement learning. *arXiv preprint arXiv:1803.02811*, 2018.

[16] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[17] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[18] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[20] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, 126(9):973–992, Sep 2018.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[22] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[23] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[24] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[25] D. Hubel and T. Wiesel. Receptive fields of single neurones in the cat's striate cortex. In *The Journal of Physiology*, pages 574–591, 1959.

[26] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets*, pages 267–285. Springer, 1982.

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

[28] Marvin L Minsky and Seymour A Papert. Perceptrons: Expanded edition, 1988.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[30] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[33] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[34] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[35] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[36] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.

[37] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[38] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv preprint arXiv:1705.02894*, 2017.

[39] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. *arXiv preprint arXiv:1606.00709*, 2016.

[40] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

[41] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *arXiv preprint arXiv:1606.03498*, 2016.

[43] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[44] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[45] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[46] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[47] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[48] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[49] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[50] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. URL `http://arxiv.org/abs/1710.10196`.

[51] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013.

[52] Vivien Seguy, Bharath Bhushan Damodaran, Rémi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. Large-scale optimal transport and mapping estimation. *arXiv preprint arXiv:1711.02283*, 2017.

[53] Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning generative models with sinkhorn divergences. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1608–1617, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

[54] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P. Xing. On unifying deep generative models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rylSzl-R-`.

[55] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*. PMLR, August 2017.

[56] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger B. Grosse. On the quantitative analysis of decoder-based generative models. *CoRR*, abs/1611.04273, 2016.

[57] Henning Petzka, Asja Fischer, and Denis Lukovnicov. On the regularization of Wasserstein GANs. *arXiv preprint arXiv:1709.08894*, 2017.

[58] Xin Guo, Johnny Hong, Tianyi Lin, and Nan Yang. Relaxed Wasserstein with applications to GANs. *arXiv preprint arXiv:1705.07164*, 2017.

[59] Soheil Feizi, Changho Suh, Fei Xia, and David Tse. Understanding GANs: the LQG setting. *arXiv preprint arXiv:1710.10793*, 2017.

[60] Maziar Sanjabi, Jimmy Ba, Meisam Razaviyayn, and Jason D Lee. Solving approximate Wasserstein GANs to stationarity. *Neural Information Processing Systems (NIPS)*, 2018.

[61] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Sinkhorn-AutoDiff: Tractable Wasserstein learning of generative models. *arXiv preprint arXiv:1706.00292*, 2017.

[62] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[63] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

[64] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[65] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722*, 2017.

[66] Yogesh Balaji, Hamed Hassani, Rama Chellappa, and Soheil Feizi. Entropic GANs meet vaes: A statistical approach to compute sample likelihoods in GANs. In *International Conference on Machine Learning*, pages 414–423. PMLR, 2019.

[67] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[68] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[69] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[70] Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings, 2010.

[71] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189. PMLR, 2015.

[72] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[73] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*. Springer, 2016.

[74] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Scholkopf. Correcting sample selection bias by unlabeled data. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, 2006.

[75] Hal Daume III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, June 2007.

[76] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, 2011.

[77] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks*, 2011.

[78] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014. URL `http://arxiv.org/abs/1412.3474`.

[79] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 97–105, 2015.

[80] Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Unsupervised domain adaptation with residual transfer networks. *CoRR*, abs/1602.04433, 2016.

[81] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2208–2217, 2017.

[82] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *CoRR*, abs/1611.02200, 2016. URL `http://arxiv.org/abs/1611.02200`.

[83] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.

[84] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 469–477. 2016.

[85] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

[86] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. *arXiv preprint arXiv:1610.09585*, 2016.

[87] Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

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

[89] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[90] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, page 5, 2011.

[91] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[92] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2014*, 2014.

[93] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *ICCV*, 2015.

[94] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[95] Xingchao Peng and Kate Saenko. Synthetic to real adaptation with deep generative correlation alignment networks. *CoRR*, abs/1701.05524, 2017. URL `http://arxiv.org/abs/1701.05524`.

[96] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, 2014.

[97] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017.

[98] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *AAAI*, pages 4058–4065. AAAI Press, 2018.

[99] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Unbalanced optimal transport: Dynamic and kantorovich formulation. *arXiv preprint arXiv:1508.05216*, 2015.

[100] Matthias Liero, Alexander Mielke, and Giuseppe Savaré. Optimal entropy-transport problems and a new Hellinger–Kantorovich distance between positive measures. *Inventiones Mathematicae*, 211(3):969–1117, Mar 2018. ISSN 1432-1297.

[101] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Robust optimal transport with applications in generative modeling and domain adaptation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12934–12944. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/9719a00ed0c5709d80dfef33795dcef3-Paper.pdf`.

[102] Justin Solomon, Raif M. Rustamov, Leonidas J. Guibas, and Adrian Butscher. Wasserstein propagation for semi-supervised learning. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 306–314. JMLR.org, 2014.

[103] Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas J. Guibas. Convolutional Wasserstein distances: efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4):66:1–66:11, 2015.

[104] Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: Histogram regression using optimal transport. *ACM Transactions on Graphics (SIGGRAPH 2016)*, 35(4), 2016.

[105] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 2015. doi: 10.1137/141000439.

[106] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced transport problems. *arXiv preprint arXiv:1607.05816*, 2016.

[107] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a Wasserstein loss. In *Advances in Neural Information Processing Systems*, pages 2053–2061, 2015.

[108] Stanislav Kondratyev, Léonard Monsaingeon, Dmitry Vorotnikov, et al. A new optimal transport distance on the space of finite Radon measures. *Advances in Differential Equations*, 21(11/12):1117–1164, 2016.

[109] Karren D. Yang and Caroline Uhler. Scalable unbalanced optimal transport using generative adversarial networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyexAiA5Fm`.

[110] Hongseok Namkoong and John C Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2208–2216. Curran Associates, Inc., 2016.

[111] Pedro Oliveira Pinheiro. Unsupervised domain adaptation with similarity learning. *CoRR*, abs/1711.08995, 2017.

[112] Mingsheng Long, ZHANGJIE CAO, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1640–1650. Curran Associates, Inc., 2018.

[113] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *arXiv preprint arXiv:1712.02560*, 2017.

[114] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *CoRR*, abs/1710.06924, 2017.

[115] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *International Conference on Computer Vision (ICCV)*, 2017.

[116] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. *arXiv preprint arXiv:1812.01754*, 2018.

[117] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Normalized Wasserstein for mixture distributions with applications in adversarial learning and domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[118] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 945–954, 2017.

[119] Yaoliang Yu and Csaba Szepesvári. Analysis of kernel mean matching under covariate shift. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[120] Qingchao Chen, Yang Liu, Zhaowen Wang, Ian Wassell, and Kevin Chetty. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[121] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. MGAN: Training generative adversarial nets with multiple generators. 2018.

[122] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. *CoRR, abs/1704.02906*, 6:7, 2017.

[123] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[124] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. URL `https://arxiv.org/abs/1611.03530`.

[125] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957, 1992.

[126] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1), January 2014.

[127] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1058–1066, 2013.

[128] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456, 2015.

[129] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1126–1135, 2017.

[130] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *In International Conference on Learning Representations (ICLR)*, 2017.

[131] Ke Li and Jitendra Malik. Learning to optimize neural nets. *CoRR*, abs/1703.00441, 2017.

[132] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3981–3989. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6461-learning-to-learn-by-gradient-descent-by-gradient-descent.pdf`.

[133] Sebastian Thrun and Lorien Pratt, editors. *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998. ISBN 0-7923-8047-9.

[134] Jürgen Schmidhuber. On learning how to learn learning strategies. Technical report, 1995.

[135] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Proceedings of Machine Learning Research*, 2017.

[136] Damien Teney and Anton van den Hengel. Visual question answering as a meta learning task. *CoRR*, abs/1711.08105, 2017.

[137] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *Proceedings of the 30th International Conference on Machine Learning, W&CP 28(1)*, pages 10–18. JMLR, 2013. Volume 28, number 1.

[138] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, 2015.

[139] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A. Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part I*, ECCV'12, 2012.

[140] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5543–5551, 2017.

[141] Fengwei Zhou, Bin Wu, and Zhenguo Li. Deep meta-learning: Learning to learn in the concept space. *CoRR*, abs/1802.03596, 2018.

[142] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, 2006.

[143] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in Neural Information Processing Systems*, 31:998–1008, 2018.

[144] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 343–351. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6254-domain-separation-networks.pdf`.

[145] Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*. icml.cc / Omnipress, 2012.

[146] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL `http://arxiv.org/abs/1312.6199`.

[147] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

[148] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

[149] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.

[150] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[151] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[152] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[153] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/zhang19p.html`.

[154] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[155] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, pages 1178–1187, 2018.

[156] Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018.

[157] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from

concentration of measure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4536–4543, 2019.

[158] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Max-margin adversarial (mma) training: Direct input space margin maximization through adversarial training. *arXiv preprint arXiv:1812.02637*, 2018.

[159] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6572`.

[160] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57, 2017.

[161] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2574–2582, 2016.

[162] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HyydRMZC-`.

[163] Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected Sinkhorn iterations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 6808–6817. PMLR, 2019.

[164] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S18Su--CW`.

[165] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkJ3ibb0-`.

[166] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=H1uR4GZRZ`.

[167] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[168] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint arXiv:1910.08051*, 2019.

[169] Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing accuracy. *CoRR*, abs/1906.06784, 2019.

[170] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[171] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020.

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

[173] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[174] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.