Novel Semi-supervised Learning Approaches without Enforcing Consistency

Jingwei MAO

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer and Information Engineering

The Chinese University of Hong Kong, Shenzhen
August 10, 2019

Thesis Assessment Committee

Professor Tsung-Hui Chang
Professor Zhi-Quan Luo
Professor Feng Yin
Professor Simon Pun
Professor Wing-Kin Ma
Professor Tong Zhang

Abstract of thesis entitled:

Novel Semi-supervised Learning Approaches without Enforcing Consistency

Submitted by Jingwei MAO

for the degree of Master of Philosophy

at The Chinese University of Hong Kong, Shenzhen

in August 10, 2019

Although supervised learning has achieved remarkable success in many artificial intelligence tasks, such as image classification, machine translation and auto speech recognition, it can only make use of labeled data and usually requires large amount of them. However, in practice, we usually have access to limited labeled data because collecting labeled data is costly. On the contrary, it's much easier to obtain large amount of unlabeled data in most tasks. Therefore, semi-supervised learning is gaining more and more popularity because it aims to exploit both labeled and unlabeled training data.

In present, most semi-supervised learning approaches take advantage of unlabeled data via consistency regularization. That is, they calculate two different outputs for each input and the distance between the two outputs is used as the regularization term in loss function. These approaches vary in the way to define the two different outputs, which are supposed to be close according to their common underlying assumption: the true mapping from input to output is smooth.

However, this assumption may not hold especially in the cases that the input data is of high-dimensional. In view of this, we propose a novel semi-supervised learning approach that does not rely on this assumption. In particular, we propose to align empirical distributions of labeled data and unlabeled data in feature space by minimizing the Wasserstein distance between them. The motivation is that the empirical distribution of unlabeled data is close to the data-generating distribution because of its large amount. Therefore, aligning the empirical distributions of labeled data and unlabeled data in feature space can make the empirical distribution of labeled data and the data-generating distribution be close in feature space, which can improve the model's generalization.

We evaluated our method on CIFAR-10 and SVHN datasets and the results showed the superior performance of our proposed method. Especially in the case of using 1,000 labeled samples on CIFAR-10 dataset, our method reduced the state-of-the-art error rate from $15.48\% \pm 0.78\%$ to $9.92\% \pm 0.58\%$.

摘要

中文摘要 尽管有监督学习已经在很多人工智能任务上取得了显著的成功，比如图片分类，机器翻译和语音识别等，它只能利用有标签数据并且通常需要大量的有标签数据。然而，在实际工程中，我们通常只有很有限的有标签数据因为收集有标签数据花费很高。与之相反的是，在大多数任务中，获取无标签数据要容易许多。因此，半监督学习正变得越来越受关注因为它设法同时利用有标签和无标签数据。

目前，大多数的半监督学习方法通过一致性正则项来利用无标签数据。意思是说，对于同一个输入，它们会计算两个不同的输出，然后这两个输出之间的距离会被用作损失函数中的正则项。这些方法的区别在于对两个不同输出的定义，根据他们共同的假设：从输入到输出的真实映射是平滑的，这两个输出应该靠近。

然而，这个假设可能不成立尤其是在输入数据维度很高的情况下。因此，我们提出一种不依赖于这个假设的新颖的半监督学习方法。具体来说，我们通过最小化有标签数据和无标签数据在特征空间的经验分布之间的Wasserstein距离来对齐这两个分布。这样做的动机是因为无标签数据的数量很大，所以它的经验分布与产生数据的真实分布接近。因此，对齐有标签数据和无标签数据在特征空间的经验分布可以使得有标签数据在特征空间的经验分布和数据在特征空间的真实分布接近，提高模型的泛化能力。

我们在CIFAR-10和SVHN两个数据集上检验了我们的方法，实验结果显示了该方法的优越性。特别是在CIFAR-10数据集上使用1,000个有标签数据时，该方法把最好的分类错误率从$15.48\% \pm 0.78\%$进一步降到了$9.92\% \pm 0.58\%$.

Acknowledgement

The past three years at CUHK(SZ) have been an unforgettable and invaluable experience to me. I still remember that in the first semester, I was nearly overwhelmed because of courses, teaching assistant duties and research. Lectures are hard to follow because they are given in English and are mathematically intensive. I had to spend much time on understanding contents delivered by professors and finishing homework after class. As a teaching assistant, I gave tutorials twice a week, which is also time-consuming and challengeable. I would not have been able to overcome these difficulties and challenges without the help and support of many people especially my families and my advisor.

First and foremost, I would like to give my greatest thanks to my advisor Prof. Zhi-Quan Luo for his continued support and guidance. When I first started my study, I knew little about how to do research. He suggested that I should actively attend various seminars, which broadened my horizon and from which I learned about different research fields. He also encouraged me to ask questions whenever I could not follow speakers. In the beginning, I didn't really realize the importance of asking questions and didn't know how to ask questions. When attending seminars or group meetings, I could always see Prof. Luo asking many good questions. Motivated by that, I started to ask questions too. The more questions I asked, the better I understood the content delivered by speakers. More importantly, I gradually formed the habit of raising questions not only to spekers but also to myself, which promotes me to think deeper and get closer to the essence of problem.

I would like to also thank Prof. Tsung-Hui Chang, Prof. Feng Yin, Prof. Simon Pun, Prof. Wing-Kin Ma and Prof. Tong Zhang for being my thesis assessment committee. I would like to thank Prof. Wen-Ye Li who gave me much guidance when we work together on the paper accepted by the NeurIPS 2018. I would like to thank Prof. Qing-Jiang Shi, who guided me when I worked in the project managed by him. I would like to thank Prof. Zhen Li, who gave me guidance when we participated in the competition of AI challenger 2018. I would like to thank Prof. Ruo-Yu Sun, who taught me a lot in the group study of GAN. I would like to thank Dr. Zhi-Guo Wang, with whom I had many

inspiring discussions.

Lastly, I would like to thank all of the fellow students in Prof. Luo's group, with whom I had many memorable moments.

Contents

List of Figures

Chapter 1

Introduction

Supervised learning has achieved remarkable success in many artificial intelligence tasks, such as image classification [1], machine translation [2] and auto speech recognition [3]. Despite of its success, supervised learning requires large amount of labeled data for training. However, in practice, we usually have access to limited labeled data since collecting them is costly. On the contrary, it's much easier to obtain large amount of unlabeled data in most tasks. Naturally, we may ask whether unlabeled data can be used so that the model's performance will be better than supervised learning. That's exactly what semi-supervised learning aims for.

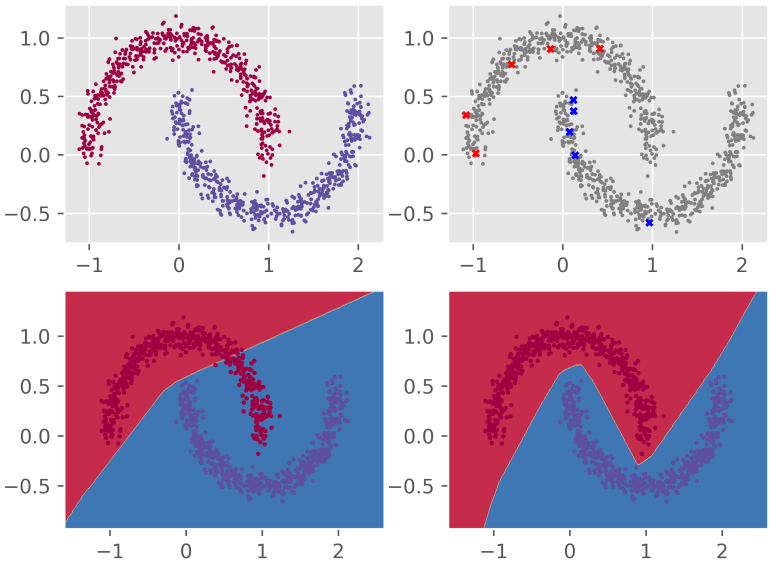


Figure 1.1: Illustration of effectiveness of semi-supervised learning

As shown in Figure 1.1, we illustrate the effectiveness of semi-supervised learning on the two-moons dataset. From the top left of Figure 1.1, we can see that the two-moons dataset consists of two classes and samples of each class form a moon-like shape. Suppose we are given 10 labeled samples and large number of unlabeled samples, which are represented by crosses and gray dots respectively on the top right of Figure 1.1. We then construct a three-layer feedforward neural network as classifier and train it using supervised learning and semi-supervised learning, respectively. The decision boundary of classifier trained by supervised learning is shown on the bottom left of Figure 1.1, from which we can see that it overfit to the 10 labeled samples and thus generalized poorly. In contrast, the classifier trained by semi-supervised learning have much better generalization, as can be seen from its decision boundary on the bottom right of Figure 1.1.

1.1 Related Work

1.1.1 Self-training

To the best of our knowledge, the history of semi-supervised learning dates back to 1960s. At that time, the idea of using unlabeled data is self-training [10–12], which is a wrapper-algorithm that uses a supervised learning method repeatedly. In specific, it starts by training an initial model on labeled set. In every iteration, a part of unlabeled data is labeled based on current model's predictions; and the newly labeled data is then used for training a new model together with labeled set.

This idea is kind of naive and the disadvantage is obvious. One can imagine that the initial model is very likely to make wrong predictions on unlabeled data and these wrongly labeled data will mislead the training of the new model. This bad effect will accumulate and may end up resulting in a model that is even worse than the initial model.

1.1.2 Co-training

Co-training was first proposed in [17, 18]. It splits features into two sets and trains two separate models on the two sub-feature sets respectively. The two models are initially trained on the labeled set. Then, each model is used to classify unlabeled data and is retrained with the additional training examples given by the other model. This process repeats. Note that co-training requires the following assumptions: (a) features can be split into two sets; (b) each sub-feature set is sufficient to train a good model; (c) the two sets are conditionally independent given the class.

To compare co-training and generative models, [9] performed extensive empirical experiments. Their results show that co-training performs well if the conditional independence assumption indeed holds. If there is no natural feature split, the authors create artificial split by randomly break the feature set into two subsets and show that co-training with random feature split still helps but not as much as before.

1.1.3 Generative Models

Generative models assume that the data-generating distribution $p(x)$ is an identifiable mixture distribution. Identifiable means that given $p(x)$, the decomposition in $\sum_y p(y)p(x|y)$ is unique. For example, Gaussian mixture is identifiable. Under this assumption, the mixture components $p(x|y)$ can be identified with large amount of unlabeled data. Once the mixture components are identified, next step is to assign class labels to these components, which can be solved exponentially quickly in the number of labeled samples [4]. Finally, the Bayes rule is used to calculate the predictive density $p(y|x)$:

$$p(y|x) = \frac{p(x,y)}{\sum_y p(x,y)} \tag{1.1}$$

If the mixture model assumption is correct, unlabeled data is guaranteed to improve accuracy [4–6]. However, if the mixture model assumption is wrong, unlabeled data may actually hurt accuracy [7]. Therefore, it is important to carefully construct the mixture model to reflect reality. For example, in text classification a topic may contain several sub-topics and was thus modeled by multiple multinomial [8].

1.1.4 Graph-based Approaches

Graph-based semi-supervised learning algorithms rely on the geometry of the data induced by both labeled and unlabeled data. This geometry can be naturally represented by a graph where nodes represent the training data and edges represent similarities between them. These similarities are given by a predefined weight matrix $\mathbf{W}$ such as Gaussian kernel matrix and k-nearest neighbor matrix. Given the graph, a simple idea for semi-supervised learning is to propagate labels [19–21] on the graph. We show a vanilla algorithm of this kind in Algorithm 1. Estimated labels on both labeled and unlabeled data are denoted by $\hat{Y} = (\hat{Y}_l, \hat{Y}_u)$.

Except for label propagation, another idea for semi-supervised learning is minimizing a regularized objective function derived from the graph [22–26]. This objective function consists of two parts: consistency with the initial labeling and consistency with the ge-

Algorithm 1 Label Propagation [19]

Compute Similarity matrix $\mathbf{W}$
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j \mathbf{W}_{ij}$
Initialize $\hat{Y}^{(0)} \leftarrow (y_1, ..., y_l, 0, 0, ..., 0)$
Iterate
  $\hat{Y}^{(t+1)} \leftarrow \mathbf{D}^{-1}\mathbf{W}\hat{Y}^{(t)}$
  $\hat{Y}_l^{t+1} \leftarrow Y_l$
until convergence to $\hat{Y}^{(\infty)}$
Label point x_i by the sign of $\hat{y}_i^{(\infty)}$

ometry of the data. Given a labeling $\hat{Y} = (\hat{Y}_l, \hat{Y}_u)$, consistency with the initial labeling can be measured, e.g., by

$$\sum_{i=1}^{l}(\hat{y}_i - y_i)^2 = ||\hat{Y}_l - Y_l||^2 \tag{1.2}$$

while consistency with the geometry of the data can be measured, e.g., by

$$\begin{aligned} \frac{1}{2}\sum_{i,j=1}^{n}\mathbf{W}_{ij}(\hat{y}_i - \hat{y}_j)^2 &= \frac{1}{2}\left(2\sum_{i=1}^{n}\hat{y}_i^2\sum_{j=1}^{n}\mathbf{W}_{ij} - 2\sum_{i,j=1}^{n}\mathbf{W}_{ij}\hat{y}_i\hat{y}_j\right) &(1.3)\\ &= \hat{Y}^\top(\mathbf{D} - \mathbf{W})\hat{Y} &(1.4)\\ &= \hat{Y}^\top L\hat{Y} &(1.5)\end{aligned}$$

where $L = \mathbf{D} - \mathbf{W}$ is the graph Laplacian. This means we penalize rapid changes in $\hat{Y}$ between points that are close.

1.1.5 Semi-supervised Deep Learning

Since the great success of supervised deep learning [1–3], many semi-supervised deep learning approaches have been proposed [27, 29–33, 57]. Most of these approaches take advantage of unlabeled data by adding a regularization term such as entropy regularization, consistency regularization and interpolation consistency regularization. Entropy regularization [27] encourages the classifier to make confident predictions on unlabeled data by penalizing the entropy of $p(y|x)$. Consistency regularization calculates two outputs for each unlabeled sample and the distance between the two outputs is used to regularize the supervised loss. The difference of consistency regularization approaches lies in the way to define the two outputs, which are supposed to be close according to their common underlying smoothness assumption: the true mapping from input to output

is smooth. We give a detailed description of these approaches in **chapter 3**.

Except for regularization, there are also approaches [36, 37] based on generative adversarial networks (GAN) [35]. As we know, GAN consists of a discriminator and a generator. In these approaches, the classifier takes place of the discriminator. Therefore, the classifier needs to not only classify labeled data but also distinguish between true data and fake data generated by the generator.

1.2 Contributions and Outline

The contributions of this thesis are summarized as follows:

1. We view the parametric classification model as a composition of two parts: feature extractor and classifier. Then, we find that it's favorable to align the distributions of labeled data and unlabeled data in feature space in order to improve generalization. Therefore, we propose a simple but effective regularization technique which aims for aligning distributions of labeled data and unlabeled data in feature space. It can be easily combined with existing semi-supervised deep learning approaches.

2. Empirically, we applied our proposed regularization technique in state-of-the-art semi-supervised deep learning approach and achieved new result on CIFAR-10 and SVHN datasets with varying number of labeled samples. Especially in the case of 1,000 labeled samples on CIFAR-10 dataset, the error rate is reduced from $15.48\% \pm 0.78\%$ to $9.92\% \pm 0.58\%$.

The rest of this thesis is organized as follows:

Chapter 2: We provide relevant preliminaries which are helpful for understanding this thesis.

Chapter 3: We describe semi-supervised deep learning approaches with consistency regularization.

Chapter 4: We introduce the motivation of our proposed method and the detailed algorithm as well as the experiments we conducted.

This thesis was typeset using LaTeX. All the simulations were preformed by PYTHON.

Notation

The notation used in this thesis are collected in Table 1.1.

Table 1.1: Notation in this thesis

Symbol	*Meaning*
$\mathbb{R}$ ($\mathbb{C}$)	The set of all real (complex) number values
$\mathbb{R}^n$	the set of n-dimensional vectors
N_c	Number of classes
N_l	Number of labeled samples
N_u	Number of unlabeled samples
N_e	Number of test samples
$\mathcal{D}_l$	Labeled dataset $\{(x_l^i, y_l^i)\|i = 1, \ldots, N_l\}$
$\mathcal{D}_u$	Unlabeled dataset $\{x_u^i\|i = 1, \ldots, N_u\}$
$\mathcal{D}_e$	Test dataset $\{(x_e^i, y_e^i)\|i = 1, \ldots, N_e\}$
$F(\cdot)$	Feature extractor
$C(\cdot)$	Classifier
$D(\cdot)$	Domain critic
$\mathbb{P}$	Probability density function in input space
$\mathbb{H}$	Probability density function in feature space
$\mathbb{C}_+$	The right half plane
$\mathbb{R}_+^n$	the set of n-dimensional vectors with positive entries
A^T (A^H)	The (conjugate) transpose of A
$\sigma_s(A)$	The spectrum of A
$\rho(A)$	The spectral radius of A
$C(c, r)$ ($\bar{C}(c, r)$)	The open (closed) disk centered at c with radius r
$\mathbb{Z}^+$	The set of nonnegative integers
$\|\| \cdot \|\|$	The spectral norm of a matrix.
$\|\| \cdot \|\|_\infty$	The induced infinity norm of a matrix.
$A^\dagger$	The pseudoinverse of a matrix A
$A^{\frac{1}{2}}$	The square root of a positive semi-definite matrix A
$\mathbf{1}_n$	The n-dimensional column vector with all elements 1
$\otimes$	The Kronecker product of matrices.
$\lfloor a \rfloor$	The largest integer less than or equal to the real number a
$A \geq B$	$A - B$ is positive semidefinite
$A > B$	$A - B$ is positive definite

Chapter 2

Preliminaries

In this chapter, we review relevant basics in probability theory, information theory and machine learning.

2.1 Probability Theory

Sample space and event Generally, the outcome of an experiment is not predictable with certainty. Although the outcome of the experiment can not be known in advance, the set of all possible outcomes is usually known. This set is referred to as the *sample space* of the experiment and is denoted by S. Any subset of the sample space is known as an *event* and is denoted by E. We say that E occurs if the outcome of the experiment is contained in E.

Probability of an event One way of defining the probability of an event is in terms of its relative frequency. Suppose that an experiment is repeatedly performed for n times under exactly the same conditions and the number of times that an event E occurs is $n(E)$. Then the probability of the event, $P(E)$, is defined as:

$$P(E) = \lim_{n\to\infty} \frac{n(E)}{n} \tag{2.1}$$

That is, $P(E)$ is defined as the limiting frequency of E occurring.

Random variable Rather than the actual outcome, we are usually more interested in some function of the outcome when an experiment is performed. For example, in the experiment of flipping a coin, we may care about the total number of heads that occurs and not interested in the actual outcome of head-tail sequence. The quantity of interest, or, more formally, the real-valued function defined on the sample space, is known as *random variable*.

PMF and PDF A random variable is said to be discrete if it can take on at most a countable number of possible values. For a discrete random variable X, we define the *probability mass function* (PMF) of X by

$$p(x) = P\{X = x\} \tag{2.2}$$

which indicates the probability of X taking on the value x. Except for discrete random variables, there also exist continuous random variables. We say that a random variable X is a continuous random variable if there exists a nonnegative function f, defined for all real $x \in (-\infty, \infty)$, having the property that, for any set B of real numbers,

$$P\{X \in B\} = \int_B f(x)dx \tag{2.3}$$

The function f is called the *probability density function* (PDF) of the random variable X. Both PMF and PDF are normalized: we have $\sum_x P(X = x) = 1$ for PMF and $\int_x p(x)dx$ for PDF.

2.1.1 Different Measures of Closeness between Distributions

Let $\mathcal{X}$ be a compact metric set and let Σ denote the set of all the Borel subsets of $\mathcal{X}$. Let Prob($\mathcal{X}$) denote the space of probability measures defined on $\mathcal{X}$. We can now define elementary distances and divergences between two distributions $\mathbb{P}_r, \mathbb{P}_g \in \text{Prob}(\mathcal{X})$:

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \tag{2.4}$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r || \mathbb{P}_g) = \int \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) d\mu(x) \tag{2.5}$$

  where both $\mathbb{P}_r$ and $\mathbb{P}_g$ are assumed to be absolutely continuous, and therefore admit densities, with respect to a same measure μ defined on $\mathcal{X}$.[1] The KL divergence is famously assymetric and possibly infinite when there are points such that $P_g(x) = 0$ and $P_r(x) > 0$.

[1] Recall that a probability distribution $\mathbb{P}_r \in \text{Prob}(\mathcal{X})$ admits a density $p_r(x)$ with respect to μ, that is, $\forall A \in \Sigma$, $\mathbb{P}_r(A) = \int_A P_r(x)d\mu(x)$, if and only it is absolutely continuous with respect to μ, that is, $\forall A \in \Sigma$, $\mu(A) = 0 \Rightarrow \mathbb{P}_r(A) = 0$.

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r||\mathbb{P}_m) + KL(\mathbb{P}_g||\mathbb{P}_m) \tag{2.6}$$

  where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[||x - y||\right], \tag{2.7}$$

  where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

2.2 Information Theory

Self-information and entropy Self-information is a measure for information content of an event x, which is defined as:

$$I(x) = -\log P(x) \tag{2.8}$$

Entropy measures the uncertainty contained in a probability distribution and is defined as:

$$H(x) = \mathbb{E}_{x\sim P}[I(x)] \tag{2.9}$$

Relative entropy and cross entropy Relative entropy of a probability distribution $P(x)$ with respect to another probability distribution $Q(x)$ is defined as:

$$D_{KL}(P||Q) = \mathbb{E}_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right] \tag{2.10}$$

It is also known as Kullback-Leibler (KL) divergence. Cross entropy is defined as follows:

$$H(P, Q) = -\mathbb{E}_{x\sim P}[\log Q(x)] \tag{2.11}$$

2.3 Machine Learning Tasks

In machine learning, input data can be seen as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of samples and each sample is represented as a d-dimensional vector and is assumed to be drawn independently from the data generating distribution. Based on the existence of labels, machine learning can be classified into three categories: supervised learning, unsupervised learning and semi-supervised learning. If for every input $\mathbf{x}_i$, there is a corresponding label y_i, then it's supervised learning; if there is no label for any input $\mathbf{x}_i$, then it's unsupervised learning; if only for some inputs $\mathbf{x}_i$, there are corresponding labels y_i, then it's semi-supervised learning. There are two common categories of machine learning tasks: classification and regression. In classification, the label y_i belongs to one of a predefined number of classes; in regression, the label y_i is a continuous value.

2.3.1 Linear Regression

Linear regression takes as input a vector $\mathbf{x} \in \mathbb{R}^d$ and aims to predict its label $y \in \mathbb{R}$ using an affine function:

$$\hat{y}(\mathbf{x}; \theta) = \theta^\top \mathbf{x} + b \tag{2.12}$$

where $\hat{y}$ is the predicted value of y and $\theta \in \mathbb{R}^d$ is vector of weights or parameters and b is an intercept or bias term.

The weights θ is estimated by minimizing the model's error, which is a measure of the closeness between the model's prediction $\hat{y}$ and the true value y. A common error measure is mean squared error, which is defined as follows:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2.13}$$

2.3.2 Classification

In the case of binary classification, we have two classes, class 0 and class 1. We can model the conditional probability of one class using a composition of sigmoid function and linear regression:

$$\hat{p}(y = 1 | \mathbf{x}; \theta) = \hat{y}(\mathbf{x}; \theta) = \sigma(\theta^\top \mathbf{x}) \tag{2.14}$$

where the sigmoid function $\sigma : \mathbb{R} \to (0, 1)$ is defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

Since there are just two classes, the probability of the other class is simply:

$$\hat{p}(y = 0|\mathbf{x}; \theta) = 1 - \hat{y} \tag{2.16}$$

For multi-class classification with C classes, each class is assigned a vector of weights θ_i $(i = 1, 2, ..., C)$ and the conditional probability of each class is modeled as a composition of softmax function and linear regression:

$$\hat{p}(y = i|\mathbf{x}; \theta) = \frac{e^{\theta_i^\top \mathbf{x}}}{\sum_{j=1}^{C} e^{\theta_j^\top \mathbf{x}}} \tag{2.17}$$

where the denominator is the so-called partition function that normalizes the distribution by summing over the scores for all C classes.

Then we can use Eq. (2.11) to calculate the cross-entropy loss between the true conditional probability $p(y|\mathbf{x})$ and the model's predicted probability $\hat{p}(y|\mathbf{x}; \theta)$ for each sample $\mathbf{x}$:

$$H(p, \hat{p}) = -\sum_{i=1}^{C} p(y = i|\mathbf{x}) \log \hat{p}(y = i|\mathbf{x}; \theta) \tag{2.18}$$

For binary classification, this simplifies to:

$$H(p, \hat{p}) = -(1 - y) \log(1 - \hat{y}) - y \log \hat{y} \tag{2.19}$$

Similar to linear regression, the weights θ can be estimated by minimizing the cross entropy loss.

2.4 Neural Networks

In linear regression, we use an affine function to make predictions. However, the underlying relationship between input $\mathbf{x}$ and its label y is not linear in general and thus using affine function is not proper. Neural network is a family of parametric functions that have specific structure and can approximate any function easily. We denote it by $f(\cdot; \theta)$.

2.4.1 Fully-connected Neural Networks

Suppose the input $\mathbf{x}$ are vectors that have 3 dimensions and can be classified into two classes. As shown in Figure 2.1, we model the relationship between input and label using fully-connected neural network, then we have:

$$f(\mathbf{x};\theta) = s(W_3\sigma(W_2\sigma(W_1\mathbf{x})))$$

where $W_1 \in \mathbb{R}^{4\times3}$, $W_2 \in \mathbb{R}^{4\times4}$, $W_3 \in \mathbb{R}^{2\times4}$; $\sigma(\cdot), s(\cdot)$ are nonlinear activations.

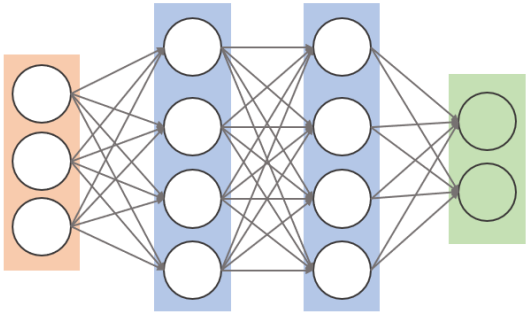


Figure 2.1: An example of fully-connected neural network

2.4.2 Convolutional Neural Networks

Fully-connected neural networks are suitable for processing data that can be naturally represented by vectors, which are of 1-dimensional. However, images are of 3-dimensional. For example, in the CIFAR-10 dataset, images are of size 32x32x3. Here, 32 means the number of pixels along width and height while 3 is number of the color channels. If using fully-connected neural network to process these images, then we have to flat them as vectors which would have 32*32*3=3072 dimensions and each neuron in the first hidden layer will have 3072 weights because of full connectivity. This size seems still manageable, but clearly fully-connected neural networks do not scale to larger images.

Convolutional neural networks (CNNs) are made up of three main types of layers: convolutional layer, pooling layer and fully-connected layer. The convolutional layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. During the forward pass, we slide each filter across the width and height of the input volume and compute dot products between the filter and the input. As we slide the filter over the width and height of the input volume, we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters

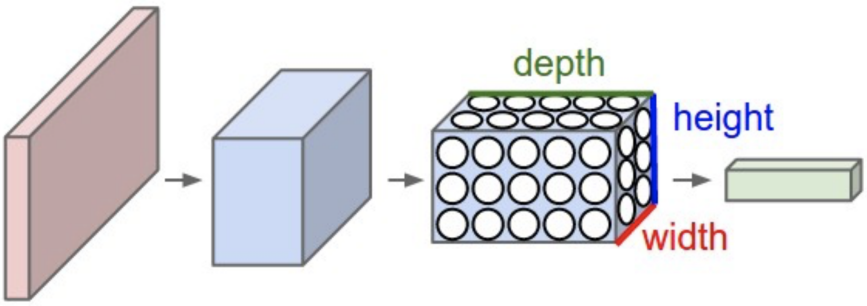


Figure 2.2: An example of convolutional neural network

that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each convolutional layer, and each of them will produce a 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

In CNNs, it's common to insert a pooling layer in-between successive convolutional layers. The pooling layer operates independently on every depth slice of the input volume and resizes it spatially, using the MAX operation. For example, when the pooling layer using filters of size 2x2 and strides of 2, the input volume will be downsampled by 2 along both width and height, as shown in Figure 2.3.

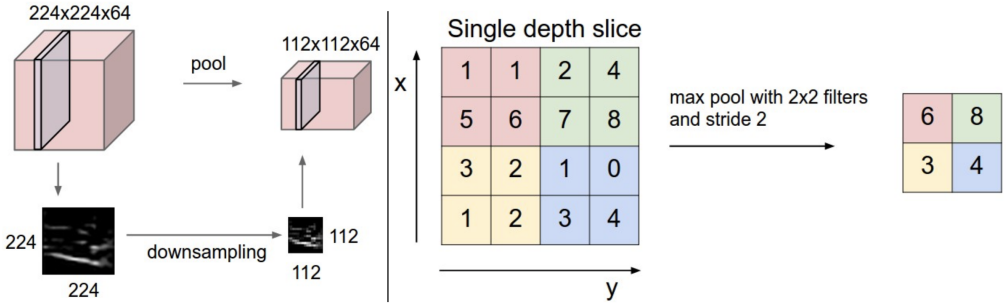


Figure 2.3: Pooling layer

Chapter 3

Semi-supervised Deep Learning by Regularization

In semi-supervised learning scenario, the training dataset consists of both labeled data and unlabeled data. We denote the labeled dataset as $\mathcal{D}_l = \{(x_l^i, y_l^i)|i = 1, \ldots, N_l\}$, where N_l is the number of labeled samples and denote the unlabeled dataset as $\mathcal{D}_u = \{x_u^i|i = 1, \ldots, N_u\}$, where N_u is the number of unlabeled samples.

Suppose the input data is of d dimension, and the number of classes is N_c. A standard classification model using deep neural network is essentially a parametric function, which we denote as $M(\cdot;\theta) : \mathbb{R}^d \mapsto [0,1]^{N_c}$. Semi-supervised deep learning by regularization can be summarized in the following framework:

$$\min_{\theta} \frac{1}{N_l} \sum_{i=1}^{N_l} L(M(x_l^i;\theta), y_l^i) + wR(\theta, \mathcal{D}_l, \mathcal{D}_u) \tag{3.1}$$

where $L(\cdot,\cdot)$ is a pre-defined loss function such as cross-entropy loss. Different approaches vary in the way to define the regularization term R, which leverages unlabeled data. w is a non-negative parameter that controls the importance of the regularization term.

3.1 Consistency Regularization

Recently, a series of semi-supervised deep learning approaches that use consistency loss as regularization were proposed in [30–32]. In these papers, two outputs are calculated for each input and the consistency loss is defined as the expected distance between the two outputs:

$$R(\theta, \mathcal{D}_l, \mathcal{D}_u) = \mathbb{E}_{x,\xi',\xi}\left[d(M(x;\theta',\xi'), M(x;\theta,\xi))\right] \tag{3.2}$$

where $M(x;\theta',\xi')$ and $M(x;\theta,\xi)$ are the two outputs for the same input x, ξ' and ξ are different perturbations applied to x, θ' and θ are different values of trainable parameters, $d(\cdot,\cdot)$ is a non-negative function that measures the distance between the two outputs. Before introducing these approaches in detail, we remind readers in advance that these approaches vary in the way to define the two different outputs.

3.1.1 Π-Model and Temporal Ensembling

Π-model and temporal ensembling are both proposed in [30]. In Π-model, the two outputs are calculated by perturbing both of the input and the network. The input is perturbed by adding Gaussian noise and stochastic augmentations such as random crop while the network is perturbed using dropout technique [48], which randomly removes each node together with its incoming and outgoing connections. As shown in Figure 3.1, $z_i = M(x_i;\theta,\xi)$ and $\tilde{z}_i = M(x_i;\theta',\xi')$ are the two outputs for the same input x_i.

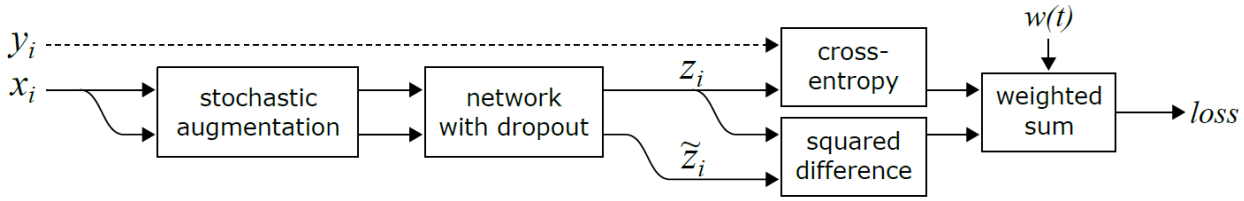


Figure 3.1: Diagram of Π-model

Π-model calculates two outputs for each input from scratch, which is time-consuming. To address this issue, temporal ensembling defines the output $M(x;\theta',\xi')$ as a weighted average of previous outputs for x:

$$\tilde{z} \leftarrow \alpha\tilde{z} + (1-\alpha)z \tag{3.3}$$

where $\tilde{z} = M(x;\theta',\xi')$ and $z = M(x;\theta,\xi)$ is output for x in current epoch. Since it takes advantage of outputs for x in previous epochs and calculates output only once for each input per epoch, it gains an approximate 2x speedup over the Π-model. We show the diagram of temporal ensembling in Figure 3.2.

3.1.2 Mean Teacher

A disadvantage of temporal ensembling is that it incorporates learned information slowly since it updates each target only once per epoch, i.e., Eq. (3.3) is executed after each epoch. The larger the dataset, the longer the span of the updates. Therefore it does

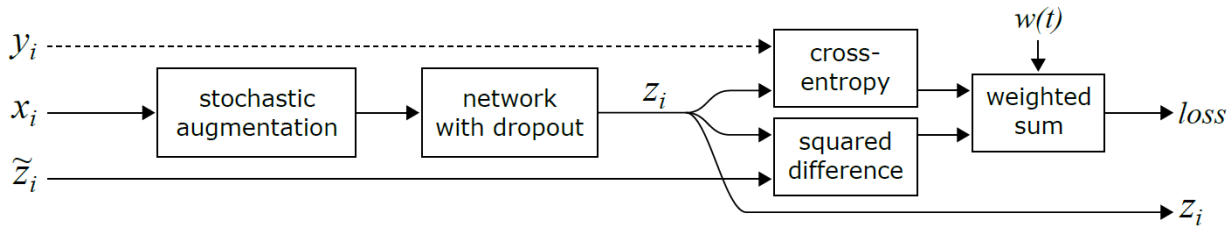


Figure 3.2: Diagram of temporal ensembling

not scale to large datasets and can not be used at all in the case of on-line learning. To overcome this limitation, Mean Teacher [31] proposed to average model weights instead of outputs. As shown in Figure 3.3, it defines θ' as exponential moving average of θ:

$$\theta' \leftarrow \alpha\theta' + (1-\alpha)\theta \tag{3.4}$$

Different from Eq. (3.3), Eq. (3.4) is executed after each step. Therefore, the teacher prediction $M(x;\theta',\xi')$ is more accurate.

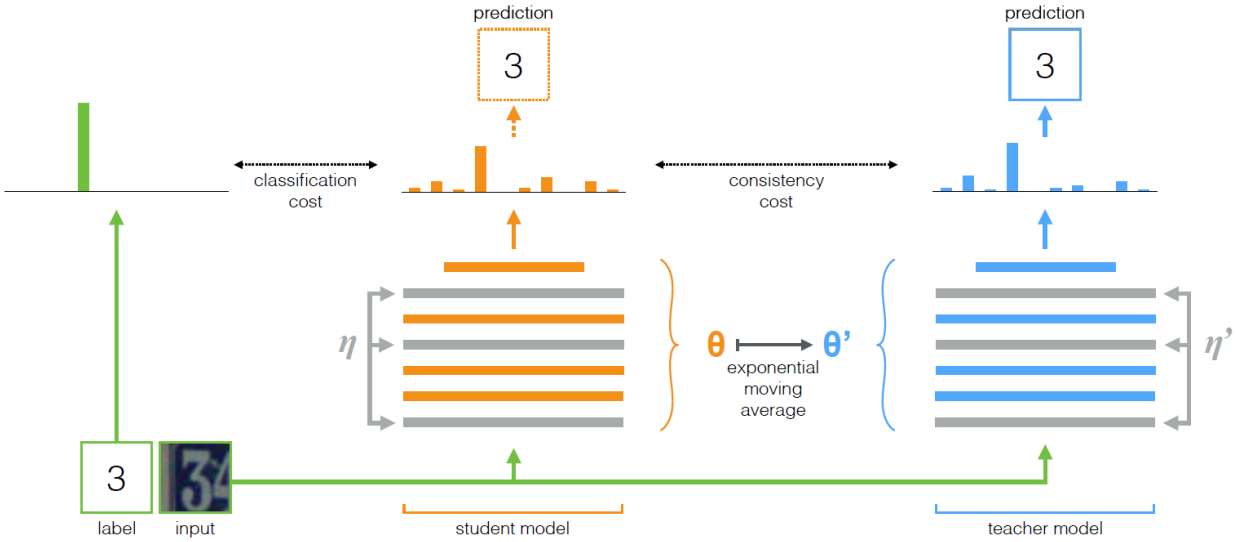


Figure 3.3: The mean teacher method

3.1.3 Virtual Adversarial Training (VAT)

Different from Π-Model which perturbs input in a random direction, VAT [32] perturbs input in the direction that maximizes the distance between outputs for original input and

perturbed input:

$$M(x;\theta',\xi') := M(x+\xi';\theta) \tag{3.5}$$

where

$$\xi' := \arg\max_{||r||_2\leq\epsilon} d\left(M(x+r;\theta), M(x;\theta)\right) \tag{3.6}$$

and $d(\cdot,\cdot)$ is the same as in Eq. (3.2). Note that ξ' in (3.6) is virtual adversarial because it's adversarial to prediction $M(x;\theta)$ instead of true label of x. That's why this approach is named so.

3.2 Interpolation Consistency Training (ICT)

ICT enforces the classification model $M(\cdot;\theta)$ to give consistent predictions at interpolations of unlabeled points. It takes advantage of the MixUp [58] operation:

$$I_\lambda(a,b) := \lambda a + (1-\lambda)b \tag{3.7}$$

where $\lambda \sim Uniform(0,1)$. As shown in Figure 3.4, x_u^j and x_u^k represent two unlabeled samples from $\mathcal{D}_u$. ICT defines the regularization term as follows:

$$R(\theta,\mathcal{D}_l,\mathcal{D}_u) := \mathbb{E}_{x_u^j,x_u^k\in\mathcal{D}_u} d\left(M(I_\lambda(x_u^j,x_u^k);\theta), I_\lambda(M(x_u^j;\theta'),M(x_u^k;\theta'))\right) \tag{3.8}$$

where θ' is defined in the same way as Eq. (3.4) and $d(\cdot,\cdot)$ is a non-negative function that measures the distance.

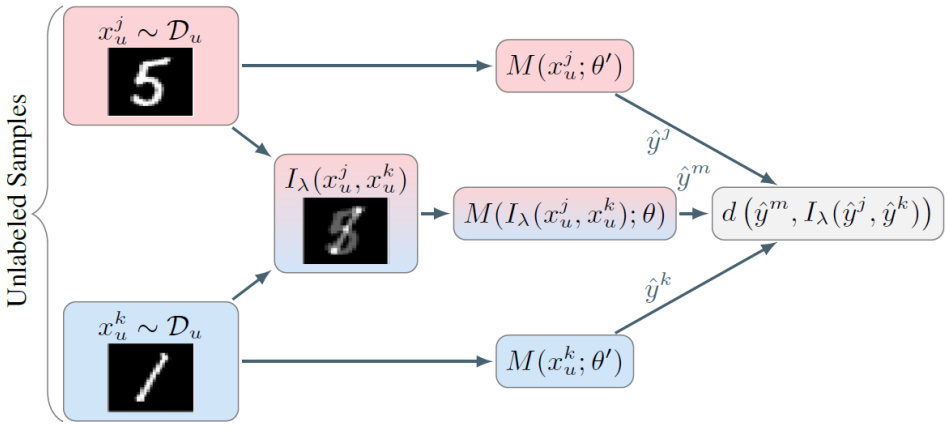


Figure 3.4: Diagram of ICT

Chapter 4

Wasserstein Distance Regularization for SSL

A standard classification model can be viewed as a composition of two components: feature extractor and classifier. Feature extractor maps from input space to feature space while classifier maps from feature space to a normalized distribution over classes. We denote the dimension of feature space as k and then we can denote the feature extractor as $F(\cdot;\theta_f) : \mathbb{R}^d \mapsto \mathbb{R}^k$ and denote the classifier as $C(\cdot;\theta_c) : \mathbb{R}^k \mapsto [0,1]^{N_c}$.

4.1 Align Distributions in Feature Space

Our goal is to minimize the population risk:

$$\min_{\theta_f,\theta_c} \mathbb{E}_{x\sim\mathbb{P}_*} \left[L(C(F(x)), y)\right] \tag{4.1}$$

where $\mathbb{P}_*$ is the data-generating distribution and $L(\cdot,\cdot)$ is the predefined loss function such as cross-entropy loss. However, (4.1) can not be solved directly since $\mathbb{P}_*$ is unknown. As a surrogate, the following optimization problem is solved in supervised learning:

$$\min_{\theta_f,\theta_c} \mathbb{E}_{x_l\sim\mathbb{P}_l} \left[L(C(F(x_l), y_l)\right] \tag{4.2}$$

where $\mathbb{P}_l$ is the empirical distribution of labeled data.

From (4.1) and (4.2), we can see that models obtained by supervised learning only generalize well when $\mathbb{P}_l$ approximates $\mathbb{P}_*$ well. And the condition of $\mathbb{P}_l$ being a good approximation of $\mathbb{P}_*$ is that there exists sufficient labeled data. That's the reason why supervised learning usually requires large amount of labeled data.

In practice, we usually have access to limited labeled data, which is not enough for training a good model using supervised learning. To address this issue, we rewrite (4.1) as:

$$\min_{\theta_f,\theta_c} \mathbb{E}_{h\sim\mathbb{H}_*} \left[L(C(h), y)\right] \tag{4.3}$$

where $h = F(x)$ and $\mathbb{H}_*$ is the distribution of h. Similarly, we rewrite (4.2) as:

$$\min_{\theta_f,\theta_c} \mathbb{E}_{h_l\sim\mathbb{H}_l} \left[L(C(h_l), y_l)\right] \tag{4.4}$$

where $h_l = F(x_l)$ and $\mathbb{H}_l$ is the distribution of h_l.

We denote $\mathbb{P}_u$ as the empirical distribution of unlabeled data. Let $h_u = F(x_u)$ and $\mathbb{H}_u$ be distribution of h_u. From (4.3) and (4.4), we observe that it's favorable to make $\mathbb{H}_l$ and $\mathbb{H}_*$ be close. Therefore, we propose a regularization term that aims for aligning $\mathbb{H}_l$ and $\mathbb{H}_u$ because $\mathbb{H}_u$ is much closer to $\mathbb{H}_*$.

4.2 Wasserstein Distance and Algorithm

Combined with supervised learning objective, our optimization problem can be formulated as follows:

$$\min_{\theta_f,\theta_c} \mathbb{E}_{x_l\sim\mathbb{P}_l} \left[L\left(C(F(x_l)), y_l\right)\right] + wW(\mathbb{H}_l, \mathbb{H}_u) \tag{4.5}$$

where w is a non-negative weight and $W(\mathbb{H}_l, \mathbb{H}_u)$ is the first Wasserstein distance (Earth-Mover distance) between $\mathbb{H}_l$ and $\mathbb{H}_u$. Earth-Mover (EM) distance is a measure of closeness between probability distributions. There are many other ways to measure the closeness between two probability distributions, such as the Total Variation (TV) distance, the Kullback-Leibler (KL) divergence and the Jensen-Shannon (JS) divergence (cf. section 2.1.1 for definitions). However, in the field of generative adversarial networks, [49] shows that the KL, JS, and TV distances are not sensible cost functions when learning distributions supported by low dimensional manifolds while the EM distance is sensible in that setup. In addition, the EM distance was also used in domain adaptation [52] and distributionally robust learning [53], which gained remarkable success. According to the Kantorovich-Rubinstein theorem [51], the dual form of the first Wasserstein distance between $\mathbb{H}_l$ and $\mathbb{H}_u$ can be written as:

$$W(\mathbb{H}_l, \mathbb{H}_u) = \sup_{D\in\mathcal{D}} \mathbb{E}_{h_l\sim\mathbb{H}_l}[D(h_l)] - \mathbb{E}_{h_u\sim\mathbb{H}_u}[D(h_u)] =: \sup_{D\in\mathcal{D}} L_{wd} \tag{4.6}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions that map from feature space to real values and $D(\cdot) : \mathbb{R}^k \mapsto \mathbb{R}$ is known as domain critic.

In practice, we set $D(\cdot)$ as parametric function $D(\cdot; \theta_d)$. To enforce the 1-Lipschitz constraint on $D(\cdot; \theta_d)$, [49] proposed to clip parameters θ_d within a compact space $[-c, c]$. However, it may downgrade model capacity and cause gradient vanishing or exploding problems [50]. To avoid these problems, [50] proposed an alternative way which penalizes the gradient of domain critic with respect to its input:

$$W(\mathbb{H}_l, \mathbb{H}_u) \approx \max_{\theta_d} \{L_{wd}(\theta_f, \theta_d) - \gamma L_{grad}(\theta_f, \theta_d)\} \tag{4.7}$$

where $L_{grad}(\theta_f, \theta_d) = \mathbb{E}_{h_i \sim \mathbb{H}_i} [(||\nabla_{h_i} D(h_i)||_2 - 1)^2]$ and $\mathbb{H}_i$ is uniform distribution over the line segment connecting h_l and h_u. Substitute (4.7) into (4.5), we end up getting a minmax problem:

$$\min_{\theta_f, \theta_c} \left\{ \mathbb{E}_{x_l \sim \mathbb{P}_l} [L(C(F(x_l)), y_l)] + w \max_{\theta_d} \{L_{wd}(\theta_f, \theta_d) - \gamma L_{grad}(\theta_f, \theta_d)\} \right\} \tag{4.8}$$

To solve the minmax problem (4.8), we alternatively solve the minimization and the maximization problem. As shown in Algorithm 2, the inner loop (from line 7 to line 15) is for solving the maximization part where θ_f and θ_c are viewed as constants while the parameter θ_d is updated (line 14). After solving the maximization part, we then solve the minimization part where θ_d will be viewed as constant while θ_f and θ_c will be updated (line 18 and line 19). Note that exactly computing L_{wd} is impossible, therefore we approximate it by sampling minibatches of labeled data and unlabeled data and then computing the average over the minibatches as shown in line 16 and line 8. Similarly, we approximate L_{grad} by sampling interpolations between sampled labeled and unlabeled data in feature space and then compute the average as shown from line 9 to line 13.

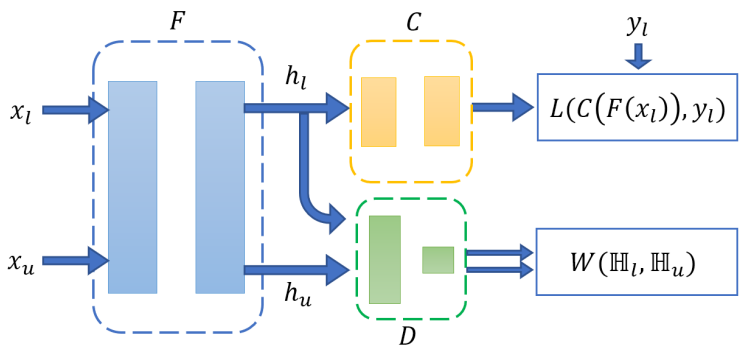


Figure 4.1: Wasserstein distance regularized SSL

Algorithm 2 WDR for Semi-supervised Learning

Require: $F(\cdot;\theta_f)$, feature extractor with trainable parameters θ_f
Require: $D(\cdot;\theta_d)$, domain critic with trainable parameters θ_d
Require: $C(\cdot;\theta_c)$, classifier with trainable parameters θ_c
Require: $\mathcal{D}_l$, labeled dataset
Require: $\mathcal{D}_u$, unlabeled dataset
Require: B, minibatch size
Require: $w(T)$, ramp function for increasing the importance of regularization term
Require: N, total number of iterations
Require: n, steps of training domain critic per iteration
Require: Q, uniform distribution on [0,1]

for $T = 1, 2..., N$ **do**
  $\{(x_l^j, y_l^j)\}_{j=1}^B \sim \mathcal{D}_l$ ▷ Sample a minibatch of labeled data
  $h_l^j = F(x_l^j), \quad j = 1, ..., B$ ▷ Compute features of sampled labeled data
  $L_{sup} = \frac{1}{B}\sum_{j=1}^B L(C(h_l^j), y_l^j)$ ▷ Compute supervised loss (cross-entropy)
  $\{x_u^j\}_{j=1}^B \sim \mathcal{D}_u$ ▷ Sample a minibatch of unlabeled data
  $h_u^j = F(x_u^j), \quad j = 1, ..., B$ ▷ Compute features of sampled unlabeled data
  for $t = 1, \ldots, n$ **do**
    $L_{wd} = \frac{1}{B}\sum_{j=1}^B [D(h_l^j) - D(h_u^j)]$
    for $j = 1, ..., B$ **do**
      $\alpha \sim Q$ ▷ Sample an interpolation coefficient
      $h_i^j = \alpha h_l^j + (1-\alpha) h_u^j$ ▷ Compute interpolation
    end for
    $L_{grad} = \frac{1}{B}\sum_{j=1}^B (||\nabla_{h_i^j} D(h_i^j)||_2 - 1)^2$
    $\text{Step}(\theta_d, \nabla_{\theta_d}[-L_{wd} + \gamma L_{grad}])$ ▷ Update parameters θ_d using e.g. SGD, Adam
  end for
  $L_{wd} = \frac{1}{B}\sum_{j=1}^B [D(h_l^j) - D(h_u^j)]$ ▷ Compute Wasserstein distance
  $L = L_{sup} + w(T) \cdot L_{wd}$ ▷ Compute total loss
  $\theta_f \leftarrow \text{Step}(\theta_f, \nabla_{\theta_f} L)$ ▷ Update parameters θ_f using e.g. SGD, Adam
  $\theta_c \leftarrow \text{Step}(\theta_c, \nabla_{\theta_c} L_{sup})$ ▷ Update parameters θ_c using e.g. SGD, Adam
end for

4.3 Experiments

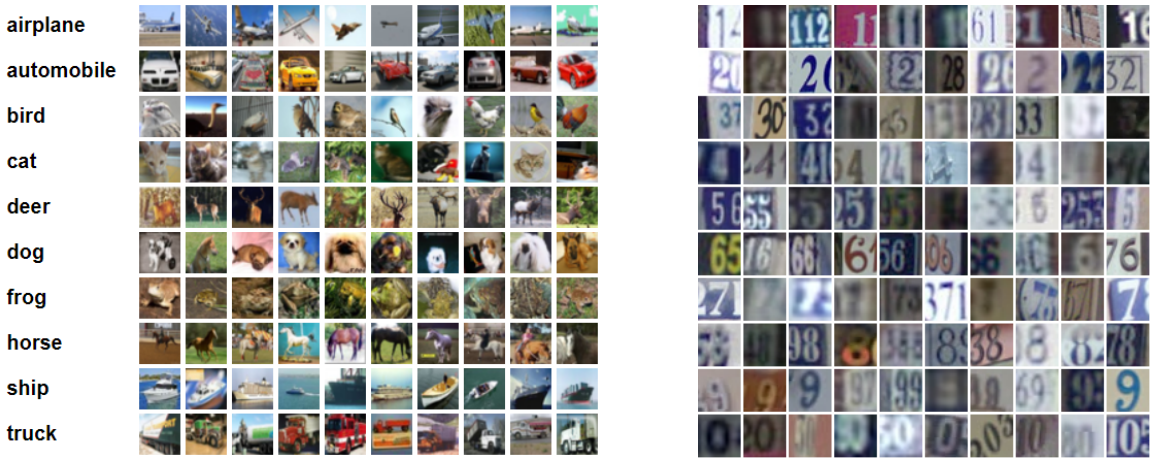
4.3.1 Datasets

Following the common practice in semi-supervised learning literature [30–33,47], we evaluate our model on the CIFAR-10 and Street View House Number (SVHN) benchmarks [56]. We use only a fraction of the training data as labeled data and the remaining data as unlabeled data.

- The CIFAR-10 dataset consists of 60K 32x32 RGB images, split into 50K training and 10K test images. All of the images belong to 10 classes of natural objects such as airplane, truck, horses and so on. Each class has the same number of images. See

some samples from the dataset[1] in Figure 4.2 (a).

- The SVHN dataset is a real-world image dataset obtained from house numbers in Google Street View images. It consists of 73,257 32x32 RGB images for training and 26,032 32x32 RGB images for testing. All of the images belong to 10 classes, which are digits from 0 to 9. See some samples from the dataset[2] in Figure 4.2 (b).



(a) CIFAR-10 (b) SVHN

Figure 4.2: Samples from CIFAR-10 and SVHN dataset

We adopt the standard data-augmentation and pre-processing procedure which has become standard practice in the semi-supervised learning literature [30–33, 47]. More specifically, for CIFAR-10, we first zero-pad each image with 2 pixels on each side. Then, the image is randomly cropped to produce a new 32x32 image. Next, the resulting image is horizontally flipped with probability 0.5, followed by per-channel standardization and ZCA preprocessing. For SVHN, we zero-pad each image with 2 pixels on each side and then randomly crop the resulting image to produce a new 32x32 image, followed by zero-mean and unit-variance image whitening. Besides, we also applied mixup [58] to augment data.

4.3.2 Implementation Details

We conduct our experiments using CNN-13 architecture since it has been adopted as the standard benchmark architecture in recent state-of-the-art SSL methods [30–33, 47]. We show the architecture details in Table 4.1, where feature extractor $F(\cdot, \theta_f)$ consists of layers from 'conv1a' to 'pool3' while classifier $C(\cdot, \theta_c)$ consists of layer 'dense' and layer

[1]https://www.cs.toronto.edu/ kriz/cifar.html
[2]http://ufldl.stanford.edu/housenumbers/

'output'.

Table 4.1: The model architecture.

Name	Description
input	32×32 RGB image
conv1a	128 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv1b	128 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv1c	128 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
pool1	Maxpool 2×2 pixels
drop1	Dropout, $p = 0.5$
conv2a	256 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv2b	256 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv2c	256 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
pool2	Maxpool 2×2 pixels
drop2	Dropout, $p = 0.5$
conv3a	512 filters, 3×3, pad = 'valid', LReLU ($\alpha = 0.1$)
conv3b	256 filters, 1×1, LReLU ($\alpha = 0.1$)
conv3c	128 filters, 1×1, LReLU ($\alpha = 0.1$)
pool3	Global average pool ($6 \times 6 \rightarrow 1 \times 1$ pixels)
dense	Fully connected $128 \rightarrow 10$
output	Softmax

For domain critic, we use a two-layer fully connected neural network and set the dimension of hidden layer to be 10 as shown in Table 4.2.

Table 4.2: The domain critic architecture.

Name	Description
input	128-dimensional feature vector
dense	Fully connected $128 \rightarrow 10$
activation	ReLU
output	Fully connected $10 \rightarrow 1$

We set the initial learning rate to be 0.1 and then annealed it using the same cosine annealing technique used by [31]. The momentum parameter is set to 0.9 and the L2 regularization coefficient is set to 0.0001. The number of steps per iteration for training domain critic is set to 3. The minibatch size B is set to 50. We list the settings of λ and γ in Table 4.3 since they vary in different experiments.

4.3.3 Results

The experimental results for CIFAR-10 and SVHN datasets is shown in Table 4.4 and table 4.5, respectively. We also visulaize them in Figure 4.3(a) and Figure 4.3(b), respectively.

Table 4.3: Settings of λ and γ for different experiments.

Experiments	Value of λ	Value of γ
CIFAR-10 1000 labeled	0.01	0.1
CIFAR-10 2000 labeled	0.01	0.1
CIFAR-10 4000 labeled	0.01	0.2
SVHN 250 labeled	0.004	0.2
SVHN 500 labeled	0.003	0.2
SVHN 1000 labeled	0.001	0.2

We can see that our proposed method reduced the error rate on both of CIFAR-10 and SVHN datasets for different number of labeled samples. Especially in the case of using 1,000 labeled samples on CIFAR-10 dataset, our method reduced the error rate from $15.48\% \pm 0.78\%$ to $9.92\% \pm 0.58\%$.

Table 4.4: Error rates (%) on CIFAR-10 using CNN-13 architecture.

Model	1000 labeled 45000 unlabeled	2000 labeled 45000 unlabeled	4000 labeled 45000 unlabeled
Supervised	39.95 ± 0.75	31.16 ± 0.66	21.75 ± 0.46
Supervised (Mixup) [58]	36.48 ± 0.15	26.24 ± 0.46	19.67 ± 0.16
Supervised (Manifold Mixup) [59]	34.58 ± 0.37	25.12 ± 0.52	18.59 ± 0.18
Π model [30]	31.65 ± 1.20	17.57 ± 0.44	12.36 ± 0.31
TempEns [30]	23.31 ± 1.01	15.64 ± 0.39	12.16 ± 0.24
MT [31]	21.55 ± 1.48	15.73 ± 0.31	12.31 ± 0.28
VAT [32]	–	–	$11.36 \pm$ NA
VAT+Ent [32]	–	–	$10.55 \pm$ NA
SNTG [33]	18.41 ± 0.52	13.64 ± 0.32	10.93 ± 0.14
ICT [47]	15.48 ± 0.78	9.26 ± 0.09	7.29 ± 0.02
ICT+WDR	$\mathbf{9.92 \pm 0.58}$	$\mathbf{8.42 \pm 0.10}$	$\mathbf{6.86 \pm 0.02}$

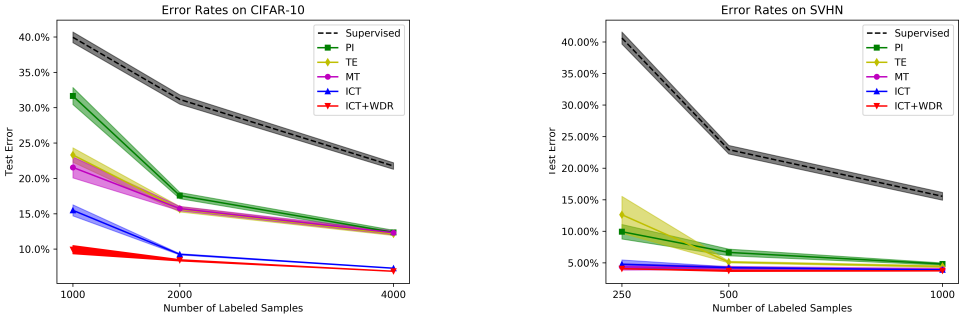
4.4 Conclusions and Future Work

We propose a simple but effective regularization technique for semi-supervised deep learning, which aims for aligning empirical distributions of labeled data and unlabeled data in feature space. It can be easily combined with most existing semi-supervised deep learning approaches. Empirically, we combine it with current state-of-the-art approach (ICT) and achieve new state-of-the-art performance on CIFAR-10 and SVHN datasets with varying number of labeled samples. Especially in the case of 1,000 labeled samples on CIFAR-10 dataset, the error rate is reduced from $15.48\% \pm 0.78\%$ to $9.92\% \pm 0.58\%$.

Table 4.5: Error rates (%) on SVHN using CNN-13 architecture.

Model	250 labeled 73257 unlabeled	500 labeled 73257 unlabeled	1000 labeled 73257 unlabeled
Supervised	40.62 ± 0.95	22.93 ± 0.67	15.54 ± 0.61
Supervised (Mixup) [58]	33.73 ± 1.79	21.08 ± 0.61	13.70 ± 0.47
Supervised (Manifold Mixup) [59]	31.75 ± 1.39	20.57 ± 0.63	13.07 ± 0.53
Π model [30]	9.93 ± 1.15	6.65 ± 0.53	4.82 ± 0.17
TempEns [30]	12.62 ± 2.91	5.12 ± 0.13	4.42 ± 0.16
MT [31]	4.35 ± 0.50	4.18 ± 0.27	3.95 ± 0.19
VAT [32]	–	–	$5.42 \pm$ NA
VAT+Ent [32]	–	–	$3.86 \pm$ NA
SNTG [33]	4.29 ± 0.23	3.99 ± 0.24	3.86 ± 0.27
ICT [47]	4.78 ± 0.68	4.23 ± 0.15	3.89 ± 0.04
ICT+WDR	$\mathbf{4.07 \pm 0.02}$	$\mathbf{3.75 \pm 0.11}$	$\mathbf{3.74 \pm 0.04}$

In future, we would like to explore more efficient and effective algorithms for estimating Wasserstein distance since it is a cirtical part of our method. We would also like to derive some theoretical results regarding the generalization ability.



(a) CIFAR-10 (b) SVHN

Figure 4.3: Test error rates on SVHN and CIFAR-10 with varying number of labeled samples 250, 500 and 1000. The mean and standard deviation of the error rates are shown as lines and shaded regions respectively.

Bibliography

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[2] Kalchbrenner, Nal, and Phil Blunsom. "Recurrent continuous translation models." Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013.

[3] Deng, Li, Geoffrey Hinton, and Brian Kingsbury. "New types of deep neural network learning for speech recognition and related applications: An overview." 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013.

[4] Castelli, Vittorio, and Thomas M. Cover. "On the exponential value of labeled samples." Pattern Recognition Letters 16.1 (1995): 105-111.

[5] Castelli, Vittorio, and Thomas M. Cover. "The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter." IEEE Transactions on information theory 42.6 (1996): 2102-2117.

[6] Ratsaby, Joel, and Santosh S. Venkatesh. "Learning from a mixture of labeled and unlabeled examples with parametric side information." Proceedings of the eighth annual conference on Computational learning theory. ACM, 1995.

[7] Cozman, Fabio G., Ira Cohen, and Marcelo C. Cirelo. "Semi-supervised learning of mixture models." Proceedings of the 20th International Conference on Machine Learning (ICML-03). 2003.

[8] Nigam, Kamal, et al. "Text classification from labeled and unlabeled documents using EM." Machine learning 39.2-3 (2000): 103-134.

[9] Nigam, Kamal, and Rayid Ghani. "Analyzing the effectiveness and applicability of co-training." Cikm. Vol. 5. 2000.

[10] Scudder, H., *Probability of error of some adaptive pattern-recognition machines.* IEEE Transactions on Information Theory, 11(3), 363-371. 1965.

[11] Fralick, S., *Learning to recognize patterns without a teacher.* IEEE Transactions on Information Theory, 13(1), 57-64. 1967.

[12] Agrawala, A, *Learning with a probabilistic teacher.* IEEE Transactions on Information Theory, 16(4), 373-379. 1970.

[13] Yarowsky, David. "Unsupervised word sense disambiguation rivaling supervised methods." 33rd annual meeting of the association for computational linguistics. 1995.

[14] Riloff, Ellen, Janyce Wiebe, and Theresa Wilson. "Learning subjective nouns using extraction pattern bootstrapping." Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. Association for Computational Linguistics, 2003.

[15] Maeireizo, Beatriz, Diane Litman, and Rebecca Hwa. "Co-training for predicting emotions with spoken dialogue data." Proceedings of the ACL 2004 on Interactive poster and demonstration sessions. Association for Computational Linguistics, 2004.

[16] Rosenberg, Chuck, Martial Hebert, and Henry Schneiderman. "Semi-supervised self-training of object detection models." (2005).

[17] Blum, Avrim, and Tom Mitchell. "Combining labeled and unlabeled data with co-training." Proceedings of the eleventh annual conference on Computational learning theory. ACM, 1998.

[18] Mitchell, Tom M. "The role of unlabeled data in supervised learning." Language, Knowledge, and Representation. Springer, Dordrecht, 2004. 103-111.

[19] Zhu, Xiaojin, and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[20] Zhou, Dengyong, et al. "Learning with local and global consistency." Advances in neural information processing systems. 2004.

[21] Szummer, Martin, and Tommi Jaakkola. "Partially labeled classification with Markov random walks." Advances in neural information processing systems. 2002.

[22] Zhu, Xiaojin, Zoubin Ghahramani, and John D. Lafferty. "Semi-supervised learning using gaussian fields and harmonic functions." Proceedings of the 20th International conference on Machine learning (ICML-03). 2003.

[23] Belkin, Mikhail, Irina Matveeva, and Partha Niyogi. "Regularization and semi-supervised learning on large graphs." International Conference on Computational Learning Theory. Springer, Berlin, Heidelberg, 2004.

[24] Delalleau, Olivier, Yoshua Bengio, and Nicolas Le Roux. "Efficient Non-Parametric Function Induction in Semi-Supervised Learning." AISTATS. Vol. 27. No. 28. 2005.

[25] Joachims, Thorsten. "Transductive learning via spectral graph partitioning." Proceedings of the 20th International Conference on Machine Learning (ICML-03). 2003.

[26] Belkin, Mikhail, and Partha Niyogi. "Using manifold stucture for partially labeled classification." Advances in neural information processing systems. 2003.

[27] Grandvalet, Yves, and Yoshua Bengio. "Semi-supervised learning by entropy minimization." Advances in neural information processing systems. 2005.

[28] Kingma, Durk P., et al. "Semi-supervised learning with deep generative models." Advances in neural information processing systems. 2014.

[29] Rasmus, Antti, et al. "Semi-supervised learning with ladder networks." Advances in neural information processing systems. 2015.

[30] Laine, Samuli, and Timo Aila. "Temporal ensembling for semi-supervised learning." arXiv preprint arXiv:1610.02242 (2016).

[31] Tarvainen, Antti, and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results." Advances in neural information processing systems. 2017.

[32] Miyato, Takeru, et al. "Virtual adversarial training: a regularization method for supervised and semi-supervised learning." IEEE transactions on pattern analysis and machine intelligence (2018).

[33] Luo, Yucen, et al. "Smooth neighbors on teacher graphs for semi-supervised learning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[34] Bachman, Philip, Ouais Alsharif, and Doina Precup. "Learning with pseudo-ensembles." Advances in Neural Information Processing Systems. 2014.

[35] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[36] Salimans, Tim, et al. "Improved techniques for training gans." Advances in neural information processing systems. 2016.

[37] Dai, Zihang, et al. "Good semi-supervised learning that requires a bad gan." Advances in neural information processing systems. 2017.

[38] M. Seeger, "Learning with labeled and unlabeled data," Univ. Edinburgh, Edinburgh, U.K., 2001. Tech. Rep.

[39] X. Zhu, "Semi-supervised learning literature survey," Comput. Sci. Dept., Univ. Wisconsin-Madison, Madison, WI, Tech. Rep. 1530, 2005.

[40] O. Chappelle, B. Schölkopf, and A. Zien, Eds., Semi-Supervised Learning. Cambridge, MA: MIT Press, 2006.

[41] Li, Ming, and Zhi-Hua Zhou. "Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples." IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 37.6 (2007): 1088-1098.

[42] Hosmer Jr, David W. "A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample." Biometrics (1973): 761-770.

[43] McLachlan, Geoffrey John. "Estimating the linear discriminant function from initial samples containing a small number of unclassified observations." Journal of the American statistical association 72.358 (1977): 403-406.

[44] O'neill, Terence J. "Normal discrimination with unclassified observations." Journal of the American Statistical Association 73.364 (1978): 821-826.

[45] McLachlan, Geoffrey J., and S. Ganesalingam. "Updating a discriminant function on the basis of unclassified data." Communications in Statistics-Simulation and Computation 11.6 (1982): 753-767.

[46] Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the Royal Statistical Society: Series B (Methodological) 39.1 (1977): 1-22.

[47] Verma, Vikas, et al. "Interpolation Consistency Training for Semi-Supervised Learning." arXiv preprint arXiv:1903.03825 (2019).

[48] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[49] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan." arXiv preprint arXiv:1701.07875 (2017).

[50] Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." Advances in Neural Information Processing Systems. 2017.

[51] Villani, Cedric. Optimal transport: old and new. Vol. 338. Springer Science and Business Media, 2008.

[52] Shen, Jian, et al. "Wasserstein distance guided representation learning for domain adaptation." arXiv preprint arXiv:1707.01217 (2017).

[53] Sinha, Aman, Hongseok Namkoong, and John Duchi. "Certifying some distributional robustness with principled adversarial training." arXiv preprint arXiv:1710.10571 (2017).

[54] Polyak, Boris T. "Some methods of speeding up the convergence of iteration methods." USSR Computational Mathematics and Mathematical Physics 4.5 (1964): 1-17.

[55] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." arXiv preprint arXiv:1608.03983 (2016).

[56] Netzer, Yuval, et al. "Reading digits in natural images with unsupervised feature learning." (2011).

[57] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[58] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.

[59] Verma, Vikas, et al. "Manifold mixup: Better representations by interpolating hidden states." International Conference on Machine Learning. 2019.

Biography

Conference Paper

1. Li Wenye, **Mao Jingwei**, Zhang Yin and Cui Shuguang. "Fast Similarity Search via Optimal Sparse Lifting." Advances in Neural Information Processing Systems. 2018.