Towards Lightweight Neural Networks with Few Data via Knowledge Distillation

A THESIS SUBMITTED TO
THE FACULTY OF ENGINEERING
OF THE UNIVERSITY OF SYDNEY
IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY

SHUMIN KONG

Supervisor: Dr. Chang Xu

School of Computer Science

Faculty of Engineering

The University of Sydney
Australia

8 March 2021

Authorship Attribution Statement

The work presented in this thesis is published as [1] in Proceedings of the 34th AAAI Conference on Artificial Intelligence, 2020.

Towards Lightweight Neural Networks with Few Data via Knowledge Distillation

Shumin Kong(Email: shumin.kong@protonmail.com)

Supervisor: Dr. Chang Xu

School of Computer Science

Faculty of Engineering

The University of Sydney

Statement of Original Authorship

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Signature:

Shumin Kong

To Those Whom I love & Those Who Love Me.

Abstract

The advancement of deep learning technology has been concentrating on deploying end-to-end solutions using high dimensional data, such as images. With the rapid increase in benchmark performance comes the significant resource requirements to train the network and make inferences with it. Deep learning models that achieve state-of-the-art benchmark result may require a huge amount of computing resources and data. To alleviate this problem, knowledge distillation with teacher-student learning has drawn much attention in compressing neural networks on low-end edge devices, such as mobile phones and wearable watches. However, current teacher-student learning algorithms mainly assume that the complete dataset for the teacher network is also available for the training of the student network. However, for real-world scenarios, users may only access to part of training examples due to commercial profits or data privacy, and severe over-fitting issues would happen as a result.

In this study, we tackle the challenge of learning student networks with few data by investigating the ground-truth data-generating distribution underlying these few data. Taking Wasserstein distance as the measurement, we assume that this ideal data distribution lies in a neighborhood of the discrete empirical distribution induced by the training examples. Thus we propose to safely optimize the worst-case cost within this neighborhood to boost the generalization. Furthermore, with theoretical analysis, we derive a novel and easy-to-implement loss for training the student network in an end-to-end fashion. Our analysis is empirically validated through experiments using benchmark datasets, and the result indicates the effectiveness of our proposed method.

Keywords

Deep Learning, Knowledge Distillation, Network Compression, Teacher-student learning, Computer Vision

Acknowledgements

Foremost, I would like to thank my supervisor Dr Chang Xu for his patience, motivation, enthusiasm, and immense knowledge, and his generous guidance during the completion of this thesis. My research would not have been possible without his support.

Finally I would like to thank my parents for supporting me in this research, who provided me with the educational foundation which made my contribution to this research possible.

Table of Contents

CHAPTER 1

Introduction

In recent years, computer vision research has rapidly advanced due to the success of deep neural networks. The image classification performance on large-scale datasets (e.g. ImageNet) has been constantly refreshed by various convolutional neural networks (CNNs), such as AlexNet [2], VGGNet [3], Inception [4] and ResNet [5]. Language models, such as GPT [6], BERT[7] and [8] has achieved superhuman performance on large textual dataset.

However, to achieve outstanding classification performance, these networks usually have a large volume of parameters and significant resource consumption. For example, to achieve a top-1 error rate of 22.16%, AlexNet requires more than 232 million parameters and more than 700 million multiplications to implement the prediction. This computational demand limits their application on low-end edge devices, such as mobile phones, tablets and wearable watches.

To minimize the resource required by deep neural networks, several techniques to directly compress existing trained networks are investigated, such as vector quantization [9], hash encoding [10], weight matrices decomposition [11] and using unlabeled data [12]. Others attempt to design efficient architectures to accelerate the inference speed, such as ResNeXt [13], Xception network [14] and MobileNets [15]. Besides these approaches, knowledge distillation paradigm, also known as teacher-student learning, serves as a complementary scheme to obtain light and efficient neural networks. By treating the pre-trained huge networks as teacher networks, the target small network is thus viewed as the student network and can be guided by the teacher network. [16] took a straightforward approach by directly minimizing the Euclidean distance between the feature maps generated from the teacher and student networks. The widely-used knowledge distillation method. [17] leaped forward by introducing a loss to encourage the student network to learn from the softened outputs of the teacher network. Others also attempt to further boost the performance by using multiple teachers [18] or investigating feature layers, such as FitNet [19] and activation boundary loss [20].

Current knowledge distillation algorithms usually assume that the complete dataset for training teacher network is also available for the learning of the student network. In real-world scenarios, however, users may only have a few data at hand. For instance, due to the consideration of commercial profits or data privacy, many applications do not open-source their large training dataset completely but only supply a fraction for verification purposes. The limited data usually induce severe over-fitting issues during the learning of the student network. Another example can be an off-line speech recognition network. In this case, a generic speech recognition network can be trained on a data center, which will be used as a teacher. Then, the end-user can provide a few samples of his or her speech to train the student network. The resulting student network would be small enough to run on a mobile device while maintaining the high-quality of the teacher network. The combination of benefits from the teacher-student learning paradigms and the generalization ability enables some large deep learning networks to be deployed to low battery and computing power devices, such as mobile phones. Thus, it is important to fit the network to scarce training examples while maintaining its generalization ability.

To boost the generalization ability of the student network, we propose to explore the ground-truth data-generating distribution. Given the training examples, we assume the ground-truth distribution lies in a neighborhood of the discrete empirical distribution, i.e. uniform distribution over i.i.d. training examples. By dint of Wasserstein distance, we thus propose to safely optimize the worst-case or every possible ground-truth distribution's cost within this neighborhood to boost the generalization as a result. However, the worst-case cost does not have a closed-form and is not appropriate for training a student network in an end-to-end fashion. In this way, we furthermore analyze its upper bound in theory and develop a novel loss function accordingly, called Wasserstein Generalization loss. As a result, via this very loss, the student network can maintain its generalization ability while being trained on a fraction of the training examples. Experimental results on benchmark datasets show the effectiveness of our proposed method, and when the training examples are very limited, our method significantly outperforms other comparison methods.

1.1 Contributions

The results and findings of this study have made the following important contributions to scientific knowledge:

(1) A formulation of the generalization ability of the student network in the context of knowledge distillation.
(2) An theoretical upper bound of the formulation proposed.
(3) A practical loss function for the student network, based on the theoretical upper bound derived.
(4) Empirical analyses and validations of the proposed loss function.

1.2 Thesis Structure

In Chapter 2, we review previous literature around deep learning, Wasserstein distance, and knowledge distillation.

In Chapter 3, we present our formulation of the problem using Wasserstein distance, perform theoretical analysis, and propose a practical loss function for the student network.

In Chapter 4, we validate our method empirically and perform analysis on some important hyperparameters.

Finally, we conclude on Chapter 5.

CHAPTER 2

Literature review

This chapter presents background knowledge and literature review for this study. We will start by introducing the Wasserstein distance and its applications in deep learning at Section 2.1. We will then proceed to Section 2.2, which analyses the advantages and weakness of some state-of-the-art knowledge distillation techniques.

2.1 Wasserstein Distance

In section introduces some background knowledge about Wasserstein distance, which forms the backbone of this study. The Wasserstein distance is a distance measure between probability distributions on a given distance space $(M; \rho)$, where $\rho(x; y)$ is a between two elements in set M.

DEFINITION 2.1.1. The Wasserstein distance between two probability measures $\mathbb{P}$ and $\mathbb{Q}$ is defined as

$$d_W(\mathbb{P}, \mathbb{Q}) := \inf_{\gamma \in \Pi(\mathbb{P}, \mathbb{Q})} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||],$$

where $\Pi(\mathbb{P}, \mathbb{Q})$ denotes the set of all joint distributions $\gamma(x, y)$ with marginals $\mathbb{P}$ and $\mathbb{Q}$, and $||\cdot||$ represents an arbitrary norm on $\mathbb{R}^m$.

Intuitively, $\gamma(x, y)$ denotes the units of mass must be transported from x to y, such that the distribution $\mathbb{P}$ can be transformed into the distribution $\mathbb{Q}$. The Wasserstein distance is the cost of the optimal transport plan.

2.1.1 Wasserstein Distance in Neural Networks

In comparison with other distances for probability distributions, such as Kullback–Leibler divergence [21], Wasserstein distance is unique for being able to measure between continuous and discrete distributions. This uniqueness enables the popularity of the Wasserstein distance in deep learning researches.

[22] empirically models the Wasserstein distance function as a neural network and apply it to the unsupervised domain adaptation problem. In the domain adaptation problem, the goal is to apply a machine learning model trained in the source domains to a target domain. During training, a domain critic is used in order to minimize the discrepancy between the source and target domains. In their work, the empirical Wasserstein distance will be used as the domain critic, which will be pre-trained to its optimality.

[23] uses Wasserstein distance to measure distributions in an adversarial learning scheme, or generative adversarial network [24]. The goal of the generative adversarial network is to train the networks such that the generator can generate data that look like real-world data, in other words, to approximate the data generating distribution. There are usually two networks in the system – a generator network G and a discriminator network D. The training objective for the generator is to generate samples that fool the discriminator, while the discriminator tries to distinguish between generated samples and real samples. The proposed method from [23] suggests that directly applying Wasserstein distance to measure the discrepancy between the generated distribution and the real data distribution improve the stability of the training process, and alleviate problems like mode collapse.

[25] proposes to view the multi-label learning problem from a distributional perspective. In their study, the semantic similarity between output dimensions is being considered. A Wasserstein loss for learning to predict a non-negative measure over a finite set, based on the Wasserstein distance, can then be defined to exploit this information to improve the multi-label classification performance of the network.

Most existing deep learning works that involve Wasserstein distance often include the Wasserstein distance as a part of their optimization objectives. In other words, the Wasserstein distance is being optimised in one way or another. However, this thesis incorporates the Wasserstein distance as a measurement as the constraint of the optimization.

2.2 Knowledge Distillation

Now we formally introduce the teacher-student learning scheme in compressing neural networks, especially the widely-used knowledge distillation method.

In the teacher-student learning scheme, the teacher network $\mathcal{N}_T$ usually consists of a large number of parameters and has powerful classification ability accordingly. In contrast, the student network $\mathcal{N}_S$ is light and small, which has much fewer parameters and is appropriate for the low-end computational devices. The goal of the teacher-student learning scheme is to learn the student network with the help of the pre-trained teacher network, instead of solely from the training data.

To transfer the knowledge from the teacher network into the student network, special training guidance or losses are imposed during the learning of student network, e.g. knowledge distillation (KD) loss [17]. Denote the training data as $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ where $\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ and $\boldsymbol{y}_i \in \mathcal{Y} \subset \{0,1\}^k$. KD loss encourages the student network to have similar softened outputs with that of teacher network,

$$\mathcal{L}_{KD}(\mathcal{N}_S) = \frac{1}{N}\sum_{i=1}^{N}[\mathcal{H}(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i) + C\mathcal{H}(\boldsymbol{p}_i, \boldsymbol{q}_i)], \tag{2.1}$$

where $\hat{\boldsymbol{y}}_i \in \mathbb{R}^k$ is the prediction output of the student network, C is a balancing constant and $\mathcal{H}(\cdot,\cdot)$ is the cross-entropy loss to measure the discrepancy between the prediction output vector and the ground-truth label vector. $\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ are called the softened outputs of the teacher network and student network, respectively, which are calculated using their raw output

Algorithm 1 Training the Student Network with FitNets [19].

```
Input: A teacher network $\mathcal{N}_T$; a student network $\mathcal{N}_S$
Output: A trained student network $\mathcal{N}_S$.
1: Initialize student network.
2: Select a middle layer for the teacher $W_T^m$ and student network $W_S^m$ and Initialize $W_H$.
3: $W_S^{m*} \leftarrow \arg\min_{W_S^m} L_{\mathcal{HT}}(W_S^m, W_H, W_T^m)$
4: Assign $W_S^{m*}$ back to $\mathcal{N}_S$.
5: $\mathcal{N}_S \leftarrow \arg\min_{\mathcal{N}_S} L_{\mathcal{KD}}(\mathcal{N}_S)$
```

logits $\boldsymbol{o}_i^T$ and $\boldsymbol{o}_i^S$ by softmax function, i.e.

$$\boldsymbol{p}_i = \frac{\exp(\boldsymbol{o}_i^T/T)}{\|\exp(\boldsymbol{o}_i^T/T)\|_1} \quad \text{and} \quad \boldsymbol{q}_i = \frac{\exp(\boldsymbol{o}_i^S/T)}{\|\exp(\boldsymbol{o}_i^S/T)\|_1}, \tag{2.2}$$

where T is a temperature parameter to control the *softness* of the probabilistic prediction outputs. The softened outputs contain more information than the one-hot-code ground-truth label vectors and are supposed to better guide the training of the student network.

2.2.1 Review of FitNets

The idea that to train a smaller network using privileged information from a larger network has enabled the emergence of researches around this topic. [19] proposed FitNets, which introduced a novel channel for the teacher network to provide privileged information to the student network.

In their approach, a hint layer W_H is introduced to reshape the dimensions of a middle layer on the student network W_S^m so that it matches a middle layer on the teacher network W_T^m. The training algorithm of such configuration is shown in Algorithm 1. Generally speaking, the training stage is divided into two stages. In the first stage, they minimize the difference between the output of W_T^m and W_H, with respect to a loss function $L_{\mathcal{HT}}$, where W_H is installed on the output of W_S^m. In this stage, shallower parts of the student network, or layers up until W_S^m, receives additional privileged information from the middle layers of the teacher network. In the second stage, W_H is discarded and the training proceeds with the original knowledge distillation loss.

Their approach successfully trained the student network that outperforms the teacher network with less number of parameters. However, one can argue that such success may come from a smart selection of the hyperparameters. To be specific, the benefits from the hint layers are highly dependent on which middle layers on the student network and the teacher network it is connected to, which dramatically increases the hyperparameter search space. Let $n_{\mathcal{S}}$ and $n_{\mathcal{T}}$ denotes the number of layers on the student network and the teacher network, respectively, then the search space of the hyperparameters will be multiplied by $n_{\mathcal{S}} \times n_{\mathcal{T}}$, on the top of the additional hyperparameters search introduced by the knowledge distillation.

2.2.2 Review of Activation Boundary Loss

[20] propose a novel manner for the teacher network to provide privileged information to the student network – the activation boundaries. Their design is based on two statements: 1) a neural network expresses a complex function with a combination of activation boundaries [26], and 2) the decision boundary of a neural network is a combination of activation boundaries from its hidden layers [27]. Hence, they believe that the transfer of activation boundary information from the teacher network to the student network could dramatically improve the knowledge transfer performance in classification problems, since a classification problem depends on the formation of decision boundaries among classes.

Based on their inspiration, they took the approach to design a loss function that can transfer the activation boundaries of the teacher network to the student network. Instead of transferring the magnitudes of the neuron responses, their method aims to transfer the activation status of the neurons. A loss function can thus be constructed as follows,

$$\mathcal{L}_{AB}(\mathbf{x}) = \left\|\rho(\mathcal{N}_T(\mathbf{x})) - \rho(\mathcal{N}_S(\mathbf{x}))\right\|_1, \tag{2.3}$$

where $\rho(\cdot)$ is an element-wise activation indicator function that returns the activation status of the input neuron.

$$\rho(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

However, this loss function is not differentiable. Therefore, gradient cannot flow through this loss function during back propagation. To alleviate this problem, inspired by the hinge loss of the support vector machine [28], a feasible approximation of Equation 2.3, which is differentiable, can be defined as

$$\mathcal{L}_{AB}(\mathbf{x}) = \left\| \rho(\mathcal{N}_T(\mathbf{x})) \odot \sigma(\mu\mathbf{1} - \mathcal{N}_S(\mathbf{x})) + (\mathbf{1} - \rho(\mathcal{N}_T(x))) \odot \sigma(\mu\mathbf{1} - \mathcal{N}_S(\mathbf{x})) \right\|, \quad (2.4)$$

where $\odot$ denotes the element-wise product between two tensors and μ denotes the margin.

Although Activation Boundary loss is simple and intuitive, several limitations exist with its design. Firstly, the activation boundary can only be applied to classification problems, because it assumes a decision boundary of the neural network. Secondly, the generalization ability of the student network is not taken into consideration when designing the loss function.

2.2.3 Review of Flow of Solution Procedure Matrix

[29] propose to view the knowledge distillation from the teacher network to the student network as a way to transfer the decision process to obtain the solution. Since for a neural network, the intermediate feature maps can be regarded as part of the decision steps for the neural network to derive a solution, distilling the knowledge by the transfer of the intermediate feature maps is viable.

Based on this intuition, they propose to use encode the intermediate feature maps from the hidden layer along with their sequences into one single matrix – the Flow of Solution Procedure(FSP) Matrix. In their definition, let h, w, m denotes the height, width and number of channels of a layer, the FSP matrix $S \in \mathbb{R}^{m \times n}$ can be defined as

$$S_{i,j}(x;W) = \sum_{s=1}^{h} \sum_{t=1}^{w} \frac{F^1_{s,t,i}(x;W) \times F^2_{s,t,j}(x;W)}{h \times w}, \quad (2.5)$$

where F_1 and F_2 denote two selected feature maps, x denote the input image, and W denotes the weight of the neural network. During training, the distillation can be done by optimizing the network such that the FSP matrices of the teacher and the student networks are similar. Having selected n layers from both networks, the distillation can be done by minimising the following loss function

$$L_{FSP}(W_t, W_s) = \frac{1}{N}\sum_x \sum_{i=1}^{n} \lambda_i \times \left\|(S_i^T(x; W_t) - S_i^S(x; W_s)\right\|_2^2, \tag{2.6}$$

where λ_i represents the weighting of the loss from each selected layer i and N denotes the number of data points.

Despite the intuitive design, the FSP matrix suffers from several issues. Firstly, the parameter search can be expensive, since n layers need to be selected from both networks, and n additional parameters λ_i needs to be decided, hyperparameter tuning and training of deep networks where the number of layers is large can be expensive. Secondly, the caching of FSP matrices can be space-consuming.

Most knowledge distillation techniques assume that the entire training set used to train the teacher network will also be available for the student networks. This thesis will focus on this overlooked aspect of knowledge distillation techniques by focusing on the generalization ability of the student network, which will be presented in the next chapter.

CHAPTER 3

Methods

In this chapter, we first formulate the generalization ability of the student network given only a small amount of data on Section 3.1. Then, an upper bound will be derived as a result of our theoretical analysis on Section 3.2. The upper bound can then be developed into a novel loss function, also known as the Wasserstein Generalization loss, to train the student network, which will be presented on Section 3.3. On Section 3.4, we analyze the theoretical similarity between training the student network with the proposed loss function and with random perturbation. Finally, some practical tips during training will also be discussed.

In this thesis, we consider the general multi-class classification problem.

3.1 Rethinking the Generalization of the Student Network

By dint of knowledge distillation loss, the teacher network's softened outputs act as additional (privileged) information during the learning of student network and is shown to enhance generalization ability of the student network by improving the learning rate [30]. However, when the number of training examples is limited, the generalization error would be still fairly large, and severe over-fitting issues would happen consequently. To handle this problem, we suggest rethinking the generalization of the student network.

Denote the ground-truth data-generating distribution of instances $\boldsymbol{x} \in \mathcal{X}$ as $\mathbb{P}$, i.e. $\boldsymbol{x} \sim \mathbb{P}$. Then the aim of the learning student network is to minimize the following population risk,

$$\mathcal{R}(\mathcal{N}_S) = \mathbb{E}^{\mathbb{P}}[\![\ell(\boldsymbol{x}; \mathcal{N}_S, \mathcal{N}_T)]\!], \tag{3.1}$$

where $\mathbb{E}^{\mathbb{P}}$ is the expectation over the distribution $\mathbb{P}$. The term $\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T)$ is a loss encouraging the student network to match with the teacher network, for example, for knowledge distillation in Equation (2.1), $\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T)=\mathcal{H}(\boldsymbol{p}_i,\boldsymbol{q}_i)$. Given the dataset $\mathcal{D}^{\mathcal{X}}=\{\boldsymbol{x}_i\}_{i=1}^N$, the corresponding empirical risk goes to

$$\hat{\mathcal{R}}(\mathcal{N}_S)=\frac{1}{N}\sum_{i=1}^{N}\ell(\boldsymbol{x}_i;\mathcal{N}_S,\mathcal{N}_T). \tag{3.2}$$

In fact, Equation (3.2) can also be written into an expectation form. Define a discrete distribution $\mathbb{P}_N$ over $\mathcal{X}$ as

$$p(\boldsymbol{x})=\begin{cases}1/N, & \text{if } \boldsymbol{x}\in\{\boldsymbol{x}_i\}_{i=1}^N\\ 0, & \text{otherwise}\end{cases} \tag{3.3}$$

which is actually a uniform distribution over all N training data and called *discrete empirical distribution*. Then Equation (3.2) equals to

$$\hat{\mathcal{R}}(\mathcal{N}_S)=\mathbb{E}^{\mathbb{P}_N}[\![\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T)]\!]. \tag{3.4}$$

In this way, the gap between the empirical risk and the population risk (i.e. the generalization error) results from the difference between the ground-truth data-generating distribution $\mathbb{P}$ and the discrete empirical distribution $\mathbb{P}_N$. To shrink the gap, we estimate the $\mathbb{P}$ using $\mathbb{P}_N$, and assume $\mathbb{P}$ lies in a neighborhood of $\mathbb{P}_N$. Since $\mathbb{P}$ is continuous while $\mathbb{P}_N$ is discrete, to measure their distance and to cover every possible (continuous or discrete) distribution "between" them, we adopt the Wasserstein distance [31] as the measurement. The neighborhood $\mathcal{B}_\epsilon(\mathbb{P}_N)$ can be thus constructed as

$$\mathcal{B}_\epsilon(\mathbb{P}_N):=\{\mathbb{Q}\in\mathcal{M}(\mathcal{X}):d_W(\mathbb{P}_N,\mathbb{Q})\leq\epsilon\}, \tag{3.5}$$

where $\mathcal{M}(\mathcal{X})$ is the set of all possible distributions $\mathbb{Q}$ over $\mathcal{X}$, and $d_W(\cdot,\cdot)$ is the Wasserstein distance between two probabilistic distributions.

The rationale behind the Wasserstein distance over other measurements, such as KL-divergence or maximum mean discrepancy, is three-fold:

(1) it can lead to a tractable solution which will be presented in Section 3.2;

(2) it can measure the distance between discrete and continuous distributions;

(3) measure concentration results from [32] guarantee that $\mathcal{B}_\epsilon(\mathbb{P}_N)$ could contain the unknown ground-truth data distribution with high confidence.

Hence, Equation (3.5) can be viewed as a Wasserstein ball with a radius of ϵ that surrounds the distribution $\mathbb{P}_N$. Since the distribution $\mathbb{P}$ lies in this ball, to boost the generalization of the student network, we suggest to safely optimize the risk of all possible ground-truth distributions within the $\mathcal{B}_\epsilon(\mathbb{P}_N)$. This goal equals to improving the worst-case risk of all distributions in $\mathcal{B}_\epsilon(\mathbb{P}_N)$, which is equivalent to minimizing the supremum of the risks [33], i.e.

$$\sup_{\mathbb{Q}\in\mathcal{B}_\epsilon(\mathbb{P}_N)} \mathbb{E}^{\mathbb{Q}}[\![\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T)]\!]. \tag{3.6}$$

In this way, training with the objective Eq. (3.6) would alleviate the over-fitting problem caused by few training data, and boost the generalization and the classification performance of the student network accordingly.

3.2 Theoretical Analysis

As illustrated in the previous section, the generalization ability of the student network can be enhanced by optimizing Equation (3.6). Nevertheless, Equation (3.6) involves a supremum operation over the Wasserstein ball $\mathcal{B}_\epsilon(\mathbb{P}_N)$, which is not computationally tractable for the training networks in an end-to-end fashion. Inspired by the Majorization Minimization method [34], we choose to analyze the upper bound of Equation (3.6) as a surrogate objective, then we can optimize the upper bound instead.

[33, Theorem 6.3] presents an upper bound related to the empirical risk $\hat{\mathcal{R}}$.

THEOREM 1. *If ℓ is proper, convex and lower semicontinous, there exists an upper bound of Equation* (3.6) *for any $\epsilon \geq 0$, i.e.*

$$\sup_{\mathbb{Q}\in\mathcal{B}_\epsilon(\mathbb{P}_N)} \mathbb{E}^{\mathbb{Q}}[\![\ell(\boldsymbol{x})]\!] \leq \psi\epsilon + \frac{1}{N}\sum_{i=1}^{N}\ell(\boldsymbol{x}_i), \tag{3.7}$$

where $\psi := \sup\{\|\theta\|_* : \ell^*(\theta) < \infty\}$, ℓ^* *is the convex conjugate function of* ℓ *and* $\|\cdot\|_*$ *is the dual norm of* $\|\cdot\|$.

Here for ease of notation, throughout this section we suppress the dependence of the network $\mathcal{N}_S$ and $\mathcal{N}_T$ in loss $\ell(\boldsymbol{x}; \mathcal{N}_S, \mathcal{N}_T)$, and denote it as $\ell(\boldsymbol{x})$. And we adopt the ℓ_2 norm throughout this paper. Next, we focus on the estimation of ψ's value, which is dependent on the loss ℓ itself. We have the following result.

THEOREM 2. *If* ℓ *is* $\overline{L}$ *- lipschitz continuous on* $\mathcal{X}$*, i.e. for any* $\boldsymbol{x}$ *and* $\boldsymbol{x}' \in \mathcal{X}$,

$$\|\ell(\boldsymbol{x}) - \ell(\boldsymbol{x}')\| \leq \overline{L}\,\|\boldsymbol{x} - \boldsymbol{x}'\| \tag{3.8}$$

holds, then the ψ *in Equation* (3.7) *is upper bounded by*

$$\psi \leq \sup_{\boldsymbol{x} \in \mathcal{X}} \|\nabla_{\boldsymbol{x}} \ell(\boldsymbol{x})\|. \tag{3.9}$$

PROOF. [33, Proposition 6.5] shows that if ℓ is $\overline{L}$ - lipschitz continuous, then

$$\psi \leq \overline{L}. \tag{3.10}$$

Furthermore, the Lipschitz constant can be selected as the supremum module of the (sub) gradient over $\mathcal{X}$ [35], which completes the proof. □

By substituting Equation (3.9) in Equation (3.7), we can derive the following inequality

$$\sup_{\mathbb{Q} \in \mathcal{B}_\epsilon(\mathbb{P}_N)} \mathbb{E}^{\mathbb{Q}}[\![\ell(\boldsymbol{x})]\!] \leq \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{x}_i) + \psi\epsilon \tag{3.11}$$

$$\leq \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{x}_i) + \epsilon \sup_{\boldsymbol{x} \in \mathcal{X}} \|\nabla_{\boldsymbol{x}} \ell(\boldsymbol{x})\|. \tag{3.12}$$

However, calculating the supremum can be computationally expensive, so we just approximate it by using the training examples, i.e.

$$\frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{x}_i) + \epsilon \max_i \|\nabla_{\boldsymbol{x}_i} \ell(\boldsymbol{x}_i)\|. \tag{3.13}$$

Moreover, we adopt a proxy of this quantity as a regularizer following [36, 37] for more efficient training, then the upper bound can be relaxed into

$$\frac{1}{N}\sum_{i=1}^{N}\ell(\boldsymbol{x}_i) + \epsilon\frac{1}{N}\sum_{i=1}^{N}\left\|\nabla_{\boldsymbol{x}_i}\ell(\boldsymbol{x}_i)\right\|. \tag{3.14}$$

Recall that the parameter ϵ controls the radius of the Wasserstein ball centered at the discrete empirical distribution $\mathbb{P}_N$.

3.3 Wasserstein Generalization Loss

To meet the requirements of the loss ℓ in Theorems 1 and 2, in this thesis we adopt the following loss,

$$\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T) = \left\|\boldsymbol{o}^S - \boldsymbol{o}^T\right\|_2^2 := \ell(\boldsymbol{o}^S,\boldsymbol{o}^T), \tag{3.15}$$

where $\boldsymbol{o}^S \in \mathbb{R}^k$ and $\boldsymbol{o}^T \in \mathbb{R}^k$ are the input $\boldsymbol{x}$'s logits of the student and teacher network, respectively. Note that the loss ℓ for Equation (3.14) is not limited to Equation (3.15). We use Equation (3.15) for it serves a straightforward and easy way to measure the discrepancy between the output from student and the teacher network and works nicely in our experiment.

Combining the adopted loss Equation (3.15) and the upper bound Equation (3.14), we can obtain a loss promoting the generalization, i.e.

$$\mathcal{L}_W(\mathcal{N}_S) = \frac{1}{N}\sum_{i=1}^{N}\ell(\boldsymbol{o}_i^S,\boldsymbol{o}_i^T) + \epsilon\frac{1}{N}\sum_{i=1}^{N}\left\|\nabla_{\boldsymbol{x}_i}\ell(\boldsymbol{o}_i^S,\boldsymbol{o}_i^T)\right\|. \tag{3.16}$$

We call this novel loss Wasserstein Generalization loss, which is motivated by focusing on the generalization within the Wasserstein ball of the discrete empirical distribution. Since Wasserstein Generalization loss does not include supervision signal from the ground-truth labels, following the previous works [19, 20], our proposed Wasserstein Generalization loss should be used in conjunction with the KD loss as

$$\mathcal{L}_{train}(\mathcal{N}_S) = \mathcal{L}_{KD}(\mathcal{N}_S) + \alpha\mathcal{L}_W(\mathcal{N}_S), \tag{3.17}$$

where α is a hyperparameter balancing the two losses. In this way, the student network can be better guided by the teacher network through distilling its knowledge as well as boosting own generalization.

The gradient term included in Equation (3.16) can be interpreted as a term that improves the generalization ability when being optimized. Intuitively, the gradient term $\nabla_{\boldsymbol{x}_i}\ell(\boldsymbol{o}_i^S, \boldsymbol{o}_i^T)$ can be viewed as the sensitivity of the network with respect to the input [38]. For example, if one single image has a large gradient during training, that means the image needs to be paid attention to. By reducing this term, we are effectively reducing the attention of the network to one single image specifically. In other words, optimizing gradient term can smoothen the decision boundaries trained on $\mathbb{P}_N$, and thus reduce the gap between network output for input data in $\mathbb{P}_N$ and $\mathbb{Q}$.

3.4 Optimizing Gradient As Data Augmentation

Training deep neural networks requires a large amount of data. However, in the case of limited data availability, a simple solution is to perform data augmentation. Image data augmentation involves horizontal or vertical flipping of images and adding noise to the images. We next proceed to show that the inclusion of the gradient term in Equation (3.14) can be explained as training the network with this data augmentation mechanism.

THEOREM 3. *Consider* $\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T) = \left\|\boldsymbol{o}^S - \boldsymbol{o}^T\right\|_2^2 := \ell(\boldsymbol{o}^S, \boldsymbol{o}^T)$ *as the loss function to train the student network. Given a random perturbation* $\boldsymbol{\gamma} \sim (\mathbf{0}, v\mathbf{I})$ *on the input data* $\boldsymbol{x}$*, then the objective on the noisy input to train the student network would be approximated as*

$$\begin{aligned}
&\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{\gamma})}[\![\ell(\boldsymbol{x}+\boldsymbol{\gamma};\mathcal{N}_S,\mathcal{N}_T)]\!] \\
&= \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T)]\!] + v \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\|\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\|_2^2]\!] + O(v^2) \\
&:= \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\tilde{\ell}(\boldsymbol{x};\mathcal{N}_T,\mathcal{N}_S)]\!] + O(v^2).
\end{aligned} \tag{3.18}$$

PROOF. Given a teacher network $\mathcal{N}_T$, student network $\mathcal{N}_S$ and their respective output $\boldsymbol{o}_T$ and $\boldsymbol{o}_S$, assuming a random perturbation $\gamma \sim (0, v\mathbf{I})$ is added to the training data, the loss function ℓ then becomes

$$\begin{aligned}
&\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{\gamma})}[\![\ell(\boldsymbol{x}+\boldsymbol{\gamma};\mathcal{N}_S,\mathcal{N}_T)]\!] \\
&= \mathop{\mathbb{E}}_{p(\boldsymbol{x},\gamma)}[\![(\mathcal{N}_S(\boldsymbol{x}+\gamma)-\boldsymbol{o}_T)^2]\!] \\
&= \mathop{\mathbb{E}}_{p(\boldsymbol{x},\gamma)}[\![\mathcal{N}_S^2(\boldsymbol{x}+\gamma)-2\boldsymbol{o}_T\mathcal{N}_S^2(\boldsymbol{x}+\gamma)+\boldsymbol{o}_T^2]\!],
\end{aligned} \tag{3.19}$$

where $p(\boldsymbol{x},\boldsymbol{\gamma})$ denotes the probability distribution of x and $\boldsymbol{\gamma}$. Assuming that the noise γ is small, $\mathcal{N}_S(\boldsymbol{x}+\gamma)$ can be approximated using Taylor series expansion of $\mathcal{N}_S(\boldsymbol{x}+\gamma)$ around $\mathcal{N}_S(\boldsymbol{x})$.

$$\mathcal{N}_S(\boldsymbol{x}+\gamma)=\mathcal{N}_S(\boldsymbol{x})+\gamma^\top\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})+\frac{1}{2}\gamma^\top\nabla^2_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\gamma+O(\gamma^3) \tag{3.20}$$

Substituting the approximation in Equation (3.19), $\tilde{\ell}$ can then be approximated as the following

$$\begin{aligned}
&\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{\gamma})}[\![\ell(\boldsymbol{x}+\boldsymbol{\gamma};\mathcal{N}_S,\mathcal{N}_T)]\!] \\
&\approx \mathop{\mathbb{E}}_{p(\boldsymbol{x},\gamma)}[\![(\mathcal{N}_S(\boldsymbol{x})+\gamma^\top\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})+\frac{1}{2}\gamma^\top\nabla^2_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\gamma)^2]\!] \\
&\quad -2\mathop{\mathbb{E}}_{p(\boldsymbol{x},\gamma)}[\![\boldsymbol{o}_T\mathcal{N}_S(\boldsymbol{x})\boldsymbol{o}_T\gamma^\top\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})+\frac{1}{2}\boldsymbol{o}_T\gamma^\top\nabla^2_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\gamma]\!] \\
&= \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![(\mathcal{N}_S(\boldsymbol{x})-\boldsymbol{o}_T)^2]\!]-2\mathop{\mathbb{E}}_{p(\boldsymbol{x},\gamma)}[\![\frac{1}{2}\boldsymbol{o}_T\gamma^\top\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\gamma]\!] \\
&\quad +\mathop{\mathbb{E}}_{p(\boldsymbol{x},\gamma)}[\![\mathcal{N}_S(\boldsymbol{x})\gamma^\top\nabla^2_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\gamma+(\gamma^\top\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x}))^2+O(\gamma^3)]\!] \\
&= \ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T)+v\mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![(\mathcal{N}_S(\boldsymbol{x})-\boldsymbol{o}_T)\nabla^2_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})]\!]+v\mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\|\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\|_2^2]\!].
\end{aligned}$$

If this loss function is minimized by taking the functional gradient of $\mathcal{N}_S(\boldsymbol{x})$ and setting the result to zero, then

$$\mathcal{N}_S(\boldsymbol{x})=\mathop{\mathbb{E}}_{p(\boldsymbol{o}_T,\boldsymbol{x})}[\![\boldsymbol{o}_T]\!]+O(v)$$

which indicates that

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![(\mathcal{N}_S(\boldsymbol{x})-\boldsymbol{o}_T)\nabla^2_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})]\!]$$

reduces to $O(v)$. Hence, Equation (3.19) becomes

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{\gamma})}[\![\ell(\boldsymbol{x}+\boldsymbol{\gamma};\mathcal{N}_S,\mathcal{N}_T)]\!] = \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T)]\!] + O(v^2) + v\mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\|\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\|_2^2]\!], \quad (3.21)$$

which we use $\tilde{\ell}$ to denote,

$$\begin{aligned} &\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{\gamma})}[\![\ell(\boldsymbol{x}+\boldsymbol{\gamma};\mathcal{N}_S,\mathcal{N}_T)]\!] \\ &= \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\ell(\boldsymbol{x};\mathcal{N}_S,\mathcal{N}_T) + v\,\|\nabla_{\boldsymbol{x}}\mathcal{N}_S(\boldsymbol{x})\|_2^2]\!] + O(v^2) \\ &:= \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\![\tilde{\ell}(\boldsymbol{x};\mathcal{N}_T,\mathcal{N}_S)]\!] + O(v^2). \end{aligned}$$

$\square$

This loss function is similar to Equation (3.14) in many aspects. Firstly, their first term is identical. Secondly, the second term is the gradient with respect to the input. Optimizing Equation (3.14) also optimizes Equation (3.18). Since the first terms of Equation (3.18) and $\tilde{\ell}$ are equal, the second terms can be seen as a regularization term that penalizes the large gradient value of $\mathcal{N}_S(\boldsymbol{x})$. The two regularization terms can also be treated as ℓ_1 and ℓ_1 regularizations with respect to the gradient. Similar to the feature of L1 regularization, the regularization term of Equation (3.14) is robust to large gradients during training. Therefore, optimizing Equation (3.14) with $\mathbb{P}_N$ resembles training with $\mathbb{P}_N$ with random perturbation, which is a popular method to perform data augmentation. The performance of the student network trained with few data can then be guaranteed by emulating the data augmentation.

Algorithm 2 Training the Student Network with Wasserstein Generalization loss

Input: A teacher network $\mathcal{N}_T$; a student network $\mathcal{N}_S$; training data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$; balancing factor α; number of epochs e; batch size b.
Output: A trained student network $\mathcal{N}_S$.

```
 1: Let $i = 0$.
 2: while until converge or $i \leq e$ do
 3:     Sample a minibatch from training data.
 4:     Calculate the logits $\boldsymbol{o}^S$ and $\boldsymbol{o}^T$
 5:     Calculate $\ell(\boldsymbol{o}^S, \boldsymbol{o}^T)$ as Equation (3.15).
 6:     Calculate $\mathcal{L}_W(\ell(\boldsymbol{o}^S, \boldsymbol{o}^T))$ as Equation (3.16).
 7:     Calculate $\mathcal{L}_{KD}$ as Equation (2.1).
 8:     Apply back propagation and update weights of $\mathcal{N}_S$ accordingly based on Equation (3.17).
 9: end while
10: return $\mathcal{N}_S$
```

CHAPTER 4

Experiment

In this chapter, we conduct experiments to validate our theoretical analysis and demonstrate the ability of the proposed Wasserstein Generalization loss to boost the generalization ability of the student network. We will begin by first discussing the datasets used and the configuration of the networks in our experiment in Section 4.1. Next, we proceed to design the method to evaluate the generalization ability of neural networks in Section 4.2. Experimental results in comparison with the baseline models will be presented in Section 4.3. Then, we discuss the structural complexity of our network configurations and their impact on the experimental results in Section 4.4. Finally, we discuss the impact of the hyperparameters selections and the effect of proxy in Section 4.5 and Section 4.6, respectively.

4.1 Datasets and Network Configuration

4.1.1 CIFAR-10 and CIFAR-100 dataset

CIFAR-10 and CIFAR-100 dataset [39] consist of 60,000 tiny RGB images with shape 32×32, where 50,000 of the images are training set and the remaining 10,000 images are intended for testing. The tiny images in CIFAR-10 are split into 10 mutually exclusive categories, which are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. For the two datasets, we use the same teacher and student network structures, while CIFAR-100 is split into 100 mutually exclusive categories. The architecture of the teacher network for the two datasets consists of three convolutional maxout [40] layers followed by a fully connected maxout layer in a 96-192-192-500 plus a softmax layer configuration. The design of the teacher

TABLE 4.1: Details of the network structure used in the experiments using CIFAR dataset. “conv" means the convolutional layers and “FC" stands for the fully-connected layers.

CIFAR Teacher	CIFAR-10 Student #0	CIFAR-10 Student #1	CIFAR-10 Student #2	CIFAR-100 Student
conv 8x8x96 maxpool 2x2 dropout	conv 3x3x16 conv 3x3x32 conv 3x3x32 maxpool 2x2	conv 3x3x16 conv 3x3x16 conv 3x3x16 maxpool 2x2	conv 3x3x32 conv 3x3x48 conv 3x3x64 conv 3x3x64 maxpool 2x2	conv 3x3x32 conv 3x3x48 conv 3x3x64 conv 3x3x64 maxpool 2x2 dropout
conv 8x8x192 maxpool 2x2 dropout	conv 3x3x48 conv 3x3x64 conv 3x3x80 maxpool 2x2	conv 3x3x32 conv 3x3x32 conv 3x3x32 maxpool 2x2	conv 3x3x80 conv 3x3x80 conv 3x3x80 conv 3x3x80 maxpool 2x2	conv 3x3x80 conv 3x3x80 conv 3x3x80 conv 3x3x80 maxpool 2x2 dropout
conv 5x5x192 maxpool 2x2	conv 3x3x96 conv 3x3x96 conv 3x3x128 maxpool 2x2	conv 3x3x48 conv 3x3x48 conv 3x3x64 maxpool 8x8	conv 3x3x128 conv 3x3x128 conv 3x3x128 conv 3x3x128 maxpool 8x8	conv 3x3x80 conv 3x3x80 conv 3x3x80 conv 3x3x80 maxpool 2x2 dropout
maxout 500 FC+softmax	maxout 500 FC+softmax	maxout 500 FC+softmax	maxout 500 FC+softmax	maxout 500 FC+softmax

network generally follows the architecture used in FitNet [19] and maxout with some minor modifications (See Table 4.1). The images are rescaled to range $[0,1]$ and are augmented by random cropping with paddings and random horizontal flipping, before feeding into the network. The mean and standard deviation of the images are rescaled to zero and one across three channels.

4.1.2 Fashion-MNIST dataset

Fashion-MNIST dataset [41] consists of 28×28 greyscale images from ten different categories of fashion items, including T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. The number of examples for training and testing is 60,000 and 10,000, respectively. The architecture of the teacher network consists of two convolutional layers

TABLE 4.2: Details of the network structure used in the experiments using Fashion-MNIST dataset. “conv" means the convolutional layers and “FC" stands for the fully-connected layers.

Fashion-MNIST Teacher	Fashion-MNIST Student
conv 8x8x32	conv 3x3x32
maxpool 2x2	conv 3x3x32
	conv 3x3x32
	conv 3x3x32
	maxpool 2x2
conv 8x8x64	conv 3x3x64
maxpool 2x2	conv 3x3x64
	conv 3x3x64
	conv 3x3x64
	maxpool 2x2
FC 4096	FC 500
FC+softmax	FC+softmax

with kernel size 8×8 followed by a fully connected layer with 4096 units and a softmax layer that predicts the probability distribution over the ten categories. Before images are fed into the network, the images are rescaled to range $[0, 1]$ and are augmented by random horizontal flipping only. Fashion-MNIST is favored over MNIST for its complexity and its more accurate presentation of modern computer vision tasks.

4.2 Evaluation of Generalization Ability

To demonstrate the advantage of the proposed algorithm in generalization, we investigate the classification performance when the student network is trained with a different number of examples. The teacher network is pre-trained with complete training examples. As for the student network, we randomly sample M examples for each of the categories and use them for training. Then different M values indicate the different size of the training set for the student network. During sampling, balance with respect to the number of examples across different categories is strictly maintained to avoid unnecessary performance impact.

We compare the classification accuracy of the student network when it is trained with different competing losses, including the original knowledge distillation (KD) loss, FitNet (FN) [19], Activation Boundary (AB) loss [20] as well as our Wasserstein Generalization (WG) loss. As mentioned in the previous section, following the related works, the Wasserstein Generalization loss is designed to be used on the top of KD loss. For the fair comparison, all methods use the same teacher network, and all student networks have an identical structure. In specific, the pretrained teacher networks have testing accuracy of 89%, 47% and 92% on the CIFAR-10, CIFAR-100 and Fashion-MNIST dataset, respectively. In our experiments, ϵ is set to 0.01 and α is set to 0.001 unless stated otherwise. Temperature T for KD loss is set to 3 and the learning rate is set to 0.001. On all three datasets, the student networks are trained using backpropagation and Stochastic Gradient Descent (SGD) with momentum for 500 epochs. During training, the learning rate and the momentum decay linearly. The experiments are run on a single NVIDIA GeForce 1080 Ti GPU.

4.3 Experimental Results

TABLE 4.3: Classification accuracy with respect to various methods and different number of training examples on CIFAR-10 dataset.

M	Ours	AB [20]	FN [19]	KD [17]
Full	**84.66%**	82.65%	77.84%	79.72%
1000	**82.01%**	76.19%	63.84%	77.63%
500	**73.08%**	66.69%	54.51%	50.63%
100	**47.24%**	44.7%	29.68%	31.83%
50	**41.16%**	38.23%	31.65%	28.32%

TABLE 4.4: Classification accuracy with respect to various methods and different number of training examples on Fashion-MNIST dataset.

M	Ours	AB [20]	FN [19]	KD [17]
Full	**94.72%**	92.1%	92.23%	92.1%
1000	**91.95%**	91.67%	91.53%	90.44%
500	**90.46%**	90.25%	82.6%	89.45%
100	**85.18%**	84.09%	81.92%	83.77%
50	**81.73%**	80.82%	81.40%	78.5%

Table 4.3 describes the performance of each technique using the CIFAR-10 dataset. The simplest model, Student #0, is used for all experiments in this table. As can be shown in Table 4.3, from the results obtained from the CIFAR-10 dataset, due to the optimization with

TABLE 4.5: Classification accuracy with respect to various methods and different number of training examples on CIFAR-100 dataset.

M	Ours	AB [20]	FN [19]	KD [17]
Full	**49.64%**	48.22%	48.94%	48.74%
250	34.98%	34.28%	**35.38%**	34.7%
100	**21.81%**	20.68%	20.37%	21.76%
50	**20.32%**	19.22%	18.64%	19.27%

respect to generalization ability, models trained with Wasserstein Generalization loss outperform in comparison with other techniques. Specifically, when the models are trained with the full training set, the testing accuracy of the model trained with Wasserstein Generalization loss outperforms AB by around 2 percentage points and outperforms other training techniques by a larger margin. When the number of training samples reduces, Wasserstein Generalization loss can still maintain its ability to generalize by outperforming all other baseline techniques. For example, Wasserstein Generalization loss outperforms the second-best baseline method by 4.25 points on KD, 6.39, 2.54 and 2.93 on AB when M equals to 1000, 500, 100 and 50, respectively.

Similar observations can be found from Table 4.4, which shows the testing accuracy of the models using the Fashion-MNIST dataset. When being trained on the full training set, the performance of all tested models is superior to the teacher networks, while Wasserstein Generalization loss achieves the highest test set accuracy of 94.72%. As M decreases, the accuracy of Wasserstein Generalization loss drops from 94.72% to 81.73%, while AB loss drops from 92.1% to 80.82%, FitNet drops from 92.23% to 81.4% and vanilla KD drops from 92.1% to 78.5%. The test set accuracy of the model trained with Wasserstein Generalization loss is higher than AB loss, FitNet and KD loss at all tested values of M.

In comparison with CIFAR-10 and Fashion-MNIST, CIFAR-100 can be difficult to train because there are only 600 images for each class in the dataset. In this case, a loss function that boosts the generalization ability of the student network will be more essential than other training algorithms or techniques. As demonstrated in Table 4.3, despite having only 600 data samples in each category, the network trained with Wasserstein Generalization loss is able to outperform all other baselines.

4.4 Structural Complexity Analysis of Student Networks

TABLE 4.6: Classification accuracy of student networks on CIFAR-10 trained by various methods using different number of samples, where AB, FN, KD denotes Activation Boundary loss [20], FitNet [19] and Knowledge Distillation [17] loss, respectively.

	Student #1				Student #2			
M	Ours	AB	FN	KD	Ours	AB	FN	KD
1000	78.37%	73.46%	64.83%	62.88%	84.23%	78.64%	62.84%	62.46%
500	67.56%	62.88%	52.71%	48.12%	75.45%	68.82%	49.12%	53.75%
100	43.96%	41.54%	30.2%	28.52%	49.69%	42.95%	42.30%	36.55%
50	41.41%	35.49%	29.0%	35.52%	37.67%	36.45%	29.89%	30.73%

TABLE 4.7: Structural complexity statistics of the teacher network and various student networks on CIFAR dataset. The compression and acceleration rate is with respect to the teacher network.

	#Params	#FLOPs	Compression Ratio	Acceleration Ratio
Teacher	11M	59.62M	100%	100%
Student #0	1M	5.27M	8.87%	8.83%
Student #1	0.3M	1.73M	2.92%	2.90%
Student #2	2M	9.95M	16.72%	16.69%

To examine the generalization ability benefit of the Wasserstein Generalization loss from the perspective of network structure, we conducted experiments with the same setting as described in Section 4.1, but with different network complexities. In this experiment, we investigate the change of performance affected by altering the network complexity. Concretely, we trained Student #1 and #2 using the techniques mentioned in the previous section, and compare the difference in terms of testing accuracy. 4.7 shows the general statistics of the architectures used in the CIFAR dataset. In this table, the compression ratio of the student networks is computed by the ratio of the number of parameters of the student network to the teacher network, assuming the parameters are single precision floating point numbers. The acceleration ratio is computed by the ratio of the number of floating point operations of the student network to the teacher network. We can observe from this table that the number of parameters in Student #1 is smaller than that of Student #0 while the number of parameters in Student #2 is larger.

Table 4.6 shows the testing accuracy achieved by various training techniques on Student #1 and Student #2. As can be seen from Table 4.6, the performance of Wasserstein Generalization loss remains on the top of other techniques on all listed M values, despite the complexity of the network structure is increased or decreased. On $M = 1000$, Student #2 trained by Wasserstein Generalization loss obtained 84.23%, despite the number of parameters being 16.72% of the teacher network. Meanwhile, the testing accuracy obtained by other techniques on Student #2 is lower than their corresponding results under the same M in Table 4.6 by a relatively large margin, which demonstrates the superiority of the proposed Wasserstein Generalization loss in terms of generalization.

4.5 Hyperparameter Analysis

TABLE 4.8: Classification accuracy of the student network #0 on CIFAR-10 trained by Wasserstein Generalization loss with respect to different hyperparameters α and ϵ in log scale.

	$log_{10}\alpha$				
$log_{10}\epsilon$	-1	-2	-3	-4	-5
0	63.16%	80.33%	80.65%	77.67%	77.63%
-1	63.55%	80.03%	79.81%	78.14%	78.49%
-2	62.62%	80.50%	79.27%	78.95%	77.77%
-3	62.54%	80.12%	79.48%	78.81%	78.21%
-4	62.64%	80.07%	78.78%	78.58%	74.90%
-5	63.05%	79.92%	78.81%	78.52%	78.10%

In this subsection, we perform a hyperparameter analysis of our proposed Wasserstein Generalization loss. To this end, the CIFAR-10 model is trained using $M = 1000$. Other parameters, such as the temperature of the distillation, are heavily discussed in previous work [17]. Hence, we focus on two important hyperparameters that are included in the proposed Wasserstein Generalization loss, which are the weight of Wasserstein Generalization loss α and ϵ. Table 4.8 shows the result of the experiments for all combinations of α from 1 to 10^{-5} with increment 0.1, and ϵ for the same range with the same increment using 1000 samples. In theory, the optimal radius of the Wasserstein ball ϵ can be computed with given confidence value within the range $(0, 1)$ using Equation 8 in [33, Theorem 3.4]. However, in our experiments, with a fixed value of α, the performance of the models shows a low correlation with the generalization ability of the network. As α decreases, the performance of the network drops dramatically as the importance of Wasserstein Generalization loss diminishes. When α is larger than 0.001, the performance of the network drops dramatically, which is due to the inclusion of the gradient term. This experiment indicates that the effect of Wasserstein Generalization loss is only sensitive to α, which controls the relative weight of Wasserstein Generalization loss itself, in comparison with other losses (i.e. KD loss).

4.6 Proxy Analysis

In Section 3.2, we have derived an upper bound to the formulation of the generalization ability of the student network, i.e. Equation 3.6. This upper bound is then relaxed into a tractable solution in Equation 3.11. Since calculating the supremum over the ground-truth data-generating distribution $\boldsymbol{x} \in \mathcal{X}$ can be computationally expensive, this supremum term is then calculated over the current training batch in practice, as shown in Equation 3.13. To further improve the stability and efficiency of the training process, a proxy of this the supremum term is adopted.

In this section, we perform experiments to verify whether adopting such proxy can improve the efficiency and stability of the training process. The experiments will be conducted by comparing the performance between the Student #0 trained with Wasserstein Generalization loss as shown in Equation 3.14 and one trained with the un-proxied version of the loss, i.e. Equation 3.13, which can be implemented as

$$\mathcal{L}'_W(\mathcal{N}_S) = \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{o}_i^S, \boldsymbol{o}_i^T) + \epsilon \max_i \left\| \nabla_{\boldsymbol{x}_i} \ell(\boldsymbol{o}_i^S, \boldsymbol{o}_i^T) \right\| .$$

In this experiment, $\mathcal{L}'_W$ will be used in addition to the Knowledge Distillation loss, which is identical to how $\mathcal{L}_W$ is used in practice as shown in Equation 3.17. The experiments will be run with CIFAR-10 dataset on Student #0 with the same configuration as Section 4.4. For both networks, we set ϵ to 10^{-2}, α to 10^{-3}, and the learning rate to 10^{-3}.

TABLE 4.9: Classification accuracy of the student network #0 on CIFAR-10 and Fashion-MNIST trained by Wasserstein Generalization loss (proxied) and the unproxied version of the Wasserstein Generalization loss (unproxied).

	Dataset			
	CIFAR-10		Fashion-MNIST	
M	Proxied	Unproxied	Proxied	Unproxied
Full	84.66%	82.77%	94.72%	93.19%
1000	82.01%	79.69%	91.95%	90.21%
500	73.08%	70.41%	90.46%	88.30%
100	47.24%	43.22%	85.18%	82.43%
50	41.16%	35.21%	81.73%	80.35%

Table 4.9 shows the accuracy of the unproxied loss compared against the proxied loss (referred to as Wasserstein Generalization loss) across various M values on the CIFAR-10 and Fashion-MNIST datasets. As can be seen from this table, Wasserstein Generalization loss outperforms the unproxied version across all M values. In general, the performance of the student network trained by the unproxied loss deteriorates faster than the proxied version when M decreases. In some circumstances, for example, when M is 500 on Fashion-MNIST dataset, it performs even worse than some other baseline methods shown in Tables 4.3 and 4.4. The fast deterioration of the unproxied loss when the number of training samples decreases indicates the necessity of adopting the proxied version because 1) it stabilises the training process by avoiding exceptionally large values returned by the loss function; and 2) it emulates the data augmentation by adding random perturbation in the training set, as discussed in Section 3.4.

CHAPTER 5

Conclusions

5.1 Thesis Summary

This thesis proposed an approach to enhance the generalization ability of the student network when the training data is limited.

Chapter 1 briefly introduces the background knowledge and the motivation of the knowledge distillation techniques, as well as the points overlooked by current studies.

Chapter 2 discusses one of the main backbone of our approach – Wasserstein distance, as well as some previous work on knowledge distillation.

In Chapter 3 we present our proposed approach. Concretely, we assume that the ground-truth data-generating distribution lies in a Wasserstein ball centered on the training examples' discrete empirical distribution. Thus we can safely optimize the risks of all possible distributions within this ball to boost the generalization ability. Furthermore, for ease of training networks in an end-to-end fashion, we theoretically relax the upper bound of the supremum risk and develop a novel loss called Wasserstein Generalization loss accordingly. In particular, the proposed Wasserstein Generalization loss is also easy to implement for the networks in real applications.

In Chapter 4, we perform extensive experiments on benchmark datasets, which result in empirical results that validate the effectiveness of our proposed method. It has been shown that our Wasserstein Generalization loss is capable of improving the classification performance of the student network, and has significant superiority over other competing methods when the training data is very limited.

5.2 Future Works

While theoretical and empirical results demonstrate that Wasserstein Generalization loss has indeed improved the generalization ability of the student network in the knowledge distillation setting, the introduction of the Wasserstein metric in the optimization constraints indicates several possible further research and extension. In addition, the idea of improving the generalization of the student network indicates its possible application on generative models.

Bibliography

[1] S. Kong, T. Guo, S. You, and C. Xu, "Learning student networks with few data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

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

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] C. Wang, M. Li, and A. J. Smola, "Language models with transformers," *CoRR*, vol. abs/1904.09408, 2019.

[9] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[10] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International Conference on Machine Learning*, 2015, pp. 2285–2294.

[11] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[12] Y. Tang, S. You, C. Xu, B. Shi, and C. Xu, "Bringing Giant Neural Networks Down to Earth with Unlabeled Data," *arXiv:1907.06065 [cs, stat]*, Jul. 2019.

[13] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.

[14] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[16] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in neural information processing systems*, 2014, pp. 2654–2662.

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

[18] S. You, C. Xu, C. Xu, and D. Tao, "Learning from Multiple Teacher Networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 1285–1294.

[19] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *CoRR*, vol. abs/1412.6550, 2015.

[20] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3779–3787.

[21] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[22] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein Distance Guided Representation Learning for Domain Adaptation," Jul. 2017.

[23] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Jan. 2017.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[25] C. Frogner, C. Zhang, H. Mobahi, M. Araya, and T. A. Poggio, "Learning with a wasserstein loss," in *Advances in Neural Information Processing Systems*, 2015, pp. 2053–2061.

[26] R. Pascanu, G. Montufar, and Y. Bengio, "On the number of response regions of deep feed forward networks with piece-wise linear activations," *arXiv preprint arXiv:1312.6098*, 2013.

[27] X. Pan and V. Srikumar, "Expressiveness of rectifier networks," in *International Conference on Machine Learning*, 2016, pp. 2427–2435.

[28] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[29] J. Yim, D. Joo, J. Bae, and J. Kim, "A Gift From Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning," 2017, pp. 4133–4141. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Yim_A_Gift_From_CVPR_2017_paper.html

[30] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," *CoRR*, vol. abs/1511.03643, 2015.

[31] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[32] N. Fournier and A. Guillin, "On the rate of convergence in wasserstein distance of the empirical measure," *Probability Theory and Related Fields*, vol. 162, no. 3-4, pp. 707–738, 2015.

[33] P. Mohajerin Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations," *Mathematical Programming*, vol. 171, no. 1, pp. 115–166, Sep 2018. [Online]. Available: https://doi.org/10.1007/s10107-017-1172-1

[34] D. R. Hunter and K. Lange, "A tutorial on mm algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004.

[35] D. P. Bertsekas, *Convex optimization theory*. Athena Scientific Belmont, 2009.

[36] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[37] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems*, 2017, pp. 2266–2276.

[38] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[39] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[40] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[41] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.