Department of Computer Science

Algorithms for Data-Efficient Training of Deep Neural Networks

Vikas Verma

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 198/2020

Algorithms for Data-Efficient Training of Deep Neural Networks

Vikas Verma

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on 16 December 2020 at 12:00.

Aalto University
School of Science
Department of Computer Science
Deep Learning

Supervising professor
Prof. Juho Kannala

Thesis advisors
Prof. Yoshua Bengio, Mila - Quebec AI Institute (Mila - Institut québécois d'intelligence artificielle), Canada

Prof. Tapani Raiko, Aalto University, Finland

Prof. Juha Karhunen, Aalto University, Finland

Preliminary examiners
Prof. José Miguel Hernández-Lobato, University of Cambridge, United Kingdom

Prof. Ben Glocker, Imperial Colloge of London, United Kingdom

Opponent
Prof. José Miguel Hernández-Lobato, University of Cambridge, United Kingdom

Aalto University publication series
DOCTORAL DISSERTATIONS 198/2020

ISBN 978-952-64-0159-1 (printed)
ISBN 978-952-64-0160-7 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-64-0160-7

Unigrafia Oy
Helsinki 2020

Finland

NORDIC SWAN ECOLABEL
Printed matter
4041-0619

Author
Vikas Verma

Name of the doctoral dissertation
Algorithms for Data-Efficient Training of Deep Neural Networks

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 198/2020

Field of research Deep Neural Networks, Machine Learning, Artificial Intelligence

Manuscript submitted 21 April 2020 **Date of the defence** 16 December 2020

Permission for public defence granted (date) 9 November 2020 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Deep Neural Networks ("deep learning") have become a ubiquitous choice of algorithms for Machine Learning applications. These systems often achieve human-level or even super-human level performances across a variety of tasks such as computer vision, natural language processing, speech recognition, reinforcement learning, generative modeling and healthcare. This success can be attributed to their ability to learn complex representations directly from the raw input data, completely eliminating the hand-crafted feature extraction from the pipeline. However, there still exists a caveat: due to the extremely large number of trainable parameters in Deep Neural Networks, their generalization ability depends heavily on the availability of a large amount of labeled data.

In many machine learning applications, gathering a large amount of labeled data is not feasible due to privacy, cost, time or expertise constraints. Examples of such applications are abundant in healthcare; for example, predicting the effect of a medicine on a new patient in the scenario where the medicine has been administered to only a few patients earlier. This thesis addresses the problem of improving the generalization ability of Deep Neural Networks using a limited amount of labeled data. More specifically, this thesis explores a class of methods that directly incorporates the inductive bias about how the Deep Neural Networks should "behave" in-between the training samples (both in the input space as well as the hidden space) into the learning algorithms. Throughout several publications included in this thesis, the author has demonstrated that such kinds of methods can outperform conventional baseline methods and achieve state-of-the-art performance across supervised, unsupervised, semi-supervised, adversarial training and graph-based learning settings.

In addition to these algorithms, the author proposes a mutual information based method for learning the representations for the "graph-level" tasks in an unsupervised and semi-supervised manner. Finally, the author proposes a method to improve the generalization of ResNets based on the iterative inference view.

Keywords

ISBN (printed) 978-952-64-0159-1 **ISBN (pdf)** 978-952-64-0160-7

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2020

Pages 222 **urn** http://urn.fi/URN:ISBN:978-952-64-0160-7

Preface

This dissertation summarizes the work I have carried out as a doctoral student at the Department of Computer Science, School of Science, Aalto University under the supervision of Prof. Juho Kannala, Prof. Tapani Raiko, Prof. Juha Karhunen and Prof. Yoshua Bengio (University of Montreal, Canada), at various times between January 2016 and March 2020. None of these had been possible without enormous support and help from my supervisors, collaborators and all the wonderful people I have met during these years. Although I cannot express my gratitude fully in words, let me try: Thank you!

In January 2016, I started my doctoral studies as part of a group on Deep Learning and Bayesian Modeling under the supervision of Prof Tapani Raiko. I had the pleasure to meet my wonderful colleagues in this group: (in a random order) Pyry Takala, Vikram Kamath, Muddassar Abbas and Yao Lu. Outside of my own research group, I was fortunate to discuss research ideas with Dr. Harri Valpola (Curious AI), Prof. Harri Lähdesmäki, Prof. Mikko Kurimo, Prof. Alexander Ilin (again, in a random order) Thank you all!

After Prof. Tapani Raiko moved to an industrial job in the mid 2016, Prof. Juha Karhunen kindly accepted me under his supervision. I thank Prof. Juha Karhunen for advising me during the early days of my doctoral studies and attending to some of the administrative matters related to my doctoral dissertation. In early 2017, I was granted a HICT scholarship to visit Montreal Institute of Learning Algorithms (MILA) for working under the supervision of Prof Yoshua Bengio for six months. Those six months allowed me to explore various topics in Deep Neural Networks, which became the foundation of several of my later publications.

After Prof. Juha Karhunen retired in mid 2017, I was kindly accepted by my current supervisor Prof. Juho Kannala. I am deeply grateful to Prof. Juho Kannala for giving me the freedom to explore the research directions of my interest, for advising on many of the research papers included in this dissertation, as well as taking care of many of the administrative matters related to my doctoral studies. It has been a pleasure to work under the

supervision of Prof. Juho Kannala. Kittos Prof. Juho!

I spent most of 2018 and 2019 at MILA working under the supervision of Prof. Yoshua Bengio. Those two years were perhaps the most important years for me to grow as a researcher. Prof. Yoshua Bengio's advice on my various projects has taught me how to select research problems which could potentially have larger research as well as societal impact, and how to conduct research of the highest standards. Merci beaucoup Prof. Yoshua!

At MILA, I was fortunate to meet a number of researchers, collaborators and friends. Although I would like to list all of these people, I can only list a few: Devansh Arpit, Akhilesh Badrinarayan, Christopher Beckham, Prof. Aaron Courville, Mojtaba Faramarzi, Simon Guiroy, Devon Hjelm, Sina Honari, Jason Jo, Alex Lamb, Prof. Ioannis Mitliagkas, Vardaan Pahuja, Prof. Christopher Pal, Meng Qu, Sai Rajeshwar, Shagun Sodhani, Prof. Jian Tang (in an alphabetical order). Thank you very much!

Outside of Aalto University and MILA, I was fortunate to collaborate with many outstanding researchers: (in a chronological order) Stanisław Jastrzębski (Jagiellonian University, Cracow, Poland), Kenji Kawaguchi (MIT/Harvard), David Lopez-Paz (Facebook AI Research), Jisoo Jeong (Seoul National University, Korea), Thang Luong (Google Brain) and Quoc V. Le (Google Brain). Thank you for having joint research projects with me! I have learned a lot from all of you.

At various times, my doctoral studies have been funded by various sources: Prof. Tapani Raiko's Academy of Finland grant, Prof. Juho Kannala's grants, Prof. Yoshua Bengio's grants and Nokia scholarship. Thank you all for being kind to me. I would also like to convey my heartfelt gratitude to all the non-academic staff at Aalto University and MILA for tirelessly helping in various matters related to my doctoral studies and research visits. Although I wish I could list all of these people, I can only list a few: (in an alphabetical order) Päivi Koivunen, Paula Latvala, Julie Mongeau, Sorana Nagy, Linda Peinthiere and Jenna Vasik. Thank you all!

I express my gratitude to Prof. José Miguel Hernández-Lobato of University of Cambridge, the opponent in my defense. I would like to thank Prof. Ben Glocker of Imperial College of London and thank Prof. José Miguel Hernández-Lobato again for being the pre-examiners and providing their valuable comments and thoughts about the dissertation.

I was fortunate to spend half of the years during my doctoral studies in Finland and the other half in Canada. These years have been enriching and exciting both academically and personally. Canada, on one hand, has probably the biggest concentration of researchers working on Deep Neural Networks; studying and working there allowed me to broaden my research spectrum and find collaborators with whom I can foresee myself collaborating for many years to come. On the other hand, Finland has a rich tradition of research on neural networks, since as early as the 1970s when Seppo Linnainmaa invented the "back-propagation" algorithm;

a foundational algorithm for training all the modern-day Deep Neural Networks. It is a delight to watch that Deep Learning research is gaining rapid momentum again in Finland since its revival across the world in the mid 2000s. I hope in my own little ways, I will be able to help foster the growth of Deep Learning research in Finland, in the years to come.

I would also like to thank all the people I have met in Finland and Canada, and these countries in general for having given me this enormous opportunity.

Again, I can not list all the friends I have met in Finland, but let me try to thank at least a few: Kiran Garimella, Akash Kumar Dhaka, Kunal Ghosh, Rinu Boney, Antti Keurulainen, Pradeep Eranti and Luiza Sayfullina (in a random order). Kittos paljon!

Lastly but certainly not the least, my gratitude goes to my parents and siblings, whose love and support has encouraged me to storm through some of the most challenging moments of my doctoral studies.

Montreal, Canada, March 30, 2020,

Vikas Verma

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, Yoshua Bengio . Manifold Mixup: Better Representations by Interpolating Hidden States. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, Long Beach, California, USA, volume 97, pages: 6438–6447, 2019.

II Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio. On Adversarial Mixup Resynthesis. In *2019 Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, pages:4346–4357, 2019.

III Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio. Interpolated Adversarial Training: Achieving Robust Neural Networks Without Sacrificing Too Much Accuracy. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (AISec'19)*, London, United Kingdom, pages:95-103, 2019.

IV Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, David Lopez-Paz. Interpolation Consistency Training for Semi-Supervised Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, Macao, China, pages:3635–3641, 2019.

V Vikas Verma, Meng Qu, Alex Lamb, Yoshua Bengio, Juho Kannala, Jian Tang. GraphMix: Improved Training of Graph Neural Networks for Semi-Supervised Learning. Submitted for review, *https://arxiv.org/pdf/1909.11715.pdf*, January 2020, 8 pages.

VI Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, Jian Tang. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *Eighth International Conference on Learning Representations (ICLR 2020, spotlight)*, Addis Ababa, Ethiopia, 2020.

VII Stanislaw Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, Yoshua Bengio. Residual Connections Encourage Iterative Inference. In *6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada, 2018.

Author's Contribution

Publication I: "Manifold Mixup: Better Representations by Interpolating Hidden States"

Vikas Verma came up with main idea and designed and executed most of the experiments. He and Alex Lamb had the main responsibility of writing the article. Chrisopher Beckham conducted the GAN based experiments. All the co-authors reviewed and proposed the suggestions to the manuscript.

Publication II: "On Adversarial Mixup Resynthesis"

Vikas Verma and Christopher Beckham jointly came up with the main idea. Vikas Verma further refined the idea and designed the experiments and contributed to writing the manuscript. Main responsibility to conduct the experiments and writing the manuscript was carried out by Christopher Beckham. All the co-authors reviewed, made suggestions and worked on improving the manuscript

Publication III: "Interpolated Adversarial Training: Achieving Robust Neural Networks Without Sacrificing Too Much Accuracy"

Vikas Verma and Alex Lamb jointly came up with the idea and shared the responsibility of designing the experiments, implementing the models. The main responsibility of the running experiments and writing the manuscript was on Vikas Verma.

Publication IV: "Interpolation Consistency Training for Semi-Supervised Learning"

Vikas Verma came up with the main idea and had the main responsibility of designing the experiments, executing the experiments and writing the manuscript. David Lopez-paz reviewed and polished the manuscript. All the authors reviewed and suggested improvements to the manuscript.

Publication V: "GraphMix: Improved Training of Graph Neural Networks for Semi-Supervised Learning"

Vikas Verma came up with the main idea and had the main responsibility of designing and executing the experiments and writing the manuscript. Meng Qu conducted the experiments GAT based experiments. All the authors reviewed and suggested improvements to the manuscript.

Publication VI: "InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization"

Vikas Verma designed and executed the experiments for the semi-supervised learning section. He also wrote that section and contributed to improve the overall manuscript. The main responsibility of executing the experiments and writing the manuscript was on Fan-Yun Sun. All the authors reviewed and suggested improvements to the manuscript

Publication VII: "Residual Connections Encourage Iterative Inference"

Vikas Verma contributed to all aspects of the project, including idea, designing, implementing and executing the experiments and writing the manuscript. The main responsibility of experiments and writing the manuscript was shared jointly by Stanislaw Jastrzebski and Devansh Arpit. All the authors reviewed and suggested improvements to the manuscript

List of Figures

Abbreviations

AMR Adversarial Mixup Resynthesizer

CNN Convolutional Neural Networks

DNN Deep Neural Networks

GCN Graph Convolutional Networks

GNN Graph Neural Networks

IAT Interpolated Adversarial Training

ICT Interpolation Consistency Training

KL Kullback-Leibler divergence

MLP Mutilayer Perceptron

ResNet Residual Networks

RNN Recurrent Neural Networks

SGD Stochastic gradient descent

SSL Semi-supervised Learning

1. Introduction

1.1 Aim of this Thesis

Machine learning systems are prevalent across many applications these days: examples include automatically translating one language to another, recognizing objects in an image/video, web searches, recommendations on e-commerce websites, automatic diagnosis of diseases etc.

The conventional machine learning systems are composed of two disjoint subsystems: The first subsystem, called as a *feature extractor* is used to extract suitable hand-crafted features(representations) from the raw input which can be used by the subsequent subsystem, which is often a *classifier*, to identify patterns in the raw input. However, this approach has a major limitation: *designing the hand-crafted representations require considerable domain expertise and careful engineering*. Due to this limitation, eliminating the need for the hand-crafted representations is an important requirement for the successful application of the machine learning systems across various domains.

Contrary to the conventional machine learning systems, Deep Neural Networks (DNNs) are composed of a stack of simple but non-linear modules, with the first module transforming the *raw inputs* and the subsequent modules transforming the *representations* (outputs) of the previous modules. This hierarchy of modules allows them to learn from *simple* to more *complex* representations. By increasing the number of these modules in the stack, increasingly complex representations can be learned. This kind of learning mechanism eliminates the need for the hand-crafted representations and thus it has made DNNs, popularly known as *deep learning*, a dominant approach to machine learning in recent years.

As such, DNNs are a *deeper* (substantially more number of stacked modules) versions of earlier learning methods known as *artificial neural networks*. The research in artificial neural networks started almost six decades ago when Rosenblatt [123] presented the *perceptron* learning algo-

rithm. Since then various artificial neural networks have been proposed that include, but are not limited to, multilayer perceptron [125], radial basis function networks [19], Hopfield networks [67], Botlzmann machines[1], autoencoders [7], sigmoid belief networks [108], self-organizing maps[83], convolutional neural networks [92], recurrent neural networks [65]. More recent networks include residual networks [57, 143], graph neural networks [170], memory networks [168], generative adversarial networks [45], variational autoencoders [78] and attention based networks [5, 159], among others.

The resurgence of interest in DNNs happened during the mid 2000s. This resurgence can be attributed to three primary factors: success in training deeper networks using the layer-wise pretraining [62, 11], the availability of a large amount of data (such as Imagenet [30]) and the ability to harness this large amount of data using special purpose parallel computing hardware such as graphics processing units (GPUs) [84].

The success of current DNNs spans across a wide range of applications where these methods achieve human-level (or even superhuman level) performances. For example, various versions of DNNs have become state-of-the-art in numerous computer vision applications such as object recognition, detection, localization, segmentation, action recognition [57, 84, 60, 21]. Similarly, for natural language understanding, DNNs are the current state-of-the-art [145, 159]. Speech recognition has been also transformed by DNNs [54]. In other areas as well, such as game playing [137], reinforcement learning [104], medical diagnosis [134] and generative modeling [45], DNNs are the best performing methods.

While the success of DNNs is indisputable across many domains, there are still some caveats. DNNs typically consist of millions of parameters, thus having a sufficiently large training dataset is an important requirement for these networks to *generalize* on unseen examples; if the training dataset is too small, these networks have the tendency to *memorize* the training samples instead of extracting meaningful features. This is at odds with how humans are able to learn abstract concepts from a handful of examples. In some domains, it is relatively easier to collect a large amount of data. Consider the image classification task, where one can use a data labeling service to label a sufficiently large number of images. However, this is not possible for all the domains. In many domains a large amount of unlabeled data is available, but annotating large amounts of data is often not affordable due to high cost and time requirement. For example, consider the task of medical image segmentation. In this task, to create an annotated sample, each pixel of an image is needed to be carefully labeled by a medical expert, which is very cost and time intensive. In other domains, creating a large training dataset is simply not feasible due to the nature of the domain. Let us consider the task of *drug repositioning*. In this task, the objective is to predict the interaction of a drug-disease pair.

Since there are only very few numbers of known drug-disease pairs that interact, the training dataset for this task is fundamentally limited by the nature of the task and extending it further is not possible. Furthermore, in other domains, collecting a large dataset may not be possible due to privacy reasons.

The aforementioned challenges in collecting the large annotated training datasets motivate the need for the algorithms that can improve the generalization of a DNN using only a limited amount of annotated data. The aim of this thesis is to propose such algorithms.

The main underlying hypothesis of the algorithms proposed in this thesis is that, the generalization of Deep Neural Networks can be improved by designing constraints on how these networks behave *not only* on the training samples *but also* at the combination (mixing) of the training samples. We consider these combinations both in the input space as well as the hidden representation space. More specifically, this thesis explores a class of algorithms that improve generalization in Deep Neural Networks by synthetically constructing examples by mixing the training samples in the input space or/and in the hidden representation space and designing the appropriate constraints about how the network should behave on these synthetic samples in various learning paradigms such as supervised, unsupervised, semi-supervised, adversarial training and graph structure based learning.

In addition to the above mixing-based methods, this thesis also proposes a mutual-information based method for improving the generalization of Deep Neural Networks for the *graph-level* learning tasks. Finally, this thesis proposes a method to improve the generalization of ResNets that is based on their iterative inference interpretation.

Most of the methods proposed in this thesis are *general* in the sense that they can be applied to a wide range of neural network architectures. For instance, the methods proposed in Publication I, Publication III, Publication IV can be applied to any variant of the feedforward neural network, the method proposed in Publication II can be applied to any variant of the autoencoders and the methods proposed in Publication V, Publication VI can be applied to any graph neural network. The method proposed in Publication VII is an exception in this sense, since it is specific to ResNets [57]. ResNets have become the standard choice for various computer vision applications, thus we have given special attention to ResNets and proposed a method that is specific to them.

All of the proposed methods in this thesis have virtually no additional computation and memory cost over the underlying Deep Neural Networks. Furthermore, despite their simplicity, these methods achieve state-of-the-art performances across all the learning tasks considered in this thesis. Similarly, the follow-up work by other researchers has shown that these methods achieve state-of-the-art in various other learning tasks. This

makes these methods an appealing choice both in academia as well as for the industrial applications of Deep Neural Networks.

1.2 Contributions

This thesis contains seven publications that propose methods for improving the generalization ability of the contemporary and widely used Deep Neural Networks. This section lists the main contribution of each of the publications.

In Publication I, we study the limitations of the hidden states learned by the existing Feed-Forward Deep Neural Networks in the supervised setting and propose a novel regularizer named *Manifold Mixup* to address these limitations. The proposed method creates synthetic training samples by combining the hidden states of a pair of real samples. This enables Deep Neural Networks to learn more *discriminative* representations for supervised learning, which allows them to have better generalization capability. Through careful experimental analysis, we show that the proposed regularizer significantly outperforms other state-of-the-art regularizers such as Dropout, BatchNorm, Cutout and Mixup. In Publication II, we propose a novel method named as *Adversarial Mixup Resynthesizer* for learning unsupervised representations in the Auto-encoder framework. This method is motivated by the idea that the Auto-Encoder can learn better features if the learned features are amenable to various forms of *mixing*. Through experiments, we show that the proposed method learns better representations for the downstream task of classification on various image datasets. In Publication III, we make an observation that the adversarial training learns uncompressed representations, which is a potential cause of its worse generalization on the *clean* samples. To address this, we propose *Interpolated Adversarial Training*, which combines *Manifold Mixup* with the adversarial training to train adversarially robust networks without degrading accuracy. In Publication IV, we propose an instance of the *consistency regularizer* for semi-supervised Deep Neural Networks, which is motivated by the idea that perturbations along the direction of other samples in the data distribution are good directions in which consistency should be enforced. Despite its simplicity, at the time of publication, this method was the state-of-the-art on various benchmark datasets.

In Publication V and Publication VI, we turn our attention to the learning problems involving the graph structured data. Broadly, these tasks can be categorized as *node-level* tasks and *graph-level* tasks. In Publication V, we propose *GraphMix*, which is a method for regularizing Graph Neural Networks for the node classification task. The proposed method proposes an adaptation of *Manifold Mixup* for graph structured data by training a Fully-connected network with a Graph Neural Network through shared pa-

rameters. Through detailed experimental results, we have shown that the proposed method achieves state-of-the-art results for node classification on many benchmark datasets. In Publication VI, we propose *InfoGraph*, which addresses the problem of unsupervised representations learning from graph structured samples (for example, learning the representations of the chemical compounds) based on the principle of maximizing the mutual information between the local and global representations of the sample. Furthermore, we extend *InfoGraph* for semi-supervised graph classification using an auxiliary network that is trained using the labeled samples, and maximizing the mutual information between the representations of the primary and the auxiliary network. For both the unsupervised and semi-supervised learning, we demonstrate that *InfoGraph* outperforms other state-of-the-art methods.

Finally, in Publication VII, we investigate the learning mechanism of Residual Networks based on the *iterative refinement* view. Consequently, based on this view, we propose a parameter efficient training method for Residual Networks.

In addition, for all the publications in which I am the first author, I have made available the source code for reproducing the experimental results on my GitHub profile: https://github.com/vikasverma1077

1.3 Outline

This dissertation is organized into four main chapters (chapter 2 to chapter 5). In each of these chapters, I start by describing the problem setting, current approaches to solve this problem and the limitations/drawbacks of the current approaches. This is followed by the motivation, the summary and the contributions of the publications. At the end of the chapter, I discuss ideas for future work whenever it is possible.

Chapter 2 encompasses Publication I, Publication II and Publication III. This chapter presents proposed methods for learning better representations in the supervised, unsupervised and adversarial training settings. I start the chapter by describing the limitations of *Expected Risk Minimization* for training Deep Neural Networks in section 2.1. This is followed by the current data augmentation methods that aim to address these limitations and the pitfalls of these data augmentation methods. This, in turn, is followed by the motivation, insights and summary of *Manifold Mixup* (Publication I). In section 2.2, I briefly describe the working principle of the Auto-Encoders, which is followed by the motivation and summary of *Adversarial Mixup Resynthesizer* (Publication II). In section 2.3, I describe how adversarial training leads to poor performance on unperturbed samples and the justification and summary of *Interpolated Adversarial Training* (Publication III), which is a proposed method to address this issue.

In chapter 3, I discuss the principle of *consistency regularizers* for semi-supervised learning, which is followed by the discussion of various current state-of-the-art methods that are based on this principle and their limitations. Following this, I discuss the motivation and summary of the proposed method: *Interpolation Consistency Training* (Publication IV).

In chapter 4, I start by giving arguments about why learning from the graph structured data is an important learning problem. Further, I give an overview of various Deep Neural Networks for graph structured data (such as Recurrence based Graph Neural Networks, Auto-encoder based Graph Neural Networks and Convolution based Graph Neural Networks). This is followed by the motivation and summary of *GraphMix* (Publication V), which is a regularizer for Graph Neural Network based node classification task and *InfoGraph* (Publication VI), which is an unsupervised and semi-supervised learning method for graph-level learning tasks.

In chapter 5, I describe the *degradation problem* associated with training the deep non-residual networks and how Residual Networks (ResNets) address this problem. Following this, I discuss various interpretations of ResNets. Further, I summarize Publication VII which investigates ResNets from the iterative estimation view.

I have written Chapter 2 to Chapter 5 in such a way that they can be read independently of the other chapters. I hope that the reader, even without much background in Deep Neural Networks, will understand the basic principles and methods presented in these chapters.

Finally, in Chapter 6, I briefly summarize the contributions of this thesis once again and propose the ideas for future work.

2. Learning Better Representations by Interpolating Hidden States

Deep neural networks excel at making accurate predictions on the training data, but often provide incorrect and yet overly confident predictions when evaluated on slightly different test examples. This includes distribution shifts, outliers, and adversarial examples. This phenomenon is known as the *generalization gap* in the performance of Deep Neural Networks. In this chapter, we present a new class of algorithms that improve the generalization of Deep Neural Networks by constructing *mixing* based synthetic samples. In Section 2.1, we start by exploring the limitations of the *Expected Risk Minimization* which is a commonly used principle for training Deep Neural Networks, and present *Manifold Mixup* as a method for addressing these limitations in the fully supervised setting.

The generalization of Deep Neural Networks can be also improved by learning the unsupervised representations using a large amount of unlabeled data and consequently fine-tuning these representations using a limited amount of labeled data. This approach is particularly appealing when the labeled data is scarce and thus training a Deep Neural Network in a fully supervised setting with a reasonable test performance is not possible. The unsupervised representation learning is an active area of research and various methods have been proposed to achieve this objective [1, 62, 11, 117]. In section 2.2, we briefly describe the unsupervised representation learning using a particular class of algorithms known as Autoencoders. Following this, we describe *Adversarial Mixup Resynthesis*, which is an approach for improving the representations learned by the Autoencoders using the *mixing* based methods.

Deep Neural Networks exhibit substantially reduced performance when tested on the *adversarial examples*. This limitation makes it challenging to deploy Deep Neural Networks in safety-critical settings. Adversarial training is a popular approach to address this limitation, however, it gives rise to another problem: Deep Neural Networks trained with Adversarial training become relatively more robust to the adversarial examples but their generalization to the *clean* sample degrades significantly in comparison to the normal training. In section 2.3, we discuss this problem associated

with Adversarial training and present *Interpolated Adversarial Training* as a method to address this. Finally in section 2.4, we discuss the future directions for improving the generalization of Deep Neural Networks using the mixing based methods.

2.1 Generalization Gap in Fully-supervised Deep Neural Networks

Let us consider a supervised learning problem: we are interested in learning a parameterized function $f \in \mathscr{F}$ that maps a random feature vector $\mathbf{x}$ to a random target vector $\mathbf{y}$, where the joint probability of $\mathbf{x}$ and $\mathbf{y}$ is defined as $P(\mathbf{x},\mathbf{y})$. We can learn such function f by defining a loss function $\mathscr{L}$ that measures the difference between the prediction $f(\mathbf{x})$ and the target vector $\mathbf{y}$, and minimizing the expectation of this loss function $\mathscr{L}$ over the data distribution $P(\mathbf{x},\mathbf{y})$. This expectation of the loss over the data distribution is given as:

$$R(f) = \int \mathscr{L}(f(\mathbf{x}),\mathbf{y}) dP(\mathbf{x},\mathbf{y}) \tag{2.1}$$

Minimizing $R(f)$ is known as *expected risk minimization*. Usually we do not know the functional form of the data distribution $P(\mathbf{x},\mathbf{y})$, rather we have access to only a finite set of samples $\mathscr{D} = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^n$ drawn from the distribution $P(\mathbf{x},\mathbf{y})$. Using this finite set of samples, we can approximate the *expected risk* by the *empirical risk* as follows:

$$R_{\text{empirical}}(f) = \sum_{i=1}^{n} \mathscr{L}(f(\mathbf{x}_i),\mathbf{y}_i) \tag{2.2}$$

Learning the parameters of the function f by minimizing Empirical Risk is known as Empirical Risk Minimization (ERM) [158].

Empirical Risk Minimization has been a widely used principle for training the supervised Deep Neural Networks. Despite being efficient to compute, there are clearly some limitations.

- In practice, we usually only have a finite number of samples from $P(\mathbf{x},\mathbf{y})$. Furthermore, in many domains, the number of samples is rather small. One particular example of such domains is the medical diagnosis of rare diseases; since only a small number of people have a particular disease, the training datasets for the rare diseases are very small. Using ERM, we may be able to make $\mathscr{L}(\mathbf{x},\mathbf{y})$ small at these finite training points, but ERM does not pose any constraints on the loss on some unseen samples from $P(\mathbf{x},\mathbf{y})$. If the number of parameters in f is sufficiently larger than the number of the training samples n, then this will lead to Deep Neural

Networks simply *memorizing* the training samples instead of learning the useful representations from them. Zhang et.al. [178] have shown that large Deep Neural Networks can memorize the training data even in the presence of a strong regularizer. Thus ERM can not guarantee *generalization* on the unseen samples from the same distribution $P(\mathbf{x},\mathbf{y})$ if only a small number of training samples are available.

- ERM principle does not put any constraint on how the function f should behave on the samples which are *only slightly different* from the training data. Szegedy et.al. [148] have shown that the predictions of a Deep Neural Networks trained with ERM change drastically on the adversarial samples which are slightly outside from the training distribution.

- We may also wish that the function f indicates reduced confidence in its prediction on the samples from the classes which it has not been trained on. More specifically, we may want to learn a function f that does not make highly confident predictions on the test samples which are away from the manifolds of the classes used in training. ERM does not provide any guarantees in this case because the functions trained with ERM have a sharp decision boundary. For example, if the function f is trained using ERM to distinguish between dogs vs cats, it will still make highly confident predictions on the test image of other classes (for example, on an image of an aeroplane).

To address the above mentioned limitations of the ERM, we consider a class of methods known as *Data Augmentation*. In Data Augmentation, the support of the training set is enlarged by constructing synthetic samples. We consider a few basic approaches for generating synthetic examples:

- **Vicinity based Data Augmentation**: This is arguably the simplest technique to generate synthetic examples. It involves defining a *vicinity* distribution around each sample in the training dataset. Synthetic samples are sampled from this vicinity distribution around each sample and added to the training set to enlarge its size. *Vicinal Risk Minimization* [25] formalizes the concept of the vicinity distribution. In particular, [25] proposes to use a Gaussian distribution around each training sample as a vicinity distribution. Using this formulation, given a training sample $(\mathbf{x},\mathbf{y})$, a synthetic sample is generated as $(\mathbf{x}+\delta,\mathbf{y})$, where δ is a random vector sampled from the Gaussian distribution $\mathcal{N}(0,\sigma^2)$.

  Other methods use domain specific knowledge for defining the vicinity distribution. For example, for image classification, the vicinity distribution of an image is defined as the set of its rotations, cropped versions, scaled versions or horizontally flipped versions. Similarly for speech, [82] and [114] and for natural language, [173] propose methods for construct-

ing synthetic samples in the vicinity of the training samples.

While being simple and effective, the above kind of data augmentation techniques have some limitations. For example, while Gaussian vicinity requires carefully tuning a hyperparameter variance σ^2 of the Gaussian noise $\mathcal{N}(0,\sigma^2)$, other above mentioned data augmentation techniques are domain specific and may require significant domain expertise to design them.

- **Generation**: This approach involves using an explicit density model or generative model to generate synthetic data points. There can be various ways in which this generative model can be formulated. For example, we can learn a parameterized distribution $G_\theta(\tilde{\mathbf{x}}|\mathbf{x})$ which generates semantically similar images $\tilde{\mathbf{x}}$ given an input image $\mathbf{x}$. Using this distribution, given a training sample $(\mathbf{x}_i,\mathbf{y}_i)$, we can construct a synthetic sample $(\tilde{\mathbf{x}}_i,\mathbf{y}_i)$, where $\tilde{\mathbf{x}}_i \sim G_\theta(\tilde{\mathbf{x}}|\mathbf{x}=\mathbf{x}_i)$. Another methods involves learning a parameterized distribution $G_\theta(\tilde{\mathbf{x}}|\mathbf{z},\mathbf{y})$, where $\mathbf{z} \sim P(\mathbf{z})$ is a random vector drawn from a random distribution. Given a random vector $\mathbf{z}$ and a class label $\mathbf{y}$, distribution $G_\theta(\tilde{\mathbf{x}}|\mathbf{z},\mathbf{y})$ is trained to produce the samples $\mathbf{x}$ which are semantically similar to the samples from class $\mathbf{y}$. In yet another method, a sample $\mathbf{x}$ from a given class $\mathbf{y}$ can be generated using a textual description of the class. These methods have been studied in [73, 102, 120, 128] among others. One disadvantage to this approach is that it involves training another model, which can potentially be more difficult than learning the original function f.

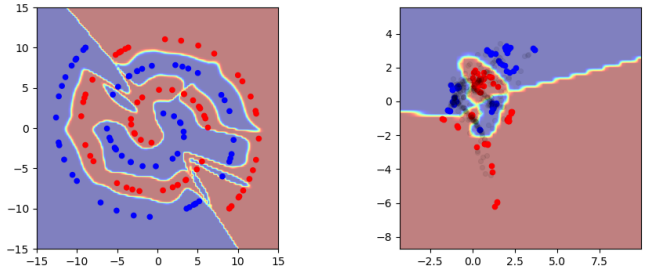
- **Synthesizing new samples by combining training samples**: A surprising and relatively simple approach, Mixup [180], involves constructing synthetic examples by taking the linear interpolation of two randomly chosen samples from the training data. This approach is appealing in practice because unlike the *Vicinity based Data Augmentation* approaches mentioned above, it is data-agnostic and unlike the *Generation* based approaches, it does not require to train an additional network and additional computation cost. We describe the Mixup [180] more formally in the following:

  Let us suppose that $\mathbf{x}_i,\mathbf{x}_j$ are input feature vector of two random training samples and $\mathbf{y}_i,\mathbf{y}_j$ are their corresponding one-hot target vectors, then in the simplest form of Mixing based approaches, called as Mixup method [180], a synthetic sample $(\tilde{\mathbf{x}},\tilde{\mathbf{y}})$ is constructed as:

$$\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j \tag{2.3}$$

$$\tilde{\mathbf{y}} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j \tag{2.4}$$

  where $\lambda \in [0,1]$.



(a) **(b)**

Figure 2.1. An experiment showing decision Boundary and hidden state representations of a Deep Neural Network trained on the 2D spiral dataset. Clearly the decision boundary is not optimal and the hidden states are not linearly separable. (Figure reproduced from Publication I)

Essentially, Mixup enforces an inductive bias in the model that the prediction on the linear interpolation of the input feature vectors should match the linear interpolation of the corresponding target vectors. Thus encouraging the model to behave linearly in between the training samples. Using this simple inductive bias, Mixup has been shown to improve generalization, reduce memorization, improve the robustness to adversarial examples [180]. We note that concurrent to Mixup, Between-Class learning [153] also proposes data augmentation based on the linear interpolation of the training samples.

2.1.1 Manifold Mixup

Although Mixup achieves better generalization by enforcing the constraint that the model should behave linearly in-between a pair of samples from the training set, it does not enforce any constraint on the hidden states of the training samples. We argue that by applying carefully designed constraints on the hidden states of the samples, we can learn better hidden states that will lead to the models that can generalize even better.

To get the right intuition about what are the properties of the "better" hidden states, we start by exploring what are the limitations of Deep Neural Networks trained with ERM from the perspective of the hidden states. Let us consider a simple two-dimensional classification task on a toy dataset of Figure 2.1

In this example, vanilla training of a Deep Neural Network leads to a complex arrangement of the hidden representations (Figure 2.1b). Moreover, every datapoint in both the input representation space (Figure 2.1a) and hidden representation space (Figure 2.1b) is assigned a prediction with very high confidence. This includes points (depicted in black) that

correspond to inputs off the data manifold! Now, we can take some time to think about what are the desirable properties of the better hidden representations for the classification task? Intuitively, the class specific representations should be concentrated into smaller volumes (more discriminative hidden states). This will lead to learning simpler classification functions to solve the task; which is a desirable property from the perspective of Occam's Razor. Moreover, the hidden states corresponding to off-manifold data points should be assigned a low confidence prediction.*Manifold Mixup* achieves both these objectives. In the following, we describe *Manifold Mixup* . For a theoretical as well as an intuitive justification for why *Manifold Mixup* achieves these objectives, refer to Publication I.

Consider training a deep neural network $f(\mathbf{x}) = f_k(g_k(\mathbf{x}))$, where g_k denotes the part of the neural network mapping the input data to the hidden representation at layer k, and f_k denotes the part mapping such hidden representation to the output $f(\mathbf{x})$. Training f using *Manifold Mixup* is performed in five steps. First, we select a random layer k from a set of eligible layers $\mathbb{S}$ in the neural network. This set may include the input layer $g_0(\mathbf{x})$. Second, we process two random data minibatches $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x}',\mathbf{y}')$ as usual, until reaching layer k. This provides us with two intermediate minibatches $(g_k(\mathbf{x}),\mathbf{y})$ and $(g_k(\mathbf{x}'),\mathbf{y}')$. Third, we perform Mixup [180] on these intermediate minibatches. This produces the mixed minibatch:

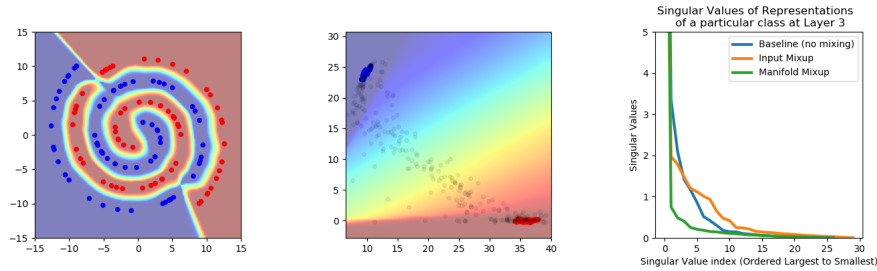
$$(\tilde{g}_k, \tilde{\mathbf{y}}) := (\mathrm{Mix}_\lambda(g_k(\mathbf{x}), g_k(\mathbf{x}')), \mathrm{Mix}_\lambda(\mathbf{y}, \mathbf{y}'),$$

where $\mathrm{Mix}_\lambda(\mathbf{a},\mathbf{b}) = \lambda \cdot \mathbf{a} + (1-\lambda)\cdot \mathbf{b}$. Here, (y, y') are one-hot labels, and the mixing coefficient $\lambda \sim \mathrm{Beta}(\alpha,\alpha)$ as proposed in Mixup [180]. For instance, $\alpha = 1.0$ is equivalent to sampling $\lambda \sim U(0,1)$. Fourth, we continue the forward pass in the network from layer k until the output using the mixed minibatch $(\tilde{g}_k, \tilde{\mathbf{y}})$. Fifth, this output is used to compute the loss value and gradients that update all the parameters of the neural network.

Mathematically, *Manifold Mixup* minimizes:

$$\mathscr{L}(f) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim P} \mathop{\mathbb{E}}_{(\mathbf{x}',\mathbf{y}')\sim P} \mathop{\mathbb{E}}_{\lambda\sim \mathrm{Beta}(\alpha,\alpha)} \mathop{\mathbb{E}}_{k\sim\mathbb{S}} \ell(f_k(\mathrm{Mix}_\lambda(g_k(\mathbf{x}), g_k(\mathbf{x}'))), \mathrm{Mix}_\lambda(\mathbf{y},\mathbf{y}')). \tag{2.5}$$

We show the decision boundary and hidden states of a Deep Neural Network trained with *Manifold Mixup* in Figure 2.2. We observe that Manifold mixup has three effects on learning when compared to vanilla training. First, it smoothens decision boundaries (from Figure 2.1a to Figure 2.2a). Second, it improves the arrangement of hidden representations and encourages broader regions of low-confidence predictions (from Figure 2.1b to Figure 2.2b). Black dots are the hidden representation of the inputs sampled uniformly from the range of the input space. Third, it flattens the representations (Figure 2.2c)



(a) **(b)** **(c)**

Figure 2.2. An experiment on a network trained on the 2D spiral dataset with a 2D bottleneck hidden representation in the middle of the network.(Figure reproduced from Publication I)

One interesting question is whether other state-of-the-art regularizers can achieve the same effects as that of *Manifold Mixup* . We examine this in Figure 2.3. It shows that the effects of *Manifold Mixup* are not accomplished by other well-studied regularizers (input mixup, weight decay, dropout, batch normalization, and adding noise to the hidden representations).

Moreover, *Manifold Mixup* addresses one another limitation of Mixup [180] as well: mixing the samples in the feature vector space may lead to the problem of *sample collision*. Let us assume that we create a synthetic sample by mixing the feature vectors and corresponding target vectors of two samples from classes A and B respectively. The feature vector of this synthetic sample may collide with the feature vector of a sample from another class C. Since the neural network is trained on both the synthetic samples and the real samples, it will be trained to produce different target vectors for the same feature vector! In *Manifold Mixup* , since the mixing happens in the hidden states, they can be learned to organize in such a way that *sample collision* does not occur or at least occurs less severely than Mixup.

The experimental results in Publication I demonstrate that *Manifold Mixup* can improve generalization on the test samples, robustness to adversarial examples and better generalization to the novel deformations in the test samples.

2.2 Unsupervised Representation Learning Using Auto-Encoders

Auto-encoders are fundamental building blocks for the unsupervised representation learning. In its most basic form, an Autoencoder is composed of two functions: an encoder function $f_\theta(.)$ that maps the input feature vector $\mathbf{x}$ to latent representation vector $\mathbf{z}$ and a decoder function $g_\phi(.)$ that maps the latent representation vector $\mathbf{z}$ to the reconstructed feature vector

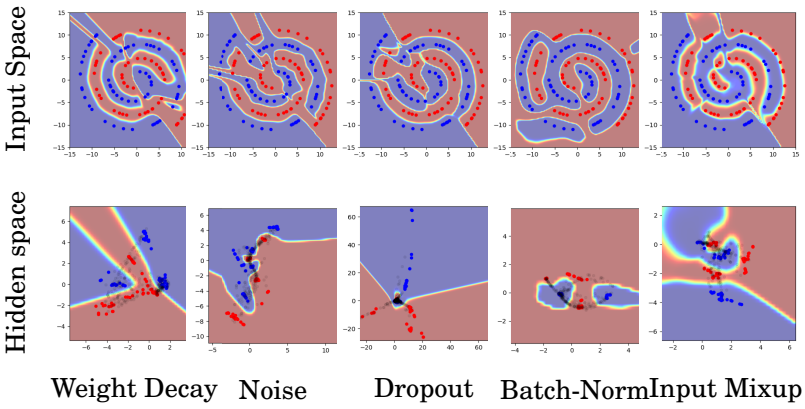


Figure 2.3. The same experimental setup as Figure 2.2, but using a variety of competitive regularizers. This shows that the effect of concentrating the hidden representation for each class and providing a broad region of low confidence between the regions is not accomplished by the other regularizers (although input space mixup does produce regions of low confidence, it does not flatten the class-specific state distribution). Noise refers to Gaussian noise in the input layer, dropout refers to dropout of 50% in all layers except the bottleneck itself (due to its low dimensionality), and batch normalization refers to batch normalization in all layers. (Figure reproduced from Publication I)

$\tilde{\mathbf{x}}$. These two functions are trained jointly to minimize the reconstruction error $\|\mathbf{x} - \tilde{\mathbf{x}}\|_2$

The primary goal of an Auto-encoder is to learn useful representations of the input data [10], which may help in downstream tasks such as classification [184, 68] or reinforcement learning [156, 48]. In its simplest form, the representations of auto-encoders can be encouraged to contain more 'useful' information by restricting the size of the bottleneck. More advanced methods for learning better representations in the Auto-encoders include, through the use of input noise (e.g., in denoising auto-encoders) [166], through regularisation of the encoder function [122], or by introducing a prior [78]. Yet other methods to learn better representations in the Auto-encoders include learning interpretable representations [26, 75], disentanglement of latent variables [96, 150] or maximization of mutual information [26, 8, 63] between the input and the code.

2.2.1 Adversarial Mixup Resynthesis

Different from the above mentioned methods, in this section, we present a mixing based Auto-encoder (Publication II). The underlying hypothesis of our method is: *an Auto-encoder learns more useful latent representations if the latent representations are such that the decoded output from the mixed latent representations of two or more data points is a semantically meaningful combination of the corresponding datapoint*. Consider a dataset of facial images: If the Auto-encoder simply memorizes how to reconstruct

a small number of facial images, then decoding the mixed representations of various facial images will most likely not be a realistic facial image. Instead, if latent representations have captured the semantic features such as age, gender, color, etc. effectively, then decoding the mixed latent representations will likely give a facial image that has the combination of these semantic features from the corresponding facial images. In this case, since latent representations have captured factors such as age, gender, etc, these representations will be more useful for down-stream tasks such as gender classification or age prediction.

While various Auto-encoder methods have been shown to have the implicit ability to *mix* between two data points (i.e. by linearly interpolating the latent representations of two data points and decoding it, an Auto-encoder produces a data point which is a semantic combination of the corresponding data points) [33, 47, 18], enforcing such kind of characteristic explicitly in the objective function of the Auto-encoders has received little attention [14, 126]. In the following, we formulate a method (Publication II) to explicitly enforce such kinds of characteristics in the Auto-encoders and also study different ways of mixing the latent representations instead of only the linear interpolations.

Let us consider an auto-encoder model $F(\cdot)$, with the encoder part denoted as $f(\cdot)$ and the decoder $g(\cdot)$. In an auto-encoder we wish to minimise the reconstruction, which is simply:

$$\min_{F} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \|\mathbf{x} - g(f(\mathbf{x}))\|_2 \tag{2.6}$$

Because auto-encoders that are trained by pixel-space reconstruction produce low quality images (characterized by blurriness), we augment this baseline by adding an adversarial game to the reconstruction (as done in [90]). In turn, the discriminator D tries to distinguish between real and reconstructed $\mathbf{x}$, and the auto-encoder tries to construct 'realistic' reconstructions so as to fool the discriminator. This formulation serves as our *baseline* (to make this clear throughout this work, we call this 'AE + GAN'), which can be written as:

$$\begin{aligned} &\min_{F}\ \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\ \lambda \|\mathbf{x} - g(f(\mathbf{x}))\|_2 + \ell_{GAN}(D(g(f(\mathbf{x}))), 1) \\ &\min_{D}\ \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\ \ell_{GAN}(D(\mathbf{x}), 1) + \ell_{GAN}(D(g(f(\mathbf{x}))), 0), \end{aligned} \tag{2.7}$$

where ℓ_{GAN} is a GAN-specific loss function. In our case, ℓ_{GAN} is the binary cross-entropy loss, which corresponds to the Jenson-Shannon GAN [45].

What we would like to do is to be able to encode an arbitrary pair of inputs $\mathbf{h}_1 = f(\mathbf{x_1})$ and $\mathbf{h}_2 = f(\mathbf{x_2})$ into their latent representation, perform some combination between them through a function we denote $\text{Mix}(\mathbf{h_1}, \mathbf{h_2})$ (more on this soon), run the result through the decoder $\mathbf{g}(\cdot)$, and then minimise some loss function which encourages the resulting decoded mix to look realistic. With this in mind, we propose *adversarial mixup resynthesis*

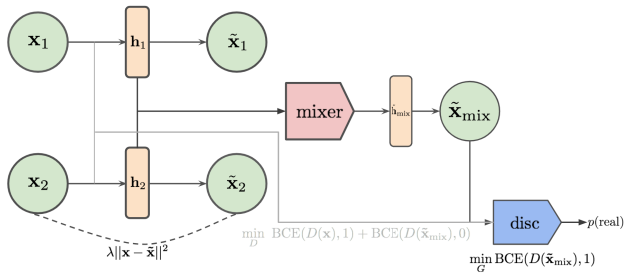


Figure 2.4. The systematic diagram for Adversarial Mixup Resynthesis (AMR). In addition to the auto-encoder loss functions, we have a mixing function Mix (called 'mixer' in the figure) which creates some combination between the latent variables $\mathbf{h}_1$ and $\mathbf{h}_2$, which is subsequently decoded into an image intended to be realistic-looking by fooling the discriminator. Subsequently the discriminator's job is to distinguish real samples from generated ones from mixes. Figure reproduced from Publication II

(AMR), where part of the auto-encoder's objective is to produce mixes which, when decoded, are indistinguishable from real images. The generator and the discriminator of AMR are trained by the following mixture of loss components:

$$\begin{aligned}
&\min_F \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim p(\mathbf{x})} \underbrace{\lambda\|\mathbf{x}-g(f(\mathbf{x}))\|_2}_{\text{reconstruction}} + \underbrace{\ell_{GAN}(D(g(f(\mathbf{x}))),1)}_{\text{fool D with reconstruction}} + \underbrace{\ell_{GAN}(D(g(\text{Mix}(f(\mathbf{x}),f(\mathbf{x}'))),1)}_{\text{fool D with mixes}} \\
&\min_D \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim p(\mathbf{x})} \underbrace{\ell_{GAN}(D(\mathbf{x}),1)}_{\text{label x as real}} + \underbrace{\ell_{GAN}(D(g(f(\mathbf{x}))),0)}_{\text{label reconstruction as fake}} + \underbrace{\ell_{GAN}(D(g(\text{Mix}(f(\mathbf{x}),f(\mathbf{x}'))),0)}_{\text{label mixes as fake}}.
\end{aligned} \tag{2.8}$$

The AMR model is shown in Figure 2.4.

2.2.2 Various Ways of Mixing the Latent Representations

There are many ways one could combine the two latent representations, and we denote this function $\text{Mix}(\mathbf{h_1},\mathbf{h_2})$. Manifold mixup [162] implements mixing in the hidden space through convex combinations:

$$\text{Mix}_{\text{mixup}}(\mathbf{h}_1,\mathbf{h}_2) = \alpha\mathbf{h}_1 + (1-\alpha)\mathbf{h}_2, \tag{2.9}$$

where $\alpha \in [0,1]$ is sampled from a Uniform$(0,1)$ distribution.

We also explore a strategy in which we randomly retain some components of the hidden representation from $\mathbf{h}_1$ and use the rest from $\mathbf{h}_2$. In this case, we would randomly sample a binary mask $\mathbf{m} \in \{0,1\}^k$ (where k denotes the number of feature maps) and perform the following operation:

$$\text{Mix}_{\text{Bern}}(\mathbf{h}_1,\mathbf{h}_2) = \mathbf{m}\mathbf{h}_1 + (1-\mathbf{m})\mathbf{h}_2, \tag{2.10}$$

where $\mathbf{m}$ is sampled from a Bernoulli(p) distribution (p can simply be sampled uniformly) and multiplication is element-wise. This formulation is interesting in the sense that it is very reminiscent of crossover in biological

reproduction: the auto-encoder has to organize feature maps in such a way that that *any* recombination between sets of feature maps must decode into realistic looking images.

We can generalise the above mixing functions to operate on more than just two examples. For instance, in the case of mixup (Equation 2.9), if we were to mix between examples $\{\mathbf{h}_1, \ldots, \mathbf{h}_k\}$, we can simply sample $\boldsymbol{\alpha} \sim \text{Dirichlet}(1, \ldots, 1)$[1], where $\boldsymbol{\alpha} \in [0,1]^k$ and $\sum_{i=1}^{k} \alpha_i = 1$ and compute the dot product between this and the hidden states:

$$\alpha_1 \cdot \mathbf{h}_1 + \cdots + \alpha_k \cdot \mathbf{h}_k = \sum_{j=1}^{k} \alpha_j \mathbf{h}_j, \tag{2.11}$$

One can think of this process as being equivalent to doing multiple iterations (or in biological terms, generations) of mixing. For example, in the case of a large k, $\alpha_1 \cdot \mathbf{h}_1 + \alpha_2 \cdot \mathbf{h}_2 + \alpha_3 \cdot \mathbf{h}_3 + \cdots = \underbrace{\underbrace{(\ldots(\alpha_1 \cdot \mathbf{h}_1 + \alpha_2 \cdot \mathbf{h}_2)}_{\text{first iteration}} + \mathbf{h}_3 \cdot \alpha_3) + \ldots}_{\text{second iteration}}$.

In Publication II, we quantitatively demonstrate that with the AMR formulation, we can learn more useful representations in the Auto-encoder framework. More specifically, we train a linear classifier on the representations extracted from the Auto-encoders and show that the AMR can achieve significant improvements over the baseline methods such as AE+GAN.

2.3 Achieving Adversarial Robustness without Sacrificing too much Accuracy

Although Deep Neural Networks have been highly successful across a variety of tasks including computer vision, speech, natural language understanding and healthcare applications, work on *Adversarial examples* [148] has shown that they can be easily fooled by adding carefully designed perturbations to the input data. By "carefully designed", we mean the perturbations which are imperceptible for the humans but Deep Neural Networks would often make completely different predictions on these perturbed inputs than the *clean* (unperturbed) inputs.

It is not surprising that an adversary of the system can try to benefit by fooling it using the adversarial examples. This limits the application of Deep Neural Networks in the areas where reliability and security are critical, which includes applications such as face recognition [131], self-driving cars [17], health care, and malware detection [91]. Defenses against the adversarial attacks are a very active research area and many defenses have been proposed, but most of them rely on *obfuscated gradients* [2] which

[1] Another way to say this is that for mixing k examples, we sample $\boldsymbol{\alpha}$ from a $k-1$ simplex. This means that when $k = 2$ we are sampling from a 1-simplex (a line segment), when $k = 3$ we are sampling from a 2-simplex (triangle), and so forth.

gives a false illusion of defense by lowering the quality of the gradient signal, without actually improving robustness [2]. Of these defenses, only Adversarial Training [44, 86] was found to still be effective without being suffering from the problem of obfuscated gradients. Adversarial Training encompasses crafting adversarial examples and using them during training to increase robustness against unseen adversarial examples. To scale Adversarial Training to large datasets and large models, often the adversarial examples are crafted using the fast single step methods such as FGSM [44].

However, Adversarial Training has a major disadvantage: it drastically reduces the generalization performance of Deep Neural Networks on the unperturbed samples, especially for the small networks. For example, [98] reports that adding Adversarial Training to a specific model increases the standard test error from 6.3% to 21.6% on CIFAR-10. This phenomenon makes Adversarial Training difficult to use in practice. If the tension between the performance and the security turns out to be irreconcilable, then many systems would either need to perform poorly or accept vulnerability.

How to train the Deep Neural Networks such that they are robust to adversarial samples as well as have high accuracy on clean samples is an open research problem. To this end, in Publication III, we propose *Interpolated Adversarial Training* (IAT)

In the following, we describe an information compression view of why Adversarial Training hurts performance on the clean test samples and describe IAT that addresses this issue.

2.3.1 Interpolated Adversarial Training

Why adversarial training hurts the performance on the clean test samples is an open research problem [98, 154, 118, 179]. We investigate this problem from the *information compression* view.

[152, 136] investigate a relationship between the compression of the information in the representations learned by the Deep Neural Networks and their generalization bounds; showing that a stronger generalization bound exists when the Deep Neural Networks have stronger compression.

To examine the degree to which the information compression is changed in the adversarially trained models vs the normal training (baseline models), we performed an experiment where we take the representations learned after training, and study how well these frozen representations are able to successfully predict fixed random labels. If the model compresses the representations well, then it will be harder to fit random labels.

We found that Adversarial Training made the representations much less compressed in comparison to normal training. More specifically, we ran a small 2-layer MLP on top of the learned representations to fit random binary labels. In all cases we trained the model for predicting the random

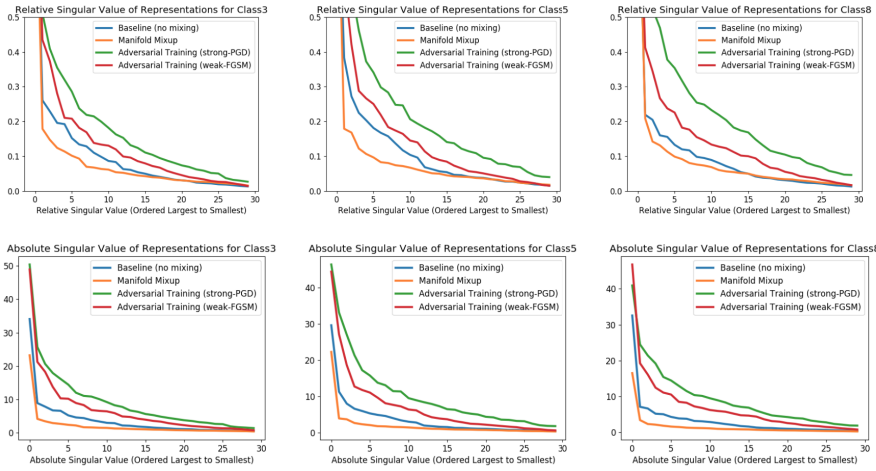


Figure 2.5. Adversarial Training (especially with PGD) training makes representations have substantially more directions of significant variability (both when measured in an absolute sense and when measured relative to the largest singular value).

labels for 200 epochs with the same hyperparameters. For fitting 10000 randomly labeled examples, we achieved accuracy of: 92.08% (Baseline), and 97.00% (PGD Adversarial Training). Since adversarially trained models can fit the random labels more easily, it suggests that Adversarial Training causes the learned representations to be less compressed.

To provide further evidence for a difference in the compression characteristics, we trained 5-layer fully-connected models on MNIST and considered a bottleneck layer of 30 units directly following the first hidden layer. We then performed singular value decomposition on the per-class representations and looked at the spectrum of singular values (Figure 2.5). We found that both Adversarial Training (with PGD) and Adversarial Training (with FGSM) increased the number of singular values with large values as compared to a baseline model.

As per the relationship drawn in [152, 136] between the information compression and the generalization, this explains why adversarial training hurts performance on clean samples.

On the contrary, in Publication I, we have demonstrated that Manifold Mixup significantly compresses the hidden state representations (also demonstrated in Figure 2.5). Based on these observations, in Publication III, we propose *Interpolated Adversarial Training*(IAT), which alleviates the above mentioned problems of Adversarial Training by augmenting it with the Mixup or Manifold Mixup training.

More concretely, in IAT, we train a Deep Neural Network using the interpolations of the adversarial examples along with the interpolations of the unperturbed examples. We use the techniques of Mixup [180] and Manifold Mixup[163] as ways of interpolating examples. Learning is

performed in the following four steps when training a network with IAT. In the first step, we compute the loss from an unperturbed (non-adversarial) batch (with interpolations based on either Mixup or Manifold Mixup). In the second step, we generate a batch of adversarial examples using an adversarial attack (such as Projected Gradient Descent (PGD) [98] or Fast Gradient Sign Method (FGSM) [44]). In the third step, we train against these adversarial examples with the original labels, with interpolations based on either Mixup or Manifold Mixup. In the fourth step, we obtain the average of the loss from the unperturbed batch and the adversarial batch and update the network parameters using this loss. Note that following [85, 154], we use both the unperturbed and adversarial samples to train the model with IAT and we use it in our baseline adversarial training models as well (refer to Publication III for a more detailed algorithm.) Through extensive experiments (refer to Publication III), we have demonstrated that IAT retains the performance on the clean test samples while being robust to adversarial samples.

2.4 Discussion and Future Directions

In this chapter, we have presented mixing based methods for improving the generalization of Deep Neural Networks. Specifically, we have presented methods for the fully-supervised learning, unsupervised representation learning and method for achieving better test accuracy with Adversarial Training. These methods have negligible added computation cost over the underlying Deep Neural Networks, which makes these methods particularly appealing for the large scale Deep Learning applications.

In the follow-up work by other researchers, Manifold Mixup has been successfully applied to other problem settings. [76] applied Manifold Mixup to U-Nets for prostate cancer segmentation. [105] applied Manifold Mixup to a recurrent (LSTM) model trained with CTC loss for handwriting recognition. These follow up works provide some preliminary evidence that Manifold Mixup can be applied to architectures other than CNNs and with tasks other than classification (note that in Publication I and Publication III we only considered CNNs and FCNs based classification tasks). For speech based applications, [119] used *Manifold Mixup* for acoustic scene classification and [15] used Manifold Mixup for voice spoofing detection. [56] considered using experience replay with quantized hidden states and also applied Manifold Mixup on the hidden states. In the few-shot learning paradigm, [99] used Manifold Mixup loss jointly with self-supervision loss. Additionally, they showed substantial improvements using Manifold Mixup on the Imagenet dataset and on a variant of Imagenet with novel deformations introduced during evaluation. CutMix [177] considered a variant of mixup where rectangular "cutouts" of the image are used to do

mixing between images. They found that this technique did not perform as well when applied in the hidden states (following the Manifold Mixup procedure). Understanding this better could be an interesting area for future work.

Another line of research has explored the ramifications of the spectral properties of the hidden layers of deep networks, which ties in closely with the theoretical results shown in the Publication I. For instance, in Publication I, we showed that using *Manifold Mixup* "flattens" the class-specific hidden representations learned by Deep Neural Networks, reducing the number of directions with substantial variability. The number of directions with significant variability has seen increasing theoretical study. [146] produced a generalization bound which explicitly depends on how quickly the spectrum of the hidden state decays (with a stronger bound for a faster decay, as seen with Manifold Mixup). That Manifold Mixup's property is explicitly class-specific, may suggest that even better bounds could be produced if they considered a class-specific (or at least target-dependent) spectral analysis.

Interpolated Adversarial Training has seen two direct follow-up research projects. Mixup Inference [113] considers using interpolations between multiple examples during evaluation as a way of improving adversarial robustness. They found that this achieves considerable adversarial robustness but at the expense of some amount of clean test accuracy. They also found that this technique performs better both in terms of clean accuracy and robustness when combined with *Interpolated Adversarial Training* [89]. Instance Adaptive Adversarial Training [6] uses an adaptive perturbation raidus for adversarial training based on how close points are to the margin and they found that this yields a more robust defense than IAT, but with somewhat worse generalization on clean examples.

There exist many future directions to be explored along the mixing based methods. For example, in Manifold Mixup, we randomly selected the mixing coefficient λ and the layer for mixing. Using a more principled way for these selections may lead to faster convergence of Manifold Mixup. Moreover, understanding the interaction of Manifold Mixup with other popular regularizers such as BatchNorm and Dropout can shed light on whether these regularizers are complementary to Manifold Mixup or not. Applying Manifold Mixup to various tasks in the natural language understanding is another interesting future direction. From the perspective of the unsupervised learning, mixing based methods can be combined with the auto-encoder based self-supervised methods such as [182] or with the more generic self-supervised methods such as [59]. Developing the theoretical understanding of the mixing based methods is yet another important future reserach direction.

3. Recent Advances in Semi-supervised Learning

In many application domains, such as computer vision, speech, natural language understanding, and medical diagnosis, Deep Neural Networks can achieve human-level or even super-human-level performance in a fully supervised setting by leveraging a large collection of labeled data. However, labeling large amounts of data is often prohibitive due to time, financial, and expertise constraints. Semi-supervised learning is an attractive alternative to supervised learning when the label data is scarce but substantially more amount of unlabeled data is available.

Current deep learning based semi-supervised learning approaches can be broadly categorized into two streams: (1) *Deep generative modeling* based approaches [79, 28] and (2) *Consistency regularization* based approaches.

While the deep generative modeling based approaches had achieved better results earlier on, the recent progress in semi-supervised learning has been primarily driven by the consistency regularization based methods. This chapter presents basic principles related to consistency regularization and how these principles have given rise to various methods for deep semi-supervised learning. We start by introducing the underlying assumptions and motivations of consistency regularization in Section 3.1. Next, we describe some of the previous state-of-the-art semi-supervised learning approaches based on consistency regularization. In Section 3.1.3, we summarize a consistency regularization method proposed in Publication IV. Finally, in Section 3.3 we discuss the major advantages of the method proposed in Publication IV and the future directions to improve existing consistency regularization approaches.

3.1 Consistency Regularization

Consistency regularization based approaches draw their motivation from the *cluster assumption* or equivalent *low-density separation assumption* [24]. In the following, we first describe the aforementioned assumptions and then present a general formulation that is used to enforce

these assumptions in various consistency regularization based deep semi-supervised Learning methods.

The *cluster assumption* posits that: if two samples belong to the same cluster in the input distribution, then they are likely to belong to the same class. The cluster assumption can be equivalently stated as *low-density separation assumption*: for the classification problems, the decision boundary is likely to traverse low-density regions in the sample space. The equivalence is easy to see: Two samples in the same cluster will be classified to the same class if and only if the decision boundary does not cut through that cluster, rather than it passes through a low density region in the sample space.

Consistency regularization methods enforce above mentioned *low density separation assumption* as a model prior by encouraging that the output of the model does not change under *realistic perturbations* (for example, small Gaussian noise or random translation/cropping of the images) to an unlabeled sample. One can easily see that such kind of constraints can be satisfied for all the samples only if the decision boundary lies in the low density region. Note that by *realistic perturbations* we mean a perturbation that does not change the semantic of the sample.

Now we mathematically define the consistency regularization. Let us assume that $\mathbf{x} \in \mathscr{D}_{UL}$ be an unlabeled sample, $\hat{\mathbf{x}}$ be a realistic perturbation to sample $\mathbf{x}$, and $f_\theta(\mathbf{x})$ be the prediction function, then the consistency regularization $R_\theta(\mathbf{x})$ can be defined as:

$$R_\theta(\mathbf{x}) = D(f_\theta(\mathbf{x}), f_\theta(\hat{\mathbf{x}})) \tag{3.1}$$

where $D(.,.)$ measures the distance between the prediction function's output on the *clean* and the *perturbed* sample. For example, $D(.,.)$ can be realized as mean-square error or Kullback-Leibler divergence (KLD). The loss term used to train the prediction function $f_\theta(\mathbf{x})$ is obtained by adding the consistency regularizer of Equation 3.1 to the supervised classification loss using a weighting hyperparameter. This seemingly simple principle has motivated a family of recent state-of-the-art methods for semi-supervised learning, which includes the Π-model [127, 87], Temporal Ensembling [87], Virtual Adversarial Training (VAT) [103], and the Mean-Teacher [149], among others. We describe these methods in the following section. We note that many of these methods also utilize a related principle: *models that are trained to be robust to noise in its parameters should generalize better than the models which are (overly) sensitive to such kind of noise* [4].

3.1.1 Stochastic Perturbation and Π Model

Stochastic Perturbation method of [127] proposes to apply linear or non-linear stochastic transformations $T(\mathbf{x})$ to a sample k times and minimize

the difference of model's prediction on each pair of these k transformed samples. Mathematically:

$$R_\theta(\mathbf{x}) = \sum_{i=1}^{k-1} \sum_{j=2}^{k} \left\| f_\theta(T^i(\mathbf{x})) - f_\theta(T^j(\mathbf{x})) \right\|_2^2 \tag{3.2}$$

where $T^i(\mathbf{x})$ is the i-th instance of transformation applied on $\mathbf{x}$.

Π model of [87] is closely related to the Stochastic Perturbation method of [127]. In addition to applying stochastic perturbations to the samples, Π model also uses perturbations in the models, such as Dropout, to compute different model predictions for a single input sample. Let us assume that $f_\theta^{'}$ and $f_\theta^{''}$ are two Dropout instances of the model f_θ, then the Π model consistency regularizer is given as:

$$R_\theta(\mathbf{x}) = \left\| f_\theta^{'}(T(\mathbf{x})) - f_\theta^{''}(T(\mathbf{x})) \right\|_2^2 \tag{3.3}$$

For image classification problems, both Stochastic Perturbation [127] and Π model [87] realize the non-linear stochastic perturbations using data-augmentation techniques (for example, random-cropping and random rotation).

3.1.2 Virtual Adversarial Training

We note that Stochastic perturbations used in [127, 87] might be inefficient in high dimensions, as only a tiny proportion of input perturbations are capable of pushing the decision boundary into low-density regions. To alleviate this problem, instead of applying stochastic perturbations to an input sample, Virtual Adversarial Training (VAT) finds an adversarial perturbation which causes maximum change to the model's output for the given input sample. Virtual Adversarial Training (VAT) is motivated by Adversarial Training [44, 148]. Formally, an adversarial perturbation r_{adv} in VAT is computed as:

$$\mathbf{r}_{adv} = \underset{\mathbf{r};\|\mathbf{r}_2\| \leq \epsilon}{\arg\max} D(f_\theta(\mathbf{x}), f_\theta(\mathbf{x}+\mathbf{r})) \tag{3.4}$$

where ϵ is a scalar hyperparameter that specifies the upper bound on the norm of the adversarial perturbation. VAT [103] further proposes to approximate the computation of r_{adv} as follows:

$$\mathbf{r}_{adv} \approx \epsilon \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \tag{3.5}$$

$$\text{where } \mathbf{g} = \nabla_{\mathbf{r}} D(f_\theta(\mathbf{x}), f_\theta(\mathbf{x}+\mathbf{r}))\Big|_{\mathbf{r}=\xi\mathbf{d}} \tag{3.6}$$

and $\xi \neq 0$ is a scalar hyperparameter whose value is set to be very small (such as $1e-6$) and $\mathbf{d}$ is a randomly sampled unit vector.

Based on the adversarial perturbation $\mathbf{r}_{adv}$ for a sample $\mathbf{x}$, the consistency regularization in VAT is computed as:

$$R_\theta(\mathbf{x}) = D(f_\theta(\mathbf{x}), f_\theta(\mathbf{x} + \mathbf{r}_{adv})) \tag{3.7}$$

3.1.3 Temporal Ensembling and Mean-Teachers

The methods described in Section 3.1.1 and 3.1.2 use the model's own prediction on the *clean* and the *perturbed* sample to design a consistency regularizer. However, there exists a major limitation: the model's prediction can be potentially inaccurate (especially in the beginning of the training and/or if the number of labeled samples are too few) and they can be potentially unstable (i.e. they can change rapidly over the course of the training). The inaccurate predicted-targets used in consistency regularizer can push the decision boundary of the model towards undesired directions instead of the low density regions of the sample space. Furthermore, the unstable predicted-targets can potentially make it more difficult for the model to converge. To make the predicted-targets $f_\theta(\mathbf{x})$ of an unlabeled sample $\mathbf{x} \in \mathscr{D}_{UL}$ more accurate and stable, Temporal Ensembling [87] and Mean-Teacher [149] employ self-ensembling technique. Specifically, Temporal Ensembling proposes to use the running average of the model's own predictions as the predicted-target $f_\theta(\mathbf{x})$. Inspired by Temporal Ensembling, Mean-Teacher [149] instead proposes to use a *Teacher* model parameterized by an exponential-moving-average of the parameters θ of the original prediction function. In practice, Mean-Teacher achieves better results than the Temporal Ensembling. Mean-Teacher has an added benefit in terms of reduced memory requirements as compared to the Temporal Ensembling; it does not require to store the running average of the model's prediction on all the unlabeled samples.

3.2 Interpolation Consistency Training

Building on this line of research of consistency regularization methods (Section 3.1), in Publication IV, we propose *Interpolation Consistency Training* (ICT).

ICT proposes a special form of consistency regularizer, whereby it encourages the prediction at an interpolation of unlabeled points to be consistent with the interpolation of the predictions at those points. This kind of consistency regularizer is motivated by a simple observation that *for applying consistency regularization, the perturbations in the directions of other samples are better than stochastic perturbations*. Why is this so?

Recall that, based on the *low density separation assumption*, our goal is to push the decision boundary in the low-density regions. This can be

achieved if, for a given sample $\mathbf{x}$, we can we find a perturbation δ such that $\mathbf{x}$ and $\mathbf{x}+\delta$ lie on opposite sides of the decision boundary. Although tempting, using random perturbations is an inefficient strategy, since the subset of directions approaching the decision boundary is a tiny fraction of the ambient space. Instead, consider interpolations $\mathbf{x}_i+\delta=(1-\alpha)\mathbf{x}_i+\alpha\mathbf{x}_j$ towards a second randomly selected unlabeled examples $\mathbf{x}_j$. Then, the two unlabeled samples $\mathbf{x}_i$ and $\mathbf{x}_j$ can either:

1. lie in the same cluster,

2. lie in different clusters but belong to the same class,

3. lie on different clusters and belong to the different classes.

Assuming the cluster assumption, the probability of (1) decreases as the number of classes increases. The probability of (2) is low if we assume that the number of clusters for each class is balanced. Finally, the probability of (3) is the highest. Then, assuming that one of $(\mathbf{x}_i,\mathbf{x}_j)$ lies near the decision boundary (it is a good candidate for enforcing consistency), it is likely (because of the high probability of (3)) that the interpolation towards $\mathbf{x}_i$ points towards a region of low density, followed by the cluster of the other class. Since this is a good direction to move the decision, the interpolation is a good perturbation for consistency-based regularization.

Based on this observation, using a sufficiently small positive scalar λ, we can design a simple consistency regularization based on Equation 3.1:

$$R_\theta(\mathbf{x}_i)=D(f_\theta(\mathbf{x}_i),f_\theta(\mathrm{Mix}_\lambda(\mathbf{x}_i,\mathbf{x}_j)) \tag{3.8}$$

$$\text{where } \mathrm{Mix}_\lambda(\mathbf{a},\mathbf{b})=\lambda\cdot\mathbf{a}+(1-\lambda)\cdot\mathbf{b} \tag{3.9}$$

Notice, we need λ to be sufficiently small for $(1-\lambda)\mathbf{x}_i+\lambda\mathbf{x}_j$ to be a *realistic perturbation* of $\mathbf{x}_i$.

While the consistency regularization scheme of Equation 3.8 is well-motivated, we further note that unlike random or adversarial perturbations of single unlabeled examples $\mathbf{x}_i$, our scheme involves two unlabeled examples $(\mathbf{x}_i,\mathbf{x}_j)$. Intuitively, we would like to push the decision boundary as far as possible from the class boundaries, as it is well known that decision boundaries with large margin generalize better [132]. In the supervised learning setting, one method to achieve large-margin decision boundaries is mixup [180]. In mixup, the decision boundary is pushed far away from the class boundaries by enforcing the prediction model to change linearly in between samples. This is done by, for random pairs of labeled samples $((\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}_j,\mathbf{y}_j))$, training the model f_θ to predict $\mathrm{Mix}_\lambda(\mathbf{y}_i,\mathbf{y}_j)$ at location $\mathrm{Mix}_\lambda(\mathbf{x}_i,\mathbf{x}_j)$. Here we extend mixup to the semi-supervised learning setting by training the model f_θ to predict the "fake

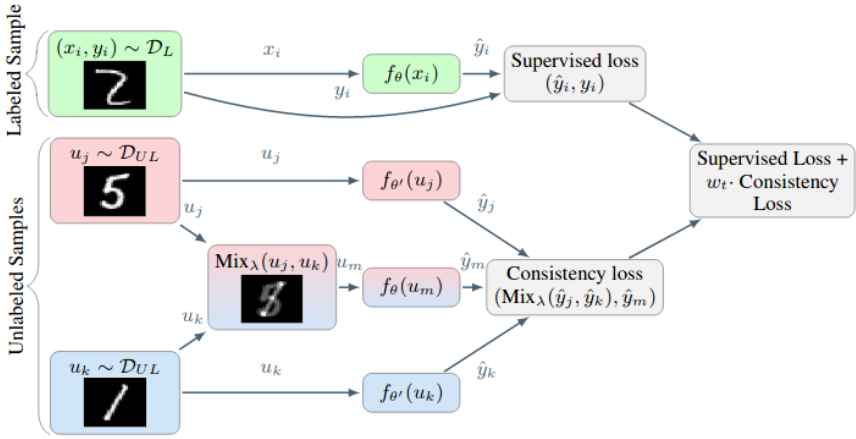


Figure 3.1. ICT uses a mean-teacher $f_{\theta'}$, where the teacher parameters θ' are an exponential moving average of the student parameters θ. During training, the student parameters θ are updated to encourage consistent predictions $f_\theta(\text{Mix}_\lambda(\mathbf{u}_j, \mathbf{u}_k)) \approx \text{Mix}_\lambda(f_{\theta'}(\mathbf{u}_j), f_{\theta'}(\mathbf{u}_k))$, and correct predictions for labeled examples $\mathbf{x}_i$. Figure reproduced from Publication IV.

label" $\text{Mix}_\lambda(f_\theta(\mathbf{x}_i), f_\theta(\mathbf{x}_j))$ at location $\text{Mix}_\lambda(\mathbf{x}_i, \mathbf{x}_j)$. In order to follow a more conservative consistent regularization, we encourage the model f_θ to predict the fake label $\text{Mix}_\lambda(f_{\theta'}(\mathbf{x}_i), f_{\theta'}(\mathbf{x}_j))$ at location $\text{Mix}_\lambda(\mathbf{x}_i, \mathbf{u}_j)$, where θ' is a moving average of θ, similar to Mean-Teacher [149].

Based on the above observations and motivations, we proposed the following consistency regularization in Publication IV

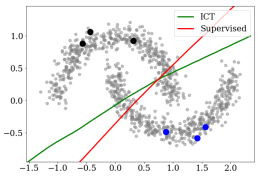
$$R_\theta(\mathbf{x}_i, \mathbf{x}_j) = D(\text{Mix}_\lambda(f_{\theta'}(\mathbf{x}_i), f_{\theta'}(\mathbf{x}_j)), f_\theta(\text{Mix}_\lambda(\mathbf{x}_i, \mathbf{x}_j))) \tag{3.10}$$

The population version of the regularizer in Equation 3.10 is given as:

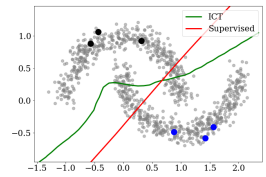
$$R_\theta(D_{UL}) = \mathop{\mathbb{E}}_{\mathbf{x}_i, \mathbf{x}_j \sim D_{UL}} \mathop{\mathbb{E}}_{\lambda \sim \text{Beta}(\alpha, \alpha)} D(\text{Mix}_\lambda(f_{\theta'}(\mathbf{x}_i), f_{\theta'}(\mathbf{x}_j)), f_\theta(\text{Mix}_\lambda(\mathbf{x}_i, \mathbf{x}_j))) \tag{3.11}$$

ICT has a major advantage as compared to VAT (Section 3.1.2): The additional computation cost associated with VAT (computing the gradient in Equation 3.5), makes it less appealing in situations where unlabeled data is available in large quantities. Furthermore, recent research has shown that training with adversarial perturbations can hurt generalization performance [106, 154].

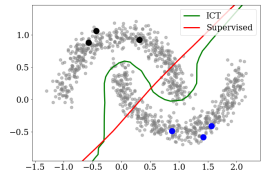
Despite its simplicity and no additional computation cost, ICT improves upon previous state-of-the-art methods across all benchmark architectures and datasets. The procedure for ICT is depicted in Figure 3.1. In Figure 3.2, we demonstrate how ICT training pushes the decision boundary into low density regions of input space, resulting in better classifiers. For more discussions and results, refer to Publication IV.



(a) After 100 updates



(b) After 500 updates



(c) After 1000 updates

Figure 3.2. Interpolation Consistency Training (ICT) applied to the "two moons" dataset, when three labels per class (large dots) and a large amount of unlabeled data (small dots) is available. When compared to supervised learning (red), ICT encourages a decision boundary traversing a low-density region that would better the unlabeled data. Both methods employ a multilayer perceptron with three hidden ReLU layers of twenty neurons. Figure reproduced from Publication IV.

3.3 Discussion and Future Directions

Deep Neural Networks are a highly flexible class of algorithms which have shown to be state-of-the-art across many domains. However, they depend on the availability of a large corpus of labeled data. Contrary to this, semi-supervised learning is a powerful learning framework that leverages unlabeled data to reduce the amount of labeled data required for successfully training Deep Neural Networks. With this motivation, researchers have recently started to develop semi-supervised methods for training Deep Neural Networks. While earlier of these methods were based on generative modeling (such as VAEs and GANs), lately the attention has shifted to consistency regularization based methods. Consistency regularization based methods are appealing because of their high generalization performance, training stability and ease of implementation.

Recent years have seen rapid improvements in the direction of consistency regularization based methods. Along these lines, we have proposed *Interpolation Consistency Training*(ICT) in Publication IV. Unlike previous consistency regularization methods such as VAT [103], ICT (Publication IV) does not has any significant additional computation or memory cost and hence it is an appealing semi-supervised learning method for the real-world applications. Furthermore, unlike the previous methods such as Stochastic Perturbation [127] and Π-model [87], which require domain-specific transformations, ICT is domain agnostic and can be applied to any

domain in which the input samples are presented as a fixed-length vector.

The consistency regularization proposed in ICT (Publication IV) has inspired a number of recent state-of-the-art methods for semi-supervised learning. The most notable examples include MixMatch [13], ReMixMatch [12] and FixMatch [139]. These methods use the same form of consistency regularization proposed in ICT along with some added techniques. One thing to note is that although these methods improve substantially over ICT when the number of labeled examples in small, these methods have an additional assumption that there is an effective form of data augmentation available, which may depend greatly on the domain.

For Domain Adaptation, Virtual Mixup Training (VMT) [100] uses the consistency regularization similar to ICT, but on the unlabeled data of the target domain. Furthermore, [40] proposes an active learning method based on ICT consistency regularizer.

ICT based methods have been applied to various other domains as well. [37] used the ICT method for semantic segmentation but uses a spatial-bernoulli mask (constructed from smoothed gaussian noise) to mix the images rather than doing linear mixing. [66] applied ICT to audio tagging. HODGEPODGE [135] applied ICT to audio event detection and found that it outperformed Mean Teacher [149] and MixMatch [13]. [88] used ICT for improving the generalization of speech separation models.

Some other approaches use a consistency regularizer similar to ICT in the *noisy lables* setting. [185] uses a meta-learning based procedure to weight between applying a consistency objective and using supervised learning as a function of how noisy the labels are for a set of data (such that it leans towards a consistency-based objective on examples with labels which are most noisy). This substantially improves results on datasets consisting of a small "trusted dataset" and a large dataset of examples with noisy labels. [55] also considered the problem of learning from a combination of clean and noisy-labeled samples, but used a fixed weighting between the supervised and consistency-based losses, leading to a simpler but less flexible procedure then [185].

Now we discuss some of the future directions. To start with, we note that ICT can be readily extended to the hidden states of a neural network as long as the hidden states are of fixed length for each sample. This is in a similar spirit of Manifold Mixup [162]. It has two advantages: (1) It will allow the hidden states to be more *discriminative* similar to *Manifold Mixup* [162], potentially achieving even better test accuracy. (2) The other more important benefit is that it will facilitate applying ICT to application domains where input samples do not have a fixed topology (for example, molecular graphs, 3D protein structures, etc).

The success of the consistency regularization based approaches for semi-supervised Learning depends heavily on achieving better predicted-targets. While methods such as Temporal Ensembling [87] and Mean-Teacher [149]

achieve reasonably accurate predicted-targets using the self-ensembling technique, there is still a scope to improve the accuracy of the predicted targets. One such potential direction is to use the meta-learning approach to *rectify* the predicted-targets of the Mean-Teacher method.

While the performance gap between fully supervised Deep Neural Networks and semi-supervised Deep Neural Networks is closing rapidly, there is still plenty of scope for improvement and we hope to see further development along these lines in the near future.

4. Learning from Graph Structured Data

Deep learning has revolutionized many applications where the data can be represented as a fixed-length vector (for example, images) or a set/sequence of fixed-length vectors (for example, language, audio, video, financial time-series data, weather time-series data, to name a few). However, there is an increasing number of applications where data are represented in the form of graphs. For example, a social network can be represented as a graph. Each node in the social network represents a user and the edges between the users encode the relational information between the users. This kind of graph structure can be used to classify users into various categories or recommend new connections between them. In a citation network, papers are linked to each other via citations and the text of the papers as well as their links can be used to classify them into different categories. In biology and chemistry, proteins and chemical compounds can be represented as graphs, and protein-chemical compound affinity is modeled to discover new drugs. The non-euclidean structure of the graph data opens new challenges for Deep Neural Networks. As a graph can have an arbitrary topology, some of the standard operations (such as convolutions), which are easy to use in other domains, are difficult to apply on the graph-structured data. Moreover, unlike the assumption in other application domains that the instances are independent of each other, the instances in graph domain are related to each other via various type of links, such as undirected links in the social networks, directed links in the citation networks, or the atomic bonds between the atoms of a chemical compound. Based on these motivations, recently there is an increasing research focus on extending Deep Neural Networks for graph data, which is often termed as *Geometric Deep Learning*.

In this chapter, we start by describing various classes of graph-based learning in Section 4.1. In Section 4.2, we describe and compare various classes of Deep Neural Networks for graph-structured data. Further in Section 4.3 and Section 4.4, we discuss the proposed techniques for regularization and unsupervised/semi-supervised learning in graphs. Finally, in Section 4.5, we discuss the contribution of the proposed techniques and

future work.

4.1 Two Classes of Graph-Based Learning

Graph-based learning problems can be broadly categorized into two main classes:

4.1.1 Node Level Learning

In this kind of learning problem, given the graph structure and the input features for each node, the supervised learning task is node regression or node classification. For example, predicting the label of a document in the citation networks. In the unsupervised learning setup, the task is to learn the latent representations for each node, also known as learning the *network embeddings*. Note that the term *network embedding* is a misnomer in the sense that the objective here is the learn the embedding of an individual node, instead of the entire network (graph) Publication V of this thesis addresses the node classification problem.

4.1.2 Graph Level Learning

In this kind of learning problem, the task is to learn the class label or regression for the entire graph in a (semi-)supervised learning setting or to learn the latent representations for the entire graph in an unsupervised manner. Publication VI of this thesis addresses the problem of the unsupervised representation learning of the entire graph and predicting the label of the entire graph in a semi-supervised learning setup.

4.2 Deep Neural Networks for Graph Structured Data

With the recent surge of interest in Deep Neural Networks, there has been a rapid development of Neural Networks methods which are specifically designed to handle graph structured data. These methods are often called Graph Neural Networks (GNN). In the following, we categorize and discuss various forms of GNNs.

4.2.1 Recurrence Based Graph Neural Networks

The earliest work, in the year 1997, that proposed a neural network for structured data, more specifically for the directed acyclic or directed cyclic graph was by Sperduti et. al. [140]. In this work, the authors proposed a *generalized neuron*, which was used to compute the hidden states of a

node using the recurrent application of the same set of parameters on its *child nodes*. Following this, [129] proposed a general class of graph neural networks that can be applied to any kind of graph data, e.g. directed, undirected, cyclic, acyclic graphs. The central idea of their approach is to update a node's hidden states by exchanging the information from neighbourhood nodes recurrently until a stable equilibrium is reached. In the following, we formally define the hidden states update rule of this class of algorithms. Given a node i and its neighbours $N(i)$, the hidden state of i at iteration t is given as:

$$\mathbf{h}_i^{(t)} = \sum_{j \in N(i)} f_\theta(\mathbf{x}_i, \mathbf{x}_{i,j}^e, \mathbf{x}_j, \mathbf{h}_j^{(t-1)}) \tag{4.1}$$

where $f_\theta(.)$ is a recurrent function parameterized by θ, $\mathbf{x}_i$ is the input features of node i, $\mathbf{x}_{i,j}^e$ is the edge feature of edge connecting node i and j and $\mathbf{h}^{(0)}$ is initialized randomly. Note that for the hidden states of the nodes to converge to a stable point, function $f_\theta(.)$ should be a contractive mapping that shrinks the distance between two nodes when projected into the hidden space. Using such kinds of function, once the stable point of the hidden states is achieved, the methods of [129] use a readout layer followed by a classification layer. To avoid the limitation of using a contractive function, Gated Graph Neural Networks (GGNN) [94] use a Gated Recurrent Unit(GRU) as a recurrent function and the recurrence is applied for a fixed number of steps. Unlike [129], GGNN uses Back-Propagation Through Time (BPTT) to update the model parameters. The hidden state update rule for GGNN can be written as:

$$\mathbf{h}_i^{(t)} = \text{GRU}(\mathbf{h}_i^{(t-1)}, \sum_{j \in N(i)} \mathbf{W}\mathbf{h}_j^{(t-1)}) \tag{4.2}$$

where $\mathbf{W}$ is the linear transformation parameters. Note that the use of BPTT requires to save the intermediate hidden states (hidden states corresponding to each application of the recurrent operation) of all the nodes. This can be problematic for larger graphs. To avoid this problem, Stochastic Steady-state Embedding (SSE)[27] proposes to alternatively sample a set of nodes for state updates and a set of nodes for parameter updates.

4.2.2 Autoencoder Based Graph Neural Networks

Unlike the Recurrence based Graph Neural Networks of Section 4.2.1, which are mainly concerned about learning from a graph structure data in a supervised setting, Autoencoder based Graph Neural Networks addresses the problem of learning the representation of the graph data in an

unsupervised manner. Most of this line of research has been focused on learning the representations of the graph's nodes, however, these methods can be extended to learn the graph-level representations by applying *pooling* operations on the node representations.

One particular class of these algorithms is based on the *Network Homophily assumption*, which posits that the nodes which are similar to each other are more likely to attach to each other than the dissimilar one. This can be equivalently stated as: *if two nodes are connected in a graph, then they are more likely to be similar than the nodes which are not connected*.

Autoencoder based Graph Neural Networks [38, 23] which enforce such kinds of constraints in its learning framework , use an additional loss function (along with the reconstruction loss) that minimizes the distance between the representations of two connected nodes. Concretely, let us assume that an Autoencoder is composed of an encoder function Enc and a decoder function Dec, then such loss can be defined as:

$$\mathscr{L} = \sum_{(i,j)\in E} D(Enc(\mathbf{x}_i), Enc(\mathbf{x}_j)) \tag{4.3}$$

where $(i,j) \in E$ indicates that there exists an edge between nodes i and j, and D is a distance function, for example, euclidean norm or Kullback-Leibler Divergence(KLD).

Another class of Autoencoder based Graph Neural Networks use only the network structure for learning the node embeddings instead of using both the network structure and node's input features. In these approaches, the first step is to construct a matrix $\mathbf{S}$ such that an element $\mathbf{S}_{i,j}$ consists of the pair-wise proximity of the node pair (i,j) in the graph. Using this matrix, each node i is represented as the i-th row in the matrix: $\mathbf{s}_\mathrm{i} \in \mathbb{R}^{|\mathcal{V}|}$, where $|\mathcal{V}|$ represents the number of nodes in the graph. Essentially, $\mathbf{s}_\mathrm{i}$ represents the neighbourhood information for the node i in a vector form. The autoencoding objective of these approaches is to embed the vector $\mathbf{s}_\mathrm{i}$ into a low-dimensional vector $\mathbf{z}_\mathrm{i}$ such that the decoder can recover the original vector $\mathbf{s}_i$ from these embeddings as follows:

$$Dec(Enc(\mathbf{s}_i)) = Dec(\mathbf{z}_i) \approx \mathbf{s}_i \tag{4.4}$$

Using above kind of objective, the loss function of these approaches can be defined as:

$$\mathscr{L} = \|Dec(\mathbf{z}_i) - \mathbf{s}_\mathrm{i}\|_2^2 \tag{4.5}$$

These approaches mainly differ in the way they compute the pairwise proximity of a node pair. For example, Deep Neural Graph Representation (DNGR) [20] uses the pairwise mutual information of two nodes

co-occurring on random walks to construct $\mathbf{s}_i$, whereas, Structural Deep Network Embedding (SDNE)[167] uses the node adjacency information to construct $\mathbf{s}_i$.

Despite showing good predictive performance on the down-stream tasks, these approaches have two major limitations. First, the input dimension of the Autoencoder is $|\mathcal{V}|$, which can be intractable for graphs with millions of nodes. Second, these methods are inherently transductive in nature and can not generalize across graphs.

4.2.3 Convolution Based Graph Neural Network

Convolution based Graph Neural Networks (CGNN) share the same design principle as that of the Recurrence based Graph Neural Networks (RGNN): Both of these networks iteratively update a node's hidden states based on the hidden states of its neighbours. However, unlike RGNN which uses the same set of parameters iteratively, CGNN uses a fixed number of layers and a separate set of parameters for each layer. The layer in a CGNN is analogous to the convolution layer of the conventional CNNs for images: Similar to an image, where a convolutional layer updates the hidden states at a particular position in a 2D grid by taking the weighted average of the hidden states of the neighbouring positions, the convolutional layer in a CGNN updates the hidden states of a node by taking the weighted average of the hidden states of the node's neighbours. Various GCNN methods differ in the way they perform this weighted average operation.

Gilmer et.al.[41] proposed a general framework, called as Message Passing Neural Networks (MPNN), and shows that various existing GCNNs can be interpreted as an instance of the MPNN. MPNN performs K-steps of message passing to propagate the information between nodes. The message passing function can be formally defined as:

$$\mathbf{h}_i^{(k)} = g_k(\mathbf{h}_i^{(k-1)}, \sum_{j \in N(i)} f_k(\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}, \mathbf{x}_{i,j}^e)) \tag{4.6}$$

where $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ and $g_k(.)$ and $f_k(.)$ are functions with learnable parameters. After the K-th layer hidden states $\mathbf{h}_i^{(K)}$ have been derived, they can be passed to a output layer for performing node-level prediction tasks or they can be passed to a readout function to obtain graph-level representations. A readout function can be generally defined as :

$$\mathbf{h}_G = R(\mathbf{h}_i^{(K)} | i \in G) \tag{4.7}$$

Various CGNNs differ in the way they define $g(.)$, $f(.)$ and $R(.)$ functions [80, 161, 174, 34, 77]. Out of these various possible GCNNs, in Publication V, we have used GCN [80] and GAT [161] as an underlying model for the

proposed method GraphMix. In Publication VI, we have used Graph Isomorphism Network (GIN) [174] as an underlying model. In the following, we describe how these methods differ in the instantiation of the Equation 4.6. We made these choices to make a fair comparison of the methods presented in Publication V and Publication VI with the existing methods. In the following, we describe the hidden state update rule of GCN, GAT, and GIN in detail.

GCN assigns predetermined weights $a_{i,j}$ between neighbouring nodes i and j for updating the hidden states of node i using the Equation 4.6:

$$a_{i,j} = \frac{1}{\sqrt{deg(i) * deg(j)}} \tag{4.8}$$

where $deg(.)$ is the degree of a particular node.

GAT uses the attention weights $\alpha_{i,j}$ to measure the connection strength between two connected nodes i and j. The attention weights are computed as:

$$\alpha_{i,j} = softmax(e_{i,j}) = \frac{\exp(e_{i,j})}{\sum_{n \in N(i)} \exp(e_{i,n})} \tag{4.9}$$

where $\mathbf{e}_{i,j}$ is a scalar attention coefficient computed using a parameterized function Atten_ϕ as follows:

$$e_{i,j} = \text{Atten}(\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}) \tag{4.10}$$

There can be various ways to instantiate Atten_ϕ function in Equation 4.10. In [161], it is instantiated using a single layer network as follows:

$$e_{i,j} = \text{LeakyRelu}(\phi^T(\mathbf{W}^{(k)}\mathbf{h}_i^{(k-1)} || \mathbf{W}^{(k)}\mathbf{h}_j^{(k-1)})) \tag{4.11}$$

where $||$ represents the concatenation of two vectors.

Using the attention weights $\alpha_{i,j}$ computed as above, GAT updates the hidden states of the nodes as:

$$\mathbf{h}_i^{(k)} = \sigma(\sum_{j \in N(i) \cup i} \alpha_{i,j}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j^{(k-1)}) \tag{4.12}$$

Graph Isomorphism Networks (GIN) show that previous MPNN based methods were incapable of distinguishing between two graphs based on their graph-level representations. To solve this issue, GIN proposed to use a learnable parameters $\epsilon^{(k)}$ for adjusting the weights of the *central* node i as follows:

$$\mathbf{h}_i^{(k)} = \text{MLP}((1 + \epsilon^k)\mathbf{h}_i^{(k-1)} + \sum_{j \in N(i)} \mathbf{h}_j^{(k-1)}) \tag{4.13}$$

where MLP denotes a Multi-layer Perceptron.

In order to provide the necessary contextual information for interpreting the significance of the contributions made in Publication V and Publication VI, so far we have described various graph-structure based learning tasks and various forms of GNNs that are used to solve these tasks. In Section 4.3, we will discuss the problem of *generalization gap* in the node classification task and further we will summarize the proposed *GraphMix* method (Publication V) to address this problem. Following this, in Section 4.4 we discuss the unsupervised representation learning at the *graph-level* and the proposed method *InfoGraph* (Publication VI) to address this task.

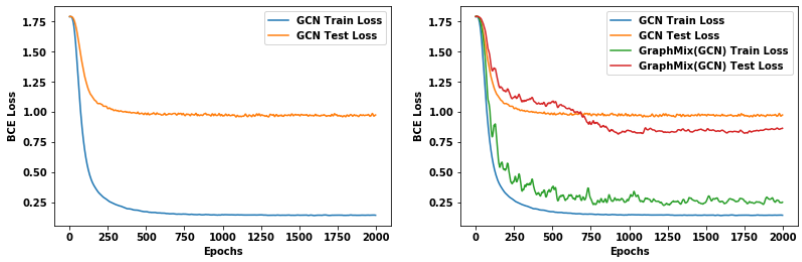
4.3 Regularizing Graph Neural Networks for Node classification

Similar to other Deep Neural Networks such as CNNs or RNNs, Graph Neural Networks also exhibit *generalization gap*. Some of the existing regularization techniques such as Dropout [141] and BatchNorm [72] can be readily applied to the Graph Neural Network layers to reduce this generalization gap, however, there still exists a substantial generalization gap. For example, in Figure 4.1a, we observe that the Train and Test loss differ significantly for a widely used Graph Convolutional Network(GCN) [80] trained with Dropout and BatchNorm for node classification task (node-level learning task Section 4.1.1.

Recently some methods have been proposed to improve the above mentioned generalization gap in Graph Neural Networks for the node classification tasks. In the following, we briefly mention some of these methods and their limitations, and further, we explain *GraphMix*, the method proposed in Publication V for improving the generalization of Graph Neural Networks. Refer to Figure 4.1b to observe how the GCN trained with GraphMix generalizes better than the normal training of GCN.

Some of the recent notable methods for improving the generalization gap for the node classification tasks include GraphSCAN [32] and BVAT [31] and graph adversarial training [35]. GraphSCAN[32] first uses an embedding method such as DeepWalk [116] and then trains generator-classifier networks in the adversarial learning setting to generate fake samples in the low-density region between sub-graphs. BVAT [31] and [35] generate adversarial perturbations to the features of the graph nodes while taking graph structure into account. While these methods improve generalization in graph-structured data, they introduce significant additional computation cost: GraphScan requires computing node embedding as a preprocessing step, BVAT and [35] require additional gradient computation for computing adversarial perturbations.

To circumvent the limitations of the above mentioned methods, we proposed *GraphMix* in (Publication V). In the following section, we describe



(a) GCN [80] **(b)** GraphMix(GCN) [165]

Figure 4.1. The gap between the train and test loss (known as the generalization gap) for the node classification task using the Citeseer citation network dataset. Figure 4.1a and Figure 4.1b shows this generalization gap for normal training of GCN and GraphMix trained GCN (GraphMix(GCN) in the plots), respectively. We observe that GraphMix(GCN) significantly reduces the generalization gap, however, there still exists scope for further improvements.

the motivation and the procedure of *GraphMix*.

4.3.1 GraphMix

Recent work based on interpolation-based regularizers [180, 162] has seen sizable improvements in generalization performance across a number of tasks. However, these techniques are not directly applicable to graphs for an important reason: Although we can create additional nodes by interpolating the features and corresponding labels, it remains unclear how these new nodes must be connected to the original nodes via synthetic edges such that the structure of the whole graph is preserved. To alleviate this issue, we propose to train an auxiliary Fully-Connected Network (FCN) using *Manifold Mixup* [162]. Note that the FCN only uses the node features (not the graph structure), thus the *Manifold Mixup* loss can be directly used for training the FCN.

Interpolation based regularization techniques have an added advantage for training GNNs. A vanilla GNN learns the representation of each node by iteratively aggregating information from the neighbors of that node (Equation 4.1 and 4.6). However, this induces the problem of *over-smoothing* while training GNNs with many layers [93, 175]. Due to this limitation, GNNs are trained only with a few layers, and thus they can only leverage the *local neighbourhood* of each node for learning its representations, without leveraging the representations of the nodes which are multiple hops away in the graph. This limitation can be addressed using the interpolation-based method such as *Manifold Mixup* : In *Manifold Mixup* , since the representations of a *randomly chosen pair* of nodes are used to facilitate better representation learning; it is possible that the randomly chosen pair of nodes will not be in the local neighbourhood of each other.

Furthermore, drawing inspiration from the success of self-training semi-supervised learning algorithms [164, 13], we explore self-training in the context of GNNs. Based on these challenges and motivations we summarize the procedure of GraphMix in the following.

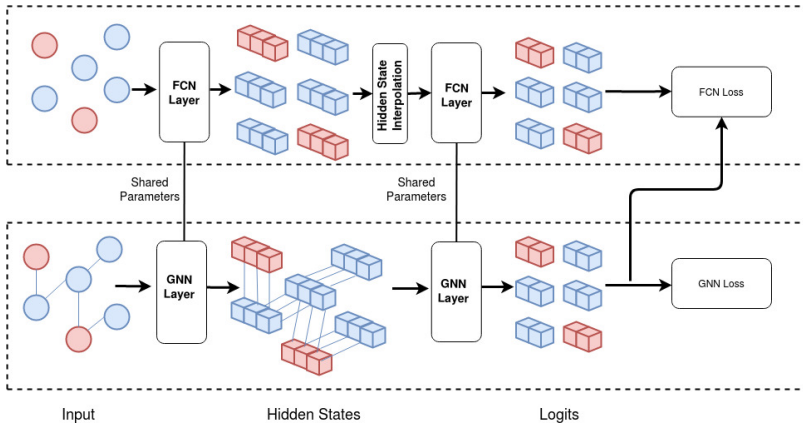


Figure 4.2. The procedure for training with GraphMix . The labeled and unlabeled nodes are shown with different colors in the graph. GraphMix augments the training of a baseline Graph Neural Network (GNN) with a Fully-Connected Network (FCN). The FCN is trained by interpolating the hidden states and the corresponding labels. This leads to better features that are transferred to the GNN by sharing the linear transformation parameters of the GNN and the FCN layers. Furthermore, the predictions made by the GNN for unlabeled data are used to augment the input data for the FCN. The FCN and the GNN losses are minimized jointly by alternate minimization.

GraphMix augments the vanilla GNN with a Fully-Connected Network (FCN). The FCN loss is computed using *Manifold Mixup* formulation (2.5) and the GNN loss is computed normally. *Manifold Mixup* training of FCN facilitates learning more discriminative node representations [162]. An important question is how these more discriminative node representations can be transferred to the GNN? One potential approach could involve maximizing the mutual information between the hidden states of the FCN and the GNN using formulations similar to those proposed by [63, 144]. However, this requires optimizing additional network parameters. Instead, we propose parameter sharing between the FCN and the GNN to facilitate the transfer of discriminative node representations from the FCN to the GNN. It is a viable option because for updating a node's hidden states, a GNN layer typically performs aggregation of the linear transformations of node representations (which are essentially pre-activation representations of the FCN layer).

Using the more discriminative node representations from the FCN, as well as the graph structure, the GNN loss is computed in the usual way to further refine the node representations. In this way, we can exploit the improved representations from *Manifold Mixup* training for training the GNNs.

Additionally, we propose to use the predicted targets from the GNN to augment the training set of the FCN. In this way, both the FCN and the GNN facilitate each other's learning process. Both the FCN loss and the GNN loss are optimized in an alternating fashion during training. At inference time, predictions are made using only the GNN.

The GraphMix procedure is highly flexible: it can be applied to any underlying GNN as long as the underlying GNN applies parametric transformations to the node features. In our experiments, we show the improvements over GCN [81], GAT [161] and Graph U-Nets [39] using GraphMix . A diagram illustrating GraphMix is presented in Figure 4.2. For more technical discussion of the method and experimental results, refer to Publication V.

4.4 Graph Level Unsupervised and Semi-supervised Learning

Graph level learning, described in Section 4.1.2, is an important problem because of its direct application to a wide variety of domains. For example, predicting the chemical properties and bioactivity of the molecules using their graph representations [41], activity recognition in videos using the graph representations of the key points (joints) of a human body [74] or predicting the interface between two proteins using their graph representations (each amino acid residue is represented as a node in the protein graph) [36]. While there has been substantial research work for supervised graph-level learning [181, 176, 138, 174, 53], unsupervised and semi-supervised graph-level learning remains under-explored. Most notable examples include graph2Vec [107] and *Pretrainig Strategies for Graph Neural Networks* [69]. Graph2Vec extracts the subgraphs from a given graph and applies the *context prediction or skip-gram* principle similar to word2vec [101] to learn graph-level representations. Graph2Vec has a major limitation that it only considers the local structure of the graphs instead of considering the global structure of the graphs. *Pretraining Strategies for Graph Neural Networks* propose to combine the node-level pretraining and graph-level pretraining. For graph-level pretraining, they consider two strategies: (1) *Graph-level multitask supervised pretraining*. For example, for molecular property prediction, one can train a GNNs to jointly predict all the properties that have been measured so far and further fine-tune the GNNs to predict the property for which relatively less labeled data is available. However, this kind of pretraining may not be applicable in many application domains because of lack of joint prediction tasks. (2) *Predicting the similarity between two graphs*. Examples of such tasks include predicting the graph edit distance or predicting the structural similarity of two graphs. However, computing ground-truth labels for such kinds of tasks is a difficult problem and there are a quadratic number of graph pairs to consider.

Keeping in mind the limitations of the above mentioned methods, we take a rather different approach for the unsupervised graph-level representation learning. We propose *InfoGraph* method in Publication VI and further extended this for the graph-level semi-supervised learning, denoted as *InfoGraph**. In the following, we explain these methods on a conceptual and mathematical level. For detailed discussions and results, refer to Publication VI.

4.4.1 InfoGraph

We start with formally defining the problem setup for the graph-level unsupervised representation learning and graph-level semi-supervised learning:

- **Graph-level Unsupervised Representation Learning:** Given a set of graphs $\mathbb{G} = \{G_1, G_2, ...\}$ and a positive integer d (the expected embedding size), our goal is to learn a d-dimensional distributed representation of every graph $G_i \in \mathbb{G}$. We denote the number of nodes in G_i as $|G_i|$. We denote the matrix of representations of all graphs as $\Phi \in \mathbb{R}^{|\mathbb{G}| \times d}$.

- **Graph-level Semi-supervised Learning:** Given a set of labeled graphs $\mathbb{G}^L = \{G_1, \cdots, G_{|\mathbb{G}^L|}\}$ with corresponding output $\{o_1, \cdots, o_{|\mathbb{G}^L|}\}$, and a set of unlabeled samples $\mathbb{G}^U = \{G_{|\mathbb{G}^L|+1}, \cdots, G_{|\mathbb{G}^L|+|\mathbb{G}^U|}\}$, our goal is to learn a model that can make predictions for unseen graphs. Note that in most cases $|\mathbb{G}^U| \gg |\mathbb{G}^L|$.

InfoGraph is motivated by Deep InfoMax [63]. Specifically, to learn the graph-level representations in an unsupervised manner, we propose to maximize the mutual information between the representations of the entire graph and the representations of its substructures of different granularity. By doing so, the graph representations can learn to encode aspects of the data that are shared across all substructures.

Assume that we are given a graph $G \in \mathbb{G}$ with $|G|$ nodes. Let ϕ denote the set of parameters of a K-layer graph neural network. After the first k layers of the graph neural network, each node i of the graph G will be represented as $\mathbf{h}_i^{(k)}$ (for example, by using Equation 4.6). Next, we summarize the representation of a node i at *all depths* of the graph neural network into a single representation vector. This single representation vector of node i captures patch information at different scales centered at (local to) node i. Following [175], we use concatenation for this summarization. That is,

$$\mathbf{h}_\phi^i = \text{CONCAT}(\{\mathbf{h}_i^{(k)}\}_{k=1}^K) \tag{4.14}$$

where $\mathbf{h}_\phi^i$ is the summarized patch representation centered at node i . Note that in Equation 4.14 we slightly abuse the notation of $\mathbf{h}_\phi^i$ for the

sake of clarity.

Further, we use a READOUT function to combine the representations of nodes into one feature vector to get graph-level representation:

$$\mathbf{H}_\phi(G) = \text{READOUT}(\{\mathbf{h}_\phi^i\}_{i=1}^{|G|}) \tag{4.15}$$

We define our mutual information (MI) estimator on global/local pairs of above mentioned representations, maximizing the estimated MI over the given dataset $\mathbf{G} := \{G_j \in \mathbb{G}\}_{j=1}^N$ as following:

$$\hat{\phi}, \hat{\psi} = \arg\max_{\phi,\psi} \sum_{G \in \mathbf{G}} \frac{1}{|G|} \sum_{u \in G} I_{\phi,\psi}(\mathbf{h}_\phi^u; \mathbf{H}_\phi(G)). \tag{4.16}$$

$I_{\phi,\psi}$ is the mutual information estimator modeled by discriminator T_ψ and parameterized by a neural network with parameters ψ. We use the Jensen-Shannon MI estimator (following the formulation of [110]),

$$I_{\phi,\psi}(\mathbf{h}_\phi^i(G); \mathbf{H}_\phi(G)) := \\ \mathbb{E}_{\mathbb{P}}[-\text{sp}(-T_{\phi,\psi}(\mathbf{h}_\phi^i(\mathbf{x}), \mathbf{H}_\phi(\mathbf{x})))] - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[\text{sp}(T_{\phi,\psi}(\mathbf{h}_\phi^i(\mathbf{x}'), \mathbf{H}_\phi(\mathbf{x})))] \tag{4.17}$$

where $\mathbf{x}$ is an input sample, $\mathbf{x}'$ (negative sample) is an input sampled from $\tilde{\mathbb{P}} = \mathbb{P}$, a distribution identical to the empirical probability distribution of the input space, and $\text{sp}(\mathbf{z}) = \log(1 + e^{\mathbf{z}})$ is the softplus function. In practice, we generate negative samples using all possible combinations of global and local patch representations across all graph instances in a batch.

Since $\mathbf{H}_\phi(G)$ is encouraged to have high MI with patches that contain information at all scales, this favours encoding aspects of the data that are shared across patches and aspects that are shared across scales. The algorithm is illustrated in Fig. 4.3.

Based on the previous unsupervised model, a straightforward way to do semi-supervised property prediction on graphs is to combine the purely supervised loss and the unsupervised objective function which acts as a regularization term. In doing so, the model is trained to predict properties for the labeled dataset while keeping a rich discriminative intermediate representation learned from both the labeled and the unlabeled dataset. That is, we try to minimize the following objective function:

$$L_{\text{total}} = \sum_{i=1}^{|\mathbb{G}^L|} L_{\text{supervised}}(\mathbf{y}_\phi(G_i), \mathbf{o}_i) + \lambda \sum_{j=1}^{|\mathbb{G}^L|+|\mathbb{G}^U|} L_{\text{unsupervised}}(\mathbf{h}_\phi(G_j); \mathbf{H}_\phi(G_j)) \tag{4.18}$$

where $L_{\text{supervised}}(\mathbf{y}_\phi(G_i), \mathbf{o}_i)$ is defined as the loss function of graph G_i that measures the discrepancy between the classifier output $\mathbf{y}_\phi(G_i)$ and the

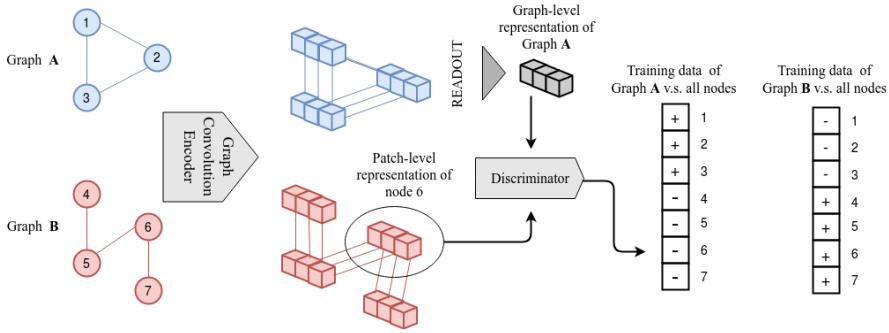


Figure 4.3. Illustration of InfoGraph. An input graph is encoded into a feature map by graph convolutions. The discriminator takes a (global representation, patch representation) pair as input and decides whether they are from the same graph. InfoGraph uses a batch-wise fashion to generate all possible positive and negative samples. For example, consider the toy example with 2 input graphs in the batch and 7 nodes (or patch representations) in total. For the global representation of the blue graph, there will be 7 input pairs to the discriminator and same for the red graph. Thus, the discriminator will take 14 (global representation, patch representation) pairs as input in this case.

true output o_i. $L_{\text{unsupervised}}(\mathbf{h}_\phi(G_j);\mathbf{H}_\phi(G_j)$ is the unsupervised InfoGraph loss term as defined in eq. (4.16) that can be optimized using both labeled and unlabeled data. The hyper-parameter λ controls the relative weight between the purely supervised and the unsupervised loss. The intuition behind this is that the model will benefit from learning a good representation from the large amount of unlabeled data while learning to predict the corresponding supervised label.

However, supervised tasks and unsupervised tasks may favor different information or a different semantic space. Simply combining the two loss functions using the same encoder may lead to "negative transfer" [112, 124]. We propose a simple way to alleviate this problem: we deploy two encoder models: the encoder on the labelled data (supervised encoder) and the encoder on the unlabelled data (unsupervised encoder). For transferring the learned representations from the unsupervised encoder to the supervised encoder, we define a loss term that encourages the representations learned by the two encoders to have high mutual information, *at all levels of representations* (third term of Eq. 4.19). Formally, let φ denote the set of parameters of another K-layered graph neural network, identical to the one parameterized by ϕ, and let λ be a tunable hyper-parameter, $\mathbf{H}^k_\phi(G)$, $\mathbf{H}^k_\varphi(G)$ be global encoder representations of the graph G at encoder layer k,

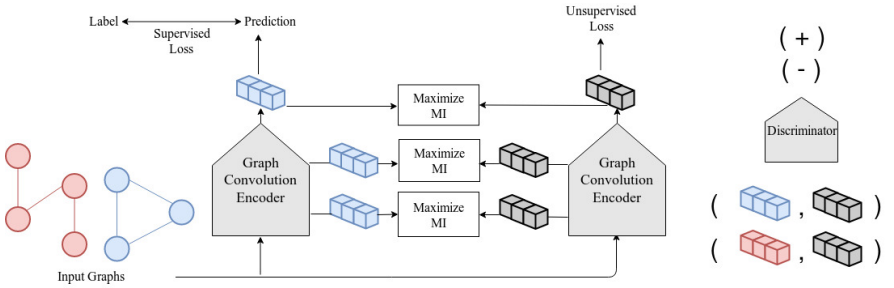


Figure 4.4. Illustration of the semi-supervised version of InfoGraph (InfoGraph*). There are two separate encoders with the same architecture, one for the supervised task and the other trained using both labeled and unlabeled data with an unsupervised objective. We encourage the mutual information of the two representations learned by the two encoders to be high by deploying a discriminator that takes a pair of representations as input and determines whether they are from the same input graph. Figure reproduced from Publication VI. Figure reproduced from Publication VI.

then total loss function can be defined as follows:

$$L_{\text{total}} = \sum_{i=1}^{|\mathbb{G}^L|} L_{\text{supervised}}(\mathbf{y}_\phi(G_i), \mathbf{o}_i) + \sum_{j=1}^{|\mathbb{G}^L|+|\mathbb{G}^U|} L_{\text{unsupervised}}(\mathbf{h}_\varphi(G_j); \mathbf{H}_\varphi(G_j)) - \lambda \sum_{j=1}^{|\mathbb{G}^L|+|\mathbb{G}^U|} \frac{1}{|G_j|} \sum_{k=1}^{K} I(\mathbf{H}_\phi^k(G_j); \mathbf{H}_\varphi^k(G_j)). \tag{4.19}$$

Notice that this formulation can be seen as a special instance of the *student-teacher* framework. However, unlike the recent *student-teacher* methods for semi-supervised learning [87, 149, 164], which enforce the predictions of the student model to be similar to the teacher model, we enforce the transfer of knowledge from the teacher model to the student model via mutual-information maximization at various levels of representations. In practice, to reduce the computation overhead introduced by the third term of Eq 4.19, instead of enforcing the mutual-information maximization over all the layers of the encoders, at each training update, we enforce mutual-information maximization on a randomly chosen layer of the encoder [162]. We refer to this method as *InfoGraph**, which is fully summarized in Figure 4.4.

4.5 Discussions and Future Directions

While there has been substantial progress in recent years addressing the node-level and the graph-level tasks in the fully-supervised learning setup, progress in addressing these tasks in the unsupervised and the

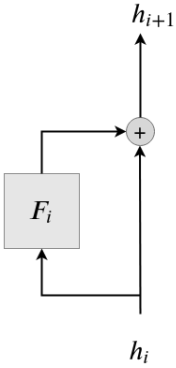
semi-supervised paradigms remains slow in comparison to other domains. One potential reason for this slow progress is that unlike other domains such as images, where there are many natural choices for designing self-supervised tasks or designing data-augmentation techniques, there are no such natural choices for graph-structured data. For instance, many of the current state-of-the-art methods for unsupervised representation learning methods for images require solving domain-specific self-supervision e.g. colorization [183], placing image patches in correct order [109], inpainting [115], among others. Similarly, the recent success of semi-supervised methods for image classification heavily depends on domain-specific data-augmentation techniques such as random cropping, horizontal flipping, etc [13, 12, 171].

This suggests that designing data-augmentation techniques and self-supervision tasks for the graph structured data is an important area of research. To this end, in Publication V, we proposed *GraphMix*, a method for adaptation of *Manifold Mixup* (which can be interpreted as a data augmentation technique) [162] for the graph structured data. For the future work, one may consider to jointly model the node features and edges of the graph in such a way that it can be used for generating the synthetic interpolated nodes and their corresponding connectivity to the other nodes in the graph. This will alleviate the need to train the auxiliary FCN in *GraphMix*. In Publication VI, we explored the application of Mutual Information Maximization as an unsupervised and semi-supervised objective for graph-level learning. In the future, one may consider further extending the framework of Publication VI by mixing the graph-level hidden representations and corresponding predicted-targets, similar to *Manifold Mixup* procedure.

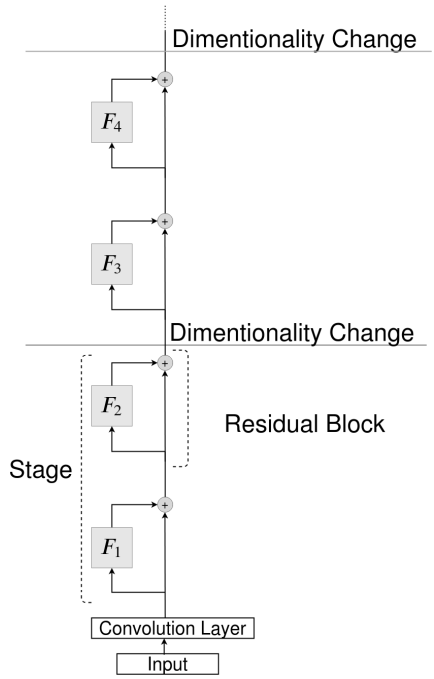
5. Understanding Residual Networks and Techniques for their Improved Training

By the early 2010s, deep convolutional neural networks had already revolutionized computer vision [84, 130]. The primary reason for this success was learning the hierarchy of useful representation and the classifier in an end-to-end manner, instead of using the hand-crafted features. Using the hand crafted features (such as HOG [29], SHIFT[97], LBP[111]) was the dominant approach in computer vision research prior to the resurgence of DNNs. Following the early success of deep convolutional networks, the follow-up works have shown the depth of the convolutional neural networks is crucial for achieving state-of-the-art performance on computer vision tasks, such as image recognition, object detection and segmentation [147, 121, 133]. Motivated by the significance of the depth, one obvious question to ask is whether better networks (networks that generalize better) can be learned by just stacking more layers? To answer this question we need to answer a related question of whether it is possible to train deeper networks to sufficiently small train loss? It was not possible earlier to train deeper neural networks to sufficiently small train loss due to the problem of vanishing and exploding gradients [64, 9]. Some recent techniques such as normalized initialization of parameters [42] and intermediate layer representation normalization [72, 3], have largely addressed these issues.

Nevertheless, even if we can train deeper convolutional neural networks to sufficiently small train loss, it is not possible to get improved generalization by simply stacking more layers, due to another problem known as *degradation problem*: with the increasing depth of the neural network, the test error saturates and then degrades rapidly. This degradation does not happen because of the overfitting (as one might suspect), rather it happens because of the increase in the train loss [61]! This is a surprising property. Why so? To understand this, let us consider a shallower network and its deeper counterpart such that all the layers from the shallower networks are copied to the deep network and the remaining layers of the deeper network are identity functions. By construction, there exists a solution for a deeper network such that it will have the training error no



(a) A residual block that converts representation h_i into h_{i+1}(Equation 5.1)



(b) A typical residual network architecture. Green line shows the change of dimensionality of hidden states.

Figure 5.1. An illustration of a ResNets. F_i represents a Residual module which consists of Convolution, BatchNorm and non-linear activation layers.

more than that of the shallower network. This increase in training loss while increasing the depth indicates that there exists some fundamental limitation in the architectures of the deep neural networks that makes them harder to train.

To overcome the *degradation problem*, Residual Networks (ResNets) [57] proposed to learn the *residual mappings* instead of the *original* desired mappings. More specifically, let us assume that we want a stack of layers to learn an *original* mapping $\mathscr{H}(\mathbf{h})$, then $\mathscr{H}(\mathbf{h})$ can be recasted as

$$\mathscr{H}(\mathbf{h}) := \mathscr{F}(\mathbf{h}) + \mathbf{h} \tag{5.1}$$

where $\mathscr{F}(\mathbf{h})$ is the residual mapping. Instead of hoping that a stack of layers could directly fit the original underlying mapping $\mathscr{H}(\mathbf{h})$, ResNets explicitly let this stack of layer to fit only the residual mapping $\mathscr{F}(\mathbf{h})$. The main hypothesis behind this formulation is that for a stack of layers, it is easier to learn residual mappings than the original underlying mappings.

In ResNets [57], the formulation of $\mathscr{F}(\mathbf{h}) + \mathbf{h}$ is realized using *skip connections* [16]. The skip connections in ResNets only perform the identity mapping and their output is added to the output of the stacked layers. Figure 5.1 shows a simplistic version of a typical ResNet.

Since their inception in the year 2016, ResNets have become a *defacto* choice of architectures for the computer vision tasks. Following their popularity, there have been several attempts to interpret and understand their working mechanism which may lead to designing a better way of training these networks.

In the following, we present various interpretations of ResNets and methods for their improved training. In Section 5.2, we briefly highlight how Publication VII builds on and differs from the previous interpretations of ResNets. (We refer the readers to Publication VII for detailed discussions and results.)

5.1 Residual Networks: Various Interpretations

In this section, we present various representations of the ResNets and how these interpretations have led to designing better training methods for ResNets.

5.1.1 Unraveled and Ensemble view

Veit et.al.[160] introduced an *unraveled* view of ResNets. This view illustrates that instead of a single deep network, a ResNet can be viewed as a collection of many paths through which data flows. Here a path refers to a unique combination of which residual blocks to enter and which to skip (i.e. pass through the identity connection) while the data flows in the network. Using this view, in a ResNet with n residual blocks, there can be 2^n unique paths. Further Veit et. al.[160], conducted various ablation studies to understand the behaviour of ResNets:

- By dropping an individual residual block at the test time, ResNets do not suffer any significant drop in the test accuracy. This is unlike other non-ResNet architectures (such as VGG), where deleting a single layer at test time reduces the accuracy to the chance level. This difference in the behavior of ResNets and VGG can be interpreted using the unraveled view of ResNets: When a residual block is removed, the number of paths is reduced to 2^{n-1} from 2^n, leaving half the number of paths still valid. On the contrary, VGG contains only a single path form the input to the output, so when a layer is removed, the only viable path is corrupted.

  This suggests that paths in the ResNets do not strongly depend on each other even though they are trained jointly.

- Since in the unraveled view of ResNets, they are interpreted as a collection of many paths, it is a natural question to ask whether ResNets behave like an ensemble of these paths. One of the key characteristics

of the ensembles is that their performance smoothly correlates with the number of members in the ensemble. Thus if the ResNets behave like an ensemble, their test-time performance will co-relate with the number of valid paths. To test this hypothesis, Veit et.al.[160] deleted an increasing number of residual blocks and observed that the test error increased smoothly. This implies that ResNets act as an ensemble of many paths.

- Another interesting question to investigate is whether all the gradient back propagates equally through all of these paths or not. To measure this, Veit et.al.[160] sampled individual paths of certain length and computed the norm of the gradient that arrives at the input from these paths. The experiments revealed that the gradient magnitude of a path decreases exponentially w.r.t its length. This gives rise to another question: whether shorter or longer paths contribute more gradient magnitude during training? To find the total gradient magnitude contributed by the paths of the certain length, Veit et.al.[160], multiplied the frequency of each path length with the expected gradient magnitude. The results show that most of the gradient comes from the paths of the *relatively shallower* length. For example, for a ResNet with 54 residual blocks, almost all the gradient during training comes from the paths which are between 5 and 17 blocks long. This implies that it is not necessary to use all the paths for training a ResNet. Using this insight, one can design better training strategies for the ResNets.

This interpretation can be used to justify some of the later variations of ResNets. For example, Stochastic Depth Networks [71], which randomly drops some of the residual blocks during the forward and backward pass, can be justified from the observations described above that a ResNet can be trained using *relatively shallower paths*. Other examples include DenseNet [70] and ResNext [172]. These networks are similar to ResNets in the spirit; both of these networks employ skip connections between layers. However there are few important differences. DenseNet employs skip connections between the layers of a *Dense block*. This leads to substantially more number of skip connections and hence more number of paths from the input to the output. ResNext has multi-branch residual blocks, which also increases the number of paths from the input to the output. The improved test accuracy of DenseNet and ResNext in comparison to the ResNet can be explained using the ensemble view described above: since there are more paths (members) in the ensemble, these networks have better test accuracy.

5.1.2 Dynamical system view

ResNets have been also interpreted from the dynamical system view [49, 50]. The residual block forward pass of a ResNet (Equation 5.1) can be written using a layer index i as:

$$\mathbf{h}_{i+1} = \mathbf{h}_i + \mathscr{F}(\mathbf{h}_i, \theta_i) \tag{5.2}$$

Without the loss of the generality, using an additional parameter k, the residual module $\mathscr{F}(.)$ in Equation 5.2 can be written as $\mathscr{F}(.) = k\mathscr{G}(.)$, and the residual block becomes :

$$\mathbf{h}_{i+1} = \mathbf{h}_i + k\mathscr{G}(\mathbf{h}_i, \theta_i) \tag{5.3}$$

Rearranging the terms in Equation 5.3:

$$\frac{\mathbf{h}_{i+1} - \mathbf{h}_i}{k} = \mathscr{G}(\mathbf{h}_i, \theta_i) \tag{5.4}$$

The new parameter k is called the step size for discretization. In the original formulation of the ResNets (Equation 5.3), step size k does not exist explicitly, rather it is absorbed by the residual module $\mathscr{F}(.,.)$.

For a sufficiently small k, Equation 5.4 can be considered as a discretization of the initial value Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \mathscr{G}(\mathbf{h}(t), \theta(t)), \quad for \quad 0 \le t \le T, \quad \mathbf{h}(0) = \mathbf{h}_0 \tag{5.5}$$

where t corresponds to the direction from input to the output, $\mathbf{h}(0)$ is the input feature map and $\mathbf{h}(T)$ is the output feature map before the softmax classifier. Thus the problem of optimizing a ResNet's parameters θ becomes equivalent to the parameter estimation problem involving the ODE in Equation 5.5.

Given the connections between the ResNets and the ODE, the existing numerical techniques for solving the ODEs can be applied to ResNets. For example, the multigrid technique [51], which is a technique for solving the differential equations using a hierarchy of discretization with varying step-size k, can be used to learn the parameters of a ResNet. Along these lines, [22] proposed the *multilevel training* of ResNets. The main idea of this approach is to start with shallower networks and large step-size k. After a few training steps, the number of residual blocks is doubled by inserting a residual block before and after each existing residual block, and the step-size is reduced to half. This process is repeated multiple times. Since this training method does not train the full-depth ResNet throughout the training process, rather the depth of the ResNet is increased gradually, [22] experimentally shows that multi-level training can reduce the training-time of ResNets to approximately half of its original value. .

5.1.3 Recurrent Network View

ResNets with shared parameters across all the layers is equivalent to a shallow Recurrent Neural Network (RNN). This equivalence can be seen by dropping the per-layer parameters in Equation 5.1 and rewriting it as:

$$\mathbf{h}_{i+1} = \mathbf{K} \odot (\mathbf{h}_i) + \mathbf{h}_i \tag{5.6}$$

where $\mathbf{K}$ is a parameterized operator. Such an equation corresponds to the Recurrent Neural Network of Figure 5.2.

Using this equivalence, [95] trains ResNets which have shared parameters for the consecutive residual blocks of the same input and output dimensions, and demonstrates that the performance of these networks is similar to the ResNets without parameter sharing. This sharing of parameters helps training ResNets which have an order of magnitude less parameters than the non-shared networks, thus achieving better memory efficiency.

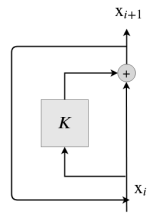


Figure 5.2. Residual Network as a Recurrent Neural Network.

5.1.4 Unrolled Iterative Estimation View

The success of the erstwhile non-residual Deep Neural Networks (such as AlexNet [84]) is often attributed to the learning hierarchy of increasingly abstract representations [91]. In this view, each layer of a DNN learns a particular level of representation. Contrary to this view, [46] presents the *unrolled iterative estimation / refinement view* for ResNets: Instead of learning an entirely new level of representations, residual blocks in the same stage work together to estimate and iteratively refine the same level of representation. For example, if the first residual block in a stage detects simple shapes then the rest of the residual blocks in that stage work at that level too. The transition to the next level of representation happens from one stage to the next one, i.e. when the dimensionality of representations changes (refer to Figure 5.1b).

This view explains the legion studies conducted in [160], where removing a residual block from the ResNet at inference time does not lead to a significant drop in the performance: since the residual blocks in the same stage iteratively refine the representations, removing one of these residual blocks does not critically disrupt the input distribution of the following layers, thus there is negligible effect on the performance.

5.2 Deeper Understanding of Iterative Estimation View

In Publication VII, we set out to attain a deeper understanding of the iterative refinement view. Most importantly, we formalize the notion of iterative estimation in ResNets and study the properties of representations that residual blocks tend to learn, as a result of being additive in nature, in contrast to traditional compositional networks.

Specifically, let us consider ResNet architectures (see figure 5.1b) where the first hidden layer is a convolution layer, which is followed by L residual blocks. A residual block transforms a representation $\mathbf{h}_i$ as,

$$\mathbf{h}_{i+1} = \mathbf{h}_i + F_i(\mathbf{h}_i) \tag{5.7}$$

Consider L such residual blocks stacked on top of each other followed by a loss function. Then, we can Taylor expand any given loss function $\mathscr{L}$ recursively as,

$$\mathscr{L}(\mathbf{h}_L) = \mathscr{L}(\mathbf{h}_{L-1} + F_{L-1}(\mathbf{h}_{L-1})) \tag{5.8}$$

$$= \mathscr{L}(\mathbf{h}_{L-1}) + F_{L-1}(\mathbf{h}_{L-1}).\frac{\partial \mathscr{L}(\mathbf{h}_{L-1})}{\partial \mathbf{h}_{L-1}} \tag{5.9}$$

$$+ \mathscr{O}(F^2_{L-1}(\mathbf{h}_{L-1}))$$

Here we have Taylor expanded the loss function around $\mathbf{h}_{L-1}$. We can similarly expand the loss function recursively around $\mathbf{h}_{L-2}$ and so on until $\mathbf{h}_i$ and get,

$$\mathscr{L}(\mathbf{h}_L) = \mathscr{L}(\mathbf{h}_i) + \sum_{j=i}^{L-1} F_j(\mathbf{h}_j).\frac{\partial \mathscr{L}(\mathbf{h}_j)}{\partial \mathbf{h}_j} + \mathscr{O}(F^2_j(\mathbf{h}_j)) \tag{5.10}$$

Notice we have explicitly only written the first order terms of each expansion. The rest of the terms are absorbed in the higher order terms $\mathscr{O}(.)$. Further, the first order term is a good approximation when the magnitude of F_j is small enough. In other cases, the higher order terms come into effect as well.

Thus in part, the loss equivalently minimizes the dot product between $F(\mathbf{h}_i)$ and $\frac{\partial \mathscr{L}(\mathbf{h}_i)}{\partial \mathbf{h}_i}$, which can be achieved by making $F(\mathbf{h}_i)$ point in the opposite half space to that of $\frac{\partial \mathscr{L}(\mathbf{h}_i)}{\partial \mathbf{h}_i}$. In other words, $\mathbf{h}_i + F(\mathbf{h}_i)$ approximately moves $\mathbf{h}_i$ in the same half space as that of $-\frac{\partial \mathscr{L}(\mathbf{h}_i)}{\partial \mathbf{h}_i}$. The overall training criteria can then be seen as approximately minimizing the dot product between these two terms along a path in the $\mathbf{h}$ space between $\mathbf{h}_i$ and $\mathbf{h}_L$ such that loss gradually reduces as we take steps from $\mathbf{h}_i$ to $\mathbf{h}_L$. Using this analysis we formalize iterative inference in Resnets as moving down the energy (loss) surface.

To empirically confirm the above formalization, we compute the cosine loss $\frac{F_i(\mathbf{h}_i).\frac{\partial \mathscr{L}(\mathbf{h}_i)}{\partial \mathbf{h}_i}}{\|F_i(\mathbf{h}_i)\|_2 \|\frac{\partial \mathscr{L}(\mathbf{h}_i)}{\partial \mathbf{h}_i}\|_2}$ and $\|F_i(.)\|_2$ A negative cosine loss and small $\|F_i(.)\|_2$

together suggest that $F_i(.)$ is refining features by moving them in the half space of $-\frac{\partial \mathscr{L}(\mathbf{h}_i)}{\partial \mathbf{h}_i}$, thus reducing the loss value for the corresponding data samples.

So far we have formally defined how residual blocks perform iterative refinement of features. In addition to this, in Publication VII, we also investigate the behaviour of individual residual blocks of a ResNet from the viewpoint of iterative refinement vs representation learning. Precisely, we investigate whether all the residual blocks perform a similar degree of refinement or some of them perform representation learning (meaning to make the output of the residual block significantly different from its input). We empirically observed that lower residual blocks learn to perform representation learning, meaning that they change representations significantly and removing these blocks can sometimes drastically hurt prediction performance. The higher blocks on the other hand essentially learn to perform iterative inference– minimizing the loss function by moving the hidden representation along the negative gradient direction.

The iterative refinement view suggests that deep networks can potentially leverage intensive parameter sharing for the layer performing iterative inference. But sharing a large number of residual blocks without loss of performance has not been successfully achieved yet. (Note that [95] shared parameters for only a few numbers of residual blocks.) Towards this end, in Publication Publication VII we propose improved training of ResNets by reusing residual blocks in two ways: 1. Sharing residual blocks during training: We find that training ResNet with naively shared blocks leads to bad performance. We expose reasons for this failure and investigate a preliminary fix for this problem. 2. Unrolling a residual block for more steps at inference time than that it was trained to unroll: We observed that the performance of a ResNet can be improved using this technique.

5.3 Discussion

In this chapter, we presented various interpretations of ResNets and how these interpretations have been used to design better training methods for ResNets. Further, we highlighted the major contributions made in Publication VII towards understanding and improving the training of the ResNets.

Due to the wide-spread use of the ResNets in computer vision applications, understanding and improving their training is an important research direction. While we have taken initial steps for using the notion of iterative refinement for improving the training of ResNets, various other hypotheses for iterative inference in the human brain are studied extensively in neuroscience research [151, 157]. In the future, one important direction

could be to leverage these hypotheses to design better training strategies for the existing ResNets.

6. Discussion

Deep Neural Networks have achieved state-of-the-art performance in many machine learning tasks. These tasks include but are not limited to, speech recognition [54], image recognition [57, 84, 60], natural language understanding [159], robotics [104], medical diagnosis [134]. Across all these tasks, Deep Neural Networks significantly outperform other conventional models and approaches.

Based on the success of Deep Neural Network on academic benchmarks and tasks, it has rapidly found its way into the commercial applications through companies such as Google, Facebook, Microsoft, Apple, and Salesforce, etc. Some of the notable examples of these commercial applications that are used by millions of customers on a daily basis include speech recognition systems such as Apple Siri and Google Voice, text translation systems such as Google translate and image recognition systems used in Google Street view and Facebook photo tagging applications.

The resurgence of interest in Deep Neural Networks started with the seminal papers on layer-wise pretraining of Deep Neural Networks [62, 11]. Since then there have been many notable breakthroughs that have further pushed the boundaries of the performance of Deep Neural Networks. For instance, novel architectures such as ResNets [57] have been proposed that facilitated training Deep Neural Networks with even thousands of layers for the first time. For generative modeling, architectures such as Generative Adversarial Networks [45] and Variational Autoencoders [78] have been proposed which have become the foundation of the current state-of-the-art generative modeling methods. Recently proposed *attention* based architectures such as Transformers [159] have achieved breakthroughs in language modeling. To address the problem of *saturation* in the previous non-linear activation units, novel activation units such as ReLU [43] have been proposed. Besides architecture and non-linearities, breakthroughs have been also made for regularization (such as Dropout [142]) and for faster convergence (such as BatchNorm [72]) of Deep Neural Networks.

Despite its indisputable success in various domains, the application of Deep Neural Networks is still largely limited by the availability of a large

amount of annotated data. This is a prohibitive constraint especially in the domains where the annotated data is fundamentally limited due to the nature of the domain, such as drug discovery. Designing Deep Neural Network algorithms and models that can generalize well with a limited amount of annotated data is an important and ongoing research direction. To this end, in this dissertation, I have proposed methods that improve the generalization of Deep Neural Networks in supervised, unsupervised, semi-supervised, adversarial learning and graph-based learning settings. The proposed methods in this thesis have achieved state-of-the-art in aforementioned learning settings and have inspired many follow-up works. These methods are particularly appealing for the industrial applications of Deep Neural Networks because of their simplicity, no computation overhead and applicability to a wide variety of architectures.

In the following sections, I will conclude this write-up by mentioning the main areas in which the follow-up work can be deployed.

6.1 Mixing based Synthetic samples for Reinforcement Learning

A widely discussed challenge in reinforcement learning is poor sample efficiency as well as the difficulty of collecting data. One reason why collecting data can be difficult is that any system which runs in the real world can only run "at real time", which increases the cost of data collection. Another challenge is that if the system needs to interact with humans, then human time or attention may be expensive to get (depending on the system). Still another challenge is that some events are very rare and may occur occasionally, if at all (for example many car crashes are highly unique and may only occur a few times). This is part of the broader problem of sparse rewards.

The *mixing* based learning techniques discussed in this thesis could play an important role in improving the sample complexity and thus the efficiency of reinforcement learning. The most straightforward project in this space would be to improve the sample efficiency of imitation learning using a *Manifold Mixup* kind of regularizer on the states. However, there are several more interesting questions that could be explored:

- What types of examples should be combined? Is it best to combine within an episode or between episodes? Is there some special consideration for high value states which could be relevant?

- What is the right type of target or loss to combine? Does it make sense to think about values, especially due to their high variance?

- Does consistency (as used in SSL) also have a relevance in reinforcement

learning? This could be one flexible way of exploiting off-policy data.

- If we have a model or a partial model, could we also use these "synthetic states", and possibly combine them with real samples?

6.2 Mixing based Self-supervision objectives

Self-supervision based methods have emerged as a promising approach for learning the representations with any explicit annotations. These methods typically design an auxiliary task that is domain-specific. For instance, for computer vision, these auxiliary tasks include colorization [183], placing image patches in correct order [109], inpainting [115], etc. Although domain specific auxiliary tasks are appealing because they can probably more effectively capture the structure in the data, designing domain-agnostic auxiliary tasks is an important research problem because of its applicability to many domains in which domain-specific tasks can not be designed. To this end, *mixing* based tasks can be a promising approach. The simplest form of such a task could be to learn the hidden representations that are amenable to *demixing* (constructing the original pair of input samples from the mixed sample). Note that demixing would be only possible if the DNN learns sufficiently good representations from the data.

Contrastive learning [52] is another family of methods that has been shown to achieve state-of-the-art for self-supervised learning [155, 169, 58]. In this kind of approach, a DNN is trained with the objective that it learns to map the "similar" samples closer in the hidden space than the "dissimilar" samples. Mixing based approaches can be directly integrated with this kind of loss. For instance, for a given sample, a similar sample can be synthesized by its linear interpolation with other samples using a small mixing coefficient. Similarly, a dissimilar sample can be created by its linear interpolation with other samples using a large mixing coefficient.

6.3 Connecting Manifold Mixup to Temporal Robustness in the Brain's computation

Deep neural networks run on a standardized clock, in which layers propagate a forward pass in order. Then the layers conduct a coordinated backward pass that uses the hidden states from the forward pass.

An intriguing observation is that the above mentioned synchronized updates in the Deep neural networks differ from computation in the brain, where neurons do not fire on a simple synchronized clock. One possibility

is that neurons in the brain might be able to relax this strict computational condition by continuously receiving new inputs, and simply mixing the newly computed hidden values with the previously stored hidden values. If the feedback signals get mixed in the same way, this might amount to something similar to *Manifold Mixup* .

The more conservative way to do this would involve using the running mean in the forward pass, while still doing a normal backward pass on these mixed hidden states. Does this confer any actual computational benefit? Perhaps it's a valid way of mixing on multiple layers at once?

However, to remove the coupling between layers, we would also need a way to do the backward pass in a decoupled way, which seems a more challenging problem.

References

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[3] Jimmy Ba, Jamie Kiros, and Geoffrey Hinton. Layer normalization. 07 2016.

[4] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *NIPS*, 2014.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[6] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint arXiv:1910.08051*, 2019.

[7] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Netw.*, 2(1):53–58, January 1989.

[8] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[9] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[10] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 17–36, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

[11] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 153–160, Cambridge, MA, USA, 2006. MIT Press.

[12] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations*, 2020.

[13] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. *arXiv e-prints*, page arXiv:1905.02249, May 2019.

[14] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019.

[15] Radoslaw Bialobrzeski, Michal Kosmider, Mateusz Matuszewski, Marcin Plata, and Alexander Rakowski. Robust bayesian and light neural networks for voice spoofing detection. *Proc. Interspeech 2019*, pages 1028–1032, 2019.

[16] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995.

[17] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[18] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL*, 2015.

[19] David Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. *ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM)*, RSRE-MEMO-4148, 03 1988.

[20] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1145–1152. AAAI Press, 2016.

[21] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[22] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level residual networks from dynamical systems view. In *International Conference on Learning Representations*, 2018.

[23] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 119–128, New York, NY, USA, 2015. Association for Computing Machinery.

[24] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[25] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir. Vicinal risk minimization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 416–422. MIT Press, 2001.

[26] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[27] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1106–1114, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[28] Zihang Dai, Zhilin Yang, Fan Yang, William W. Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6513–6523, Red Hook, NY, USA, 2017. Curran Associates Inc.

[29] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1:886–893, 2005.

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

[31] Zhijie Deng, Yinpeng Dong, and Jun Zhu. Batch virtual adversarial training for graph convolutional networks. *CoRR*, abs/1902.09192, 2019.

[32] Ming Ding, Jie Tang, and Jie Zhang. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 913–922, New York, NY, USA, 2018. ACM.

[33] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[34] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[35] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *CoRR*, abs/1902.08226, 2019.

[36] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6530–6539. Curran Associates, Inc., 2017.

[37] Geoff French, Timo Aila, Samuli Laine, Michal Mackiewicz, and Graham Finlayson. Consistency regularization and cutmix for semi-supervised semantic segmentation. *arXiv preprint arXiv:1906.01916*, 2019.

[38] Hongchang Gao and Heng Huang. Deep attributed network embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3364–3370. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[39] Hongyang Gao and Shuiwang Ji. Graph u-nets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2083–2092, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[40] Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan O. Arik, Larry S. Davis, and Tomas Pfister. Consistency-Based Semi-Supervised Active Learning: Towards Minimizing Labeling Cost. *arXiv e-prints*, page arXiv:1910.07153, October 2019.

[41] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[42] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[43] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[44] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[46] Klaus Greff, Rupesh Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. 12 2016.

[47] David Ha and Douglas Eck. A neural representation of sketch drawings. *ArXiv*, abs/1704.03477, 2017.

[48] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018. https://worldmodels.github.io.

[49] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34, 05 2017.

[50] Eldad Haber, Lars Ruthotto, Elliot Holtham, and Seong-Hwan Jun. Learning across scales - multiscale methods for convolution neural networks. In *AAAI*, pages 3142–3148, 2018.

[51] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*, volume 4. 01 1985.

[52] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, page 1735–1742, USA, 2006. IEEE Computer Society.

[53] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates, Inc., 2017.

[54] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[55] Ryuichiro Hataya and Hideki Nakayama. Unifying semi-supervised and robust learning by mixup. 2019.

[56] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. *arXiv preprint arXiv:1910.02509*, 2019.

[57] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[58] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. *arXiv e-prints*, page arXiv:1911.05722, November 2019.

[59] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *ArXiv*, abs/1911.05722, 2019.

[60] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[61] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5353–5360, 2015.

[62] G E Hinton and R R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[63] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

[64] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

[65] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[66] Xiaofeng Hong and Gang Liu. Multi-label audio tagging system for freesound 2019: Focusing on network architectures, label noisy and loss functions.

[67] J. J. Hopfield. *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, page 457–464. MIT Press, Cambridge, MA, USA, 1988.

[68] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. In *International Conference on Learning Representations*, 2019.

[69] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[70] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269. IEEE Computer Society, 2017.

[71] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. volume 9908, pages 646–661, 10 2016.

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

[73] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[74] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5308–5317, June 2016.

[75] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[76] Wonmo Jung, Sejin Park, Kyu-Hwan Jung, and Sung Il Hwang. Prostate cancer segmentation using manifold mixup u-net. 2019.

[77] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016.

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

[79] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014.

[80] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[81] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[82] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *INTERSPEECH*, 2015.

[83] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, January 1982.

[84] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[85] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[86] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016.

[87] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[88] Max W. Y. Lam, Jun Wang, Dan Su, and Dong Yu. Mixup-breakdown: a consistency training method for improving generalization of speech separation models. *arXiv e-prints*, page arXiv:1910.13253, October 2019.

[89] Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, page 95–103, New York, NY, USA, 2019. Association for Computing Machinery.

[90] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1558–1566, New York, New York, USA, 20–22 Jun 2016. PMLR.

[91] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[92] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[93] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

[94] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated Graph Sequence Neural Networks. *arXiv e-prints*, page arXiv:1511.05493, November 2015.

[95] Qianli Liao and Tomaso Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. 04 2016.

[96] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 700–708. Curran Associates, Inc., 2017.

[97] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, page 1150, USA, 1999. IEEE Computer Society.

[98] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[99] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[100] Xudong Mao, Yun Ma, Zhenguo Yang, Yangbin Chen, and Qing Li. Virtual Mixup Training for Unsupervised Domain Adaptation. *arXiv e-prints*, page arXiv:1905.04215, May 2019.

[101] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[102] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014.

[103] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[104] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[105] Bastien Moysset and Ronaldo Messina. Manifold mixup improves text recognition with ctc loss. *arXiv preprint arXiv:1903.04246*, 2019.

[106] Preetum Nakkiran. Adversarial robustness may be at odds with simplicity. *arXiv preprint arXiv:1901.00532*, 2019.

[107] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen andvYang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017.

[108] Radford M. Neal. Connectionist learning of belief networks. *Artif. Intell.*, 56(1):71–113, July 1992.

[109] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. *arXiv e-prints*, page arXiv:1603.09246, March 2016.

[110] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[111] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, July 2002.

[112] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[113] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. *arXiv preprint arXiv:1909.11515*, 2019.

[114] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *arXiv e-prints*, page arXiv:1904.08779, Apr 2019.

[115] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. *arXiv e-prints*, page arXiv:1604.07379, April 2016.

[116] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[117] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. cite arxiv:1511.06434Comment: Under review as a conference paper at ICLR 2016.

[118] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Adversarial Training Can Hurt Generalization. *arXiv e-prints*, page arXiv:1906.06032, Jun 2019.

[119] Alexander Rakowski and Michał Kosmider. Frequency-aware cnn for open set acoustic scene classification. Technical report, DCASE2019 Challenge, Tech. Rep, 2019.

[120] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1060–1069. JMLR.org, 2016.

[121] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NIPS*, pages 91–99, 2015.

[122] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 833–840, USA, 2011. Omnipress.

[123] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[124] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, page 3, 2005.

[125] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA, 1988.

[126] Tim Sainburg, Marvin Thielk, Brad Theilman, Benjamin Migliori, and Timothy Gentner. Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions. *CoRR*, abs/1807.06650, 2018.

[127] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1171–1179, USA, 2016. Curran Associates Inc.

[128] V. Sandfort, K. Yan, and P.J. Pickhardt. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Nature Sci Rep 9, 16884 (2019)*, 2019.

[129] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, January 2009.

[130] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv e-prints*, page arXiv:1312.6229, December 2013.

[131] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349*, 2017.

[132] John Shawe-Taylor, Peter Bartlett, Robert C. Williamson, and Martin Anthony. A framework for structural risk minimisation. pages 68–76, 01 1996.

[133] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, April 2017.

[134] Li Shen, Laurie Margolies, Joseph Rothstein, Eugene Fluder, Russell McBride, and Weiva Sieh. Deep learning to improve breast cancer detection on screening mammography. *Scientific Reports*, 9:1–12, 08 2019.

[135] Ziqiang Shi, Liu Liu, Huibin Lin, Rujie Liu, and Anyan Shi. Hodgepodge: Sound event detection based on ensemble of semi-supervised learning methods. *arXiv preprint arXiv:1907.07398*, 2019.

[136] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810, 2017.

[137] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.

[138] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs, 2017. cite arxiv:1704.02901Comment: Accepted to CVPR 2017; extended version.

[139] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. *arXiv e-prints*, page arXiv:2001.07685, January 2020.

[140] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE transactions on neural networks*, 8 3:714–35, 1997.

[141] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[142] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[143] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[144] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

[145] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[146] Taiji Suzuki. Compression based bound for non-compressed network: unified generalization error analysis of large compressible deep neural network. *arXiv preprint arXiv:1909.11274*, 2019.

[147] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[148] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[149] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1195–1204. Curran Associates, Inc., 2017.

[150] Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *CoRR*, abs/1708.01289, 2017.

[151] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 1996.

[152] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *CoRR*, abs/1503.02406, 2015.

[153] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[154] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. *arXiv e-prints*, page arXiv:1805.12152, May 2018.

[155] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[156] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6306–6315. Curran Associates, Inc., 2017.

[157] S. Vanmarcke, F. Calders, and F. Wagemans. The time-course of ultrarapid categorization: The influence of scene congruency and top-down processing. *i-Perception*, 2016.

[158] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[159] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[160] Andreas Veit, Michael Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 550–558, Red Hook, NY, USA, 2016. Curran Associates Inc.

[161] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[162] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold Mixup: Better Representations by Interpolating Hidden States. *arXiv e-prints*, page arXiv:1806.05236, Jun 2018.

[163] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6438–6447, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[164] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation Consistency Training for Semi-Supervised Learning. *arXiv e-prints*, page arXiv:1903.03825, Mar 2019.

[165] Vikas Verma, Meng Qu, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. GraphMix: Regularized Training of Graph Neural Networks for Semi-Supervised Learning. *arXiv e-prints*, page arXiv:1909.11715, Sep 2019.

[166] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM.

[167] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1225–1234, New York, NY, USA, 2016. Association for Computing Machinery.

[168] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory Networks. *arXiv e-prints*, page arXiv:1410.3916, October 2014.

[169] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *CoRR*, abs/1805.01978, 2018.

[170] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *arXiv e-prints*, page arXiv:1901.00596, January 2019.

[171] Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation. *ArXiv*, abs/1904.12848, 2019.

[172] Saining Xie, Ross Girshick, Piotr Dollár, Z. Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. 11 2016.

[173] Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Daniel Jurafsky, and Andrew Y. Ng. Data noising as smoothing in neural network language models. *ArXiv*, abs/1703.02573, 2017.

[174] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[175] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[176] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

[177] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *arXiv preprint arXiv:1905.04899*, 2019.

[178] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2016. cite arxiv:1611.03530Comment: Published in ICLR 2017.

[179] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[180] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[181] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

[182] Richard Zhang, Phillip Isola, and Alexei Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. pages 645–654, 07 2017.

[183] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[184] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.

[185] Zizhao Zhang, Han Zhang, Sercan O Arik, Honglak Lee, and Tomas Pfister. Ieg: Robust neural network training to tackle severe label noise. *arXiv preprint arXiv:1910.00701*, 2019.

Aalto-DD 198/2020

ISBN 978-952-64-0159-1 (printed)
ISBN 978-952-64-0160-7 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS