DISS. ETH NO. 30215

# FORMAL AND STATISTICAL CERTIFICATION OF ROBUSTNESS AND FAIRNESS FOR AI

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by

MISLAV BALUNOVIĆ
MSc ETH CS, ETH Zurich

born on 15.08.1995

accepted on the recommendation of

Prof. Dr. Martin Vechev
Prof. Dr. Matt Fredrikson
Prof. Dr. Mario Fritz

2024

Mislav Balunović: *Formal and Statistical Certification of Robustness and Fairness for AI*, © 2024

# ABSTRACT

As deep learning permeates domains such as computer vision and natural language, and becomes more integrated with larger critical systems that can affect humans, resolving the AI safety issues becomes one of the central challenges in the field. In such important systems, the overarching goal is no longer only to build the most accurate AI models, but to build those AI models that are both highly accurate and provably safe at the same time. This thesis considers two important aspects of AI safety: robustness and fairness. Robustness requires that the model also performs well under conditions where input distribution differs from the one encountered during training. Fairness is a safety property that requires model predictions to be fair across different individuals and groups.

In the first part of the thesis, we focus on robustness. We first propose a novel certification method that can guarantee that models are robust to input transformations, and in the second chapter we extend this to actually train the models to be provably robust. These contributions are based on novel techniques such as certification via optimization and sampling, as well as training by finding adversarial examples inside of convex relaxations. The second part of the thesis considers fairness where we develop novel methods for learning fair representations that provably satisfy individual or group fairness. We also make a connection between individual fairness and robustness, enabling us to use the techniques from the first part of the thesis to also address fairness.

The methods presented in this thesis can be applied broadly, e.g. robustness methods can be applied to more complex input transformations, while group fair representation learning can also be used to encode inputs into more private representations. We believe that the methods presented in this thesis improve our toolbox for provable AI safety, and could in the future be applied in even more systems.

# ZUSAMMENFASSUNG

In dem Maße, in dem Deep Learning Bereiche wie Computer Vision und natürliche Sprache durchdringt und zunehmend in größere kritische Systeme integriert wird, die Auswirkungen auf den Menschen haben können, wird die Lösung von KI-Sicherheitsfragen zu einer der zentralen Herausforderungen in diesem Bereich. Bei solch wichtigen Systemen besteht das übergreifende Ziel nicht mehr nur darin, die genauesten KI-Modelle zu entwickeln, sondern solche KI-Modelle zu bauen, die gleichzeitig hochgenau und nachweislich sicher sind. In dieser Arbeit werden zwei wichtige Aspekte der KI-Sicherheit betrachtet: Robustheit und Fairness. Robustheit setzt voraus, dass das Modell auch unter Bedingungen gut funktioniert, bei denen die Eingabeverteilung von der beim Training angetroffenen abweicht. Fairness ist eine Sicherheitseigenschaft, die voraussetzt, dass die Modellvorhersagen für verschiedene Personen und Gruppen fair sind.

Im ersten Teil der Arbeit konzentrieren wir uns auf die Robustheit. Wir schlagen zunächst eine neuartige Zertifizierungsmethode vor, die garantiert, dass die Modelle gegenüber Eingabetransformationen robust sind, und im zweiten Kapitel erweitern wir diese Methode, um die Modelle tatsächlich so zu trainieren, dass sie nachweislich robust sind. Diese Beiträge basieren auf neuartigen Techniken wie der Zertifizierung durch Optimierung und Sampling sowie dem Training durch das Auffinden von Gegenbeispielen innerhalb von konvexen Relaxationen. Der zweite Teil der Arbeit befasst sich mit der Fairness, wobei wir neuartige Methoden zum Erlernen fairer Repräsentationen entwickeln, die nachweislich individuelle oder gruppenbezogene Fairness erfüllen. Wir stellen auch eine Verbindung zwischen individueller Fairness und Robustheit her, die es uns ermöglicht, die Techniken aus dem ersten Teil der Arbeit für das Thema Fairness zu nutzen.

Die in dieser Arbeit vorgestellten Methoden können breit angewendet werden, z.B. können Robustheitsmethoden auf komplexere Eingabetransformationen angewendet werden, während gruppengerechtes Repräsentationslernen auch dazu verwendet werden kann, Eingaben in privatere Repräsentationen zu kodieren. Wir glauben, dass die in dieser Arbeit vorgestellten Methoden unseren Werkzeugkasten für nachweisbare KI-Sicherheit verbessern und in Zukunft in noch mehr Systemen angewendet werden könnten.

# PUBLICATIONS

This thesis is based on the following publications:

- *Certifying Geometric Robustness of Neural Networks*, NeurIPS 2019 [1]
  **Mislav Balunović**, Maximilian Baader, Gagandeep Singh, Timon Gehr, Martin Vechev.
- *Adversarial Training and Provable Defenses: Bridging the Gap*, ICLR 2020 [2]
  **Mislav Balunović**, Martin Vechev.
- *Learning Certified Individually Fair Representations*, NeurIPS 2020 [3]
  Anian Ruoss, **Mislav Balunović**, Marc Fischer, Martin Vechev.
- *Fair Normalizing Flows*, ICLR 2022 [4]
  **Mislav Balunović**, Anian Ruoss, Martin Vechev.

The following publications were part of my Ph.D. research and contain results that are supplemental to this work or build upon the results of this thesis:

- *Efficient Certification of Spatial Robustness*, AAAI 2021 [5]
  Anian Ruoss, Maximilian Baader, **Mislav Balunović**, Martin Vechev
- *Certify or Predict: Boosting Certified Robustness with Compositional Architectures*, ICLR 2021 [6]
  Mark Niklas Müller, **Mislav Balunović**, Martin Vechev
- *Scalable Polyhedral Verification of Recurrent Neural Networks*, CAV 2021 [7]
  Wonryong Ryou, Jiayu Chen, **Mislav Balunović**, Gagandeep Singh, Andrei Dan, Martin Vechev
- *Robustness Certification for Point Cloud Models*, ICCV 2021 [8]
  Tobias Lorenz, Anian Ruoss, **Mislav Balunović**, Gagandeep Singh, Martin Vechev
- *On the Paradox of Certified Training*, TMLR 2022 [9]
  Nikola Jovanović*, **Mislav Balunović***, Maximilian Baader, Martin Vechev

- *Latent Space Smoothing for Individually Fair Representations*, ECCV 2022 [10]
  Momchil Peychev, Anian Ruoss, **Mislav Balunović**, Maximilian Baader, Martin Vechev

- *FARE: Provably Fair Representation Learning with Practical Certificates*, NeurIPS 2023 [11]
  Nikola Jovanović, **Mislav Balunović**, Dimitar I. Dimitrov, Martin Vechev

- *From Principle to Practice: Vertical Data Minimization for Machine Learning*, IEEE S&P 2024 [11]
  Robin Staab, Nikola Jovanović, **Mislav Balunović**, Martin Vechev

The following publications were part of my doctoral research but present results outside the scope of this thesis:

- *Bayesian Framework for Gradient Leakage*, ICLR 2022 [12]
  **Mislav Balunović**, Dimitar I. Dimitrov, Robin Staab, Martin Vechev

- *Data Leakage in Federated Averaging*, TMLR 2022 [13]
  Dimitar I. Dimitrov, **Mislav Balunović**, Nikola Konstantinov, Martin Vechev

- *LAMP: Extracting Text from Gradients with Language Model Priors*, NeurIPS 2022 [14]
  **Mislav Balunović***, Dimitar I. Dimitrov*, Nikola Jovanović, Martin Vechev

- *TabLeak: Tabular Data Leakage in Federated Learning*, ICML 2023 [15]
  Mark Vero, **Mislav Balunović**, Dimitar I. Dimitrov, Martin Vechev

# ACKNOWLEDGEMENTS

First, I would like to thank my advisor Prof. Martin Vechev who gave me the opportunity to be part of his research lab. Thank you for teaching me how to be a researcher and demonstrating by example how much can be achieved through personal commitment and dedication. I greatly enjoyed my time in the SRI Lab, the atmosphere and the collaborative environment there are what any PhD students should wish for. I would also like to thank Prof. Matt Fredrikson and Prof. Mario Fritz for agreeing to serve on my doctoral examination committee.

Most of the research I did during my PhD would not have been possible without my collaborators. I would especially like to thank my co-authors on papers that were part of this thesis: Anian Ruoss, Maximilian Baader, Gagandeep Singh, Marc Fischer and Timon Gehr. Furthermore, I am also grateful to my collaborators on all other papers I published during my PhD. I learned a lot from all of you.

I was happy to be part of the SRI Lab, and I greatly enjoyed lunches and dinners we had together, and thank you for the support while we were working towards deadlines. I would also like to thank our administrative assistant Fiorella Meyer who was always very helpful.

Throughout my educational journey I was supported by great number of people. I especially want to thank Prof. Slavica Danilović in whose informatics class I learned to program, and to ETH for providing me with a scholarship for my Master studies.

I am grateful to all of my friends who were with me in both good and bad moments: Ivan, Marko, Robert, Marko, Roko, Toni, Jurica, Jelena, Domagoj, Toni, Ivica and others. I would especially like to thank my girlfriend Andrea for always being here to listen and making me feel like I never left home.

Finally, my biggest thanks go to my family who have been with me all the way: my parents Blažena and Zvonimir, and my sister Neda. Hvala vam za podršku na cijelom putu!

# CONTENTS

1 Introduction 1
  1.1 Chapter 3: Certifying Geometric Robustness 2
  1.2 Chapter 4: Training provably robust networks 3
  1.3 Chapter 5: Provably individually fair representations 3
  1.4 Chapter 6: Provably group fair representations 4
2 Background 5
3 Certifying Geometric Robustness of Neural Networks 9
  3.1 Related work 11
  3.2 Background 13
  3.3 Asymptotically optimal linear constraints 16
  3.4 Experimental evaluation 21
  3.5 Discussion 26
4 Adversarial Training and Provable Defenses: Bridging the Gap 27
  4.1 Related work 29
  4.2 Background 30
  4.3 Provable Defense via Convex Layerwise Adversarial Training 32
  4.4 Convex layerwise adversarial training using linear relaxations 35
  4.5 Certification of neural networks 38
  4.6 Experimental Evaluation 39
  4.7 Discussion 43
5 Learning Certified Individually Fair Representations 45
  5.1 Overview 46
  5.2 Related Work 50
  5.3 Learning Individually Fair Representations 51
  5.4 Certifying Individual Fairness 53
    5.4.1 Certifying Latent Similarity 54
    5.4.2 Certifying Local Robustness 55
  5.5 Experimental Evaluation 56
  5.6 Discussion 59
6 Learning Certified Group Fair Representations 61
  6.1 Related Work 63
  6.2 Background 65
  6.3 Motivation 66

6.4 Fair Normalizing Flows 68
6.5 Experimental Evaluation 72
6.6 Discussion 76
7 Conclusion and outlook 79

Bibliography 81

8 Curriculum Vitae 101

# 1

# INTRODUCTION

Deep learning has achieved significant success over the last decade. The models based on deep learning are now used in a variety of different domains such as recommendation [16], computer vision [17, 18, 19], and culminating in the general tasks involving natural language [20]. However, the increase in capabilities of deep learning models has also raised significant safety concerns, as these models are not used in isolation, but as part of a larger system where their predictions could have downstream consequences for humans. For example, a computer vision model can be used as part of an autonomous vehicle to help it detect street signs, and its predictions affect the safety of the passengers and other participants in the traffic. A model for predicting a student's GPA could be used as a part of college admissions, meaning that the quality of its predictions can affect the educational path of a person. This underlines the importance of studying AI safety, and more concretely worst-case performance of these systems so that we know they will behave correctly in as many situations as possible.

SAFE AI: ROBUSTNESS AND FAIRNESS In this thesis, we focus on two areas of safe AI: *robustness* and *fairness*. A machine learning model is robust if it behaves correctly under changes to the input. These changes can be performed by an adversary (e.g. someone intentionally adds noise to the image) or occur naturally during inference (e.g. by shifting the camera recording an image). Solving this problem is important in order to develop machine learning systems that can function well even when the environment changes. Fairness is another important topic in AI safety, which has gained importance as machine learning models are often being trained on biased data and consequently often start to exhibit such biased behavior themselves. While there are many different fairness definitions, in this thesis we focus on individual fairness, requiring that similar individuals receive similar outcomes, and group fairness, requiring that average prediction across two groups is similar. As machine learning is increasingly being used to make important societal decisions, it is important to adequately address the problem of fairness.

GOAL The main goal of this thesis is to advance the methods that can guarantee robustness and fairness of machine learning models. To do this, we use a combination of formal methods (e.g. abstract interpretation and convex relaxations) and statistical approaches (e.g. finite-sample bounds). Specifically, in Chapter 3 we introduce a method, based on the combination of optimization and sampling, to certify that the machine learning model is robust against geometric transformations. Then, in Chapter 4 we consider not only certifying the model, but training the model itself to be provably robust, using the novel concept of training with latent adversarial examples. Chapter 5 for the first time tackles the problem of preprocessing the input data so that individual fairness can be provably certified by the data consumers. Finally, Chapter 6 introduces a new preprocessing method so that transformed inputs provably cannot be used to infer sensitive attributes, guaranteeing group fairness of any downstream classifier built on top of it.

In Table 1.1 we provide a summary of contributions in this thesis, where each row corresponds to a single chapter: safety property considered in the chapter, as well as the key method used for checking safety. We now provide a high level overview of the contributions of this thesis, separated into different chapters.

| Safety property | Method | Chapter |
|---|---|---|
| Geometric robustness | Optimization and sampling | Chapter 3 |
| Local robustness | Latent adversarial examples | Chapter 4 |
| Individual fairness | Logical and continuous constraints | Chapter 5 |
| Group fairness | Encoding via normalizing flows | Chapter 6 |

TABLE 1.1: Summary of the thesis contributions (each row corresponds to one chapter).

## 1.1 CHAPTER 3: CERTIFYING GEOMETRIC ROBUSTNESS

Our first contribution is motivated by a practical scenario where the input is first transformed with a geometric transformation before being passed to the network. This could for example occur when the camera which is recording the image is rotated or shifted. While the prior work has approached this problem by applying interval bounds to each operation in the transformation, the key insight of our work is that we can formulate an optimization problem whose solutions are tightest possible linear bounds for the *entire* sequence of transformations. We develop and implement practical algorithms and show that they can certify robustness to wide range of geometric transformations and their compositions (e.g. rotations, translations, shearing, etc.) for significantly more images than prior work.

IMPACT The techniques presented in Chapter 3 also have a broader impact and were later applied more generally to also certify robustness to spatial [5], point cloud [8] and audio [7] transformations. This increased variety of specifications and domains for which we can successfully certify robustness, enabling more safe applications of AI.

## 1.2 CHAPTER 4: TRAINING PROVABLY ROBUST NETWORKS

While in Chapter 3 the focus is on certifying the robust error of the model, the goal of Chapter 4 is to *train* the model to be both provably highly robust and have high standard accuracy. This is an important line of research as models trained without such techniques typically cannot easily be certified to be robust. Before our work presented in the chapter, it was difficult to train networks that have high certified robustness and high accuracy, especially at smaller levels of noise. The main insight in this chapter is to use the approach of adversarial training not for searching for adversarial inputs in the input region, but for searching in the convex regions produced by propagating the original shape through the network. This allows a more fine-grained trade-off between provable robustness and accuracy, ultimately resulting in better models than prior work.

IMPACT Our training method also had further impact, as most of the latest methods for provable training [21, 22, 23] are based on similar observation as ours, namely that connecting heuristic and provable defenses allows for training models with better trade-off between certified robustness and accuracy. We also studied [9] theoretical questions raised in this chapter in order to increase understanding of why certain convex relaxations perform better in training.

## 1.3 CHAPTER 5: PROVABLY INDIVIDUALLY FAIR REPRESENTATIONS

In Chapter 5 conceptually we switch from robustness to fairness. However, at the technical level the whole idea presented in this chapter is based on the fact that there is a connection between individual fairness and robustness. More concretely, individual fairness requires similar individuals to be classified similarly, the same as robustness considered in Chapter 3 and Chapter 4 requires the input and its neighbors to be classified similarly. In this chapter we introduce a new method for pre-processing the data (or representation learning) so that data consumers know any model they train on such data will provably satisfy individual fairness.

IMPACT We later extended this approach to individual fairness in computer vision models [10], thus showing broader applicability of the framework presented here to other domains. Other work has applied similar approaches to other types of individual fairness guarantees for representations [24]. Overall, contributions in this chapter make training provably individually fair representations significantly more practical.

## 1.4 CHAPTER 6: PROVABLY GROUP FAIR REPRESENTATIONS

In the final chapter, we continue with the concept of learning provably fair representations, this time for group fairness definition of privacy (e.g. demographic parity or equalized odds). Group fairness definitions are often more widely used in practice as they are easier to define and evaluate than the individual ones, and thus provably learning such fair representations is of great practical importance. Prior work has shown equivalence between satisfying group fairness and making sure that sensitive attributes cannot be recovered from the representations. The key idea of our method (called FNF) is to use statistical approach based on normalizing flows [25] to compute probability densities of the representations in the latent space, allowing us to compute an upper bound on the maximum accuracy any adversarial classifier can have in predicting the sensitive attribute.

IMPACT FNF also had impact on future work: in the follow-up paper we proposed a novel method FARE [11] which addressed limitations of FNF (requiring knowledge of the prior distribution). There has also been broader impact in privacy where we investigated predicting sensitive attributes from data [26, 27] which can be seen as representation learning methods.

# 2

# BACKGROUND

In this chapter we introduce the background and the key concepts which are necessary to understand contributions of this thesis which follow in other chapters. We also briefly discuss the most related work to the topics considered here.

ROBUST MACHINE LEARNING The first key concept is robust machine learning and how it distinguishes itself from the traditional machine learning. Traditional machine learning focuses on learning model parameters $\theta$ which minimize the expected loss of the model $h_\theta$ when input-output pairs $(\boldsymbol{x}, y)$ are sampled from the data distribution $D$. Formally the goal is to find parameters $\theta$ which minimize the expected error $\operatorname{Err}(h_\theta)$:

$$\operatorname{Err}(h_\theta) = \mathbb{E}_{(\boldsymbol{x},y)\sim D}\mathcal{L}(h_\theta(\boldsymbol{x}), y), \tag{2.1}$$

where $\mathcal{L}(y', y) = \mathbb{1}_{y' \neq y}$ is so-called 0-1 error.

While this objective learns the parameters that perform well when inputs are sampled from the distribution $D$, research has found that neural networks trained in such way are not robust to so-called *adversarial examples* [28, 29]. The goal of robust machine learning is to train a model for a case where inputs are transformed before being fed into the network (e.g. with a perturbation or a geometric transformation). Let $\mathcal{S}_0(\boldsymbol{x}) = \{t_{\boldsymbol{\kappa}}(\boldsymbol{x}) \mid \boldsymbol{\kappa} \in \mathcal{H}\}$ be the set of all inputs that can be obtained by passing the original input $\boldsymbol{x}$ through a transformation $t_{\boldsymbol{\kappa}}$ with a parameter $\boldsymbol{\kappa} \in \mathcal{H}$. Formally, the robust error of the model $h_\theta$, denoted as $\operatorname{Err}_{\text{rob}}(h_\theta)$, is defined as the following expected worst-case error:

$$\operatorname{Err}_{\text{rob}}(h_\theta) = \mathbb{E}_{(\boldsymbol{x},y)\sim D} \max_{\boldsymbol{x}' \in \mathcal{S}_0(\boldsymbol{x})} \mathcal{L}(h_\theta(\boldsymbol{x}'), y) \tag{2.2}$$

Here, we are interested in the worst case loss on the input $t_{\boldsymbol{\kappa}}(\boldsymbol{x})$ obtained by transforming originally sampled input $\boldsymbol{x}$ from the data distribution using worst case parameters $\boldsymbol{\kappa} \in \mathcal{H}$. This changes the learning problem from finding parameters $\theta$ which minimize standard error $\operatorname{Err}(h_\theta)$ to those which minimize robust error $\operatorname{Err}_{\text{rob}}(h_\theta)$.

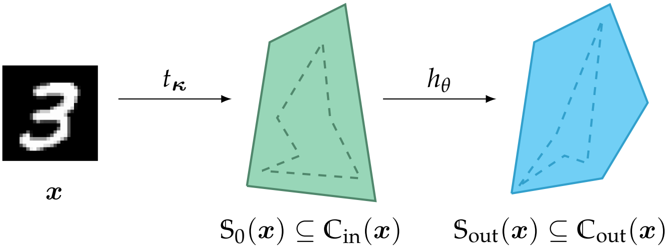

$t_{\boldsymbol{\kappa}}$
$h_\theta$
$\boldsymbol{x}$
$\mathbb{S}_0(\boldsymbol{x}) \subseteq \mathbb{C}_{\text{in}}(\boldsymbol{x})$
$\mathbb{S}_{\text{out}}(\boldsymbol{x}) \subseteq \mathbb{C}_{\text{out}}(\boldsymbol{x})$

FIGURE 2.1: Convex shapes capturing the propagation of an input $\boldsymbol{x}$ through a transformation $t_{\boldsymbol{\kappa}}$ and a neural network $h_\theta$. The exact, but potentially non-convex, shape $\mathbb{S}_0(\boldsymbol{x})$ containing all images that can be obtained by transforming $\boldsymbol{x}$ with $t_{\boldsymbol{\kappa}}$ is over-approximated by a convex shape $\mathbb{C}_{\text{in}}(\boldsymbol{x})$. Final shapes, obtained after passing $\mathbb{S}_0(\boldsymbol{x})$ and $\mathbb{C}_{\text{in}}(\boldsymbol{x})$ through the network are $\mathbb{S}_0(\boldsymbol{x})$ and $\mathbb{C}_{\text{out}}(\boldsymbol{x})$.

In practice, after the first introduction of adversarial examples [28, 29], defense mechanisms to train robust neural networks were built based on the inclusion of adversarial examples to the training set [30, 31]. Models trained using adversarial training with projected gradient descent (PGD) [32] were shown to be robust against the strongest known attacks [33]. This is in contrast to other defense mechanisms which have been broken by new attack techniques [34]. While models trained using adversarial training achieve robustness against strong adversaries, there are no guarantees that model is robust against new adversarial attacks which are constantly being developed [35].

CERTIFICATION VIA CONVEX RELAXATIONS The next concept concerns the following question: how can we formally prove that the model is robust for some input $\boldsymbol{x}$? The general approach we take in this thesis, of certifying using convex relaxations, is shown in Fig. 2.1. The main idea is to compute a linear convex relaxation $\mathbb{C}_{\text{in}}(\boldsymbol{x})$ instead of the possibly non-convex shape $\mathbb{S}_0(\boldsymbol{x})$, such that $\mathbb{S}_0(\boldsymbol{x}) \subseteq \mathbb{C}_{\text{in}}(\boldsymbol{x})$ (*soundness* property) and $\mathbb{C}_{\text{in}}(\boldsymbol{x})$ is as tight as possible. In Chapter 3 we will more formally define what it means for a convex relaxation to be linear and sound. We can then propagate $\mathbb{C}_{\text{in}}(\boldsymbol{x})$ through the model $h_\theta$ and obtain the final output shape $\mathbb{C}_{\text{out}}(\boldsymbol{x})$ for which we can typically easily check if all contained outputs are classified correctly.

A wide range of methods have been proposed to certify robustness of neural networks. Those methods are typically based on abstract interpretation [36, 37], linear relaxation [38, 39, 40], duality [41], SMT solving [42, 43, 44], mixed integer programming [45], symbolic intervals [46], Lipschitz optimization [47], semi-definite relaxations [48] and combining approximations with solvers [49, 50]. Certification procedures can also be extended into end-to-end training of neural networks to be provably robust against adversarial examples [51, 52]. Recent line of work [53, 54, 55] proposes to construct a classifier based on the smoothed neural network which comes with probabilistic guarantees on the robustness against $L_2$ perturbations. None of these works except [56] consider geometric transformations, while [56] only verifies robustness against rotation. The work of [57] also generates linear relaxations of non-linear specifications, but they do not handle geometric transformations. We remark that [58] considers a much more restricted (discrete) setting leading to a finite number of images which allows easier certification.

PROVABLE DEFENSE WITH CONVEX RELAXATIONS Techniques for certifying robustness can also be used during the training to train models which are actually provably robust. This is achieved through minimizing an upper bound to the robust error:

$$\min_{\theta} \overline{\text{Err}_{\text{rob}}}(h_\theta) \tag{2.3}$$

Here $\overline{\text{Err}_{\text{rob}}}(h_\theta) \geq \text{Err}_{\text{rob}}(h_\theta)$ can be computed using the certification techniques introduced earlier.

There has been a considerable amount of work on methods to train classifiers with robustness guarantees. These approaches are typically based on Lipschitz regularization [59], linear [60] or semidefinite [61, 62] relaxations, hybrid zonotope [63] or interval bound propagation [64]. While these approaches obtain robustness guarantees, accuracy of these networks is relatively small and limits practical use of these methods. Another line of work proposes to replace neural networks with a randomized classifier [55, 65, 66] which comes with probabilistic guarantees on its robustness. While these approaches scale to larger datasets such as ImageNet (although with probabilistic instead of exact guarantees), their bounds come from the relationship between $L_2$ robustness and Gaussian distribution. In this thesis, we also consider general verification problem where input is not necessarily limited to an $L_p$ ball, but obtained through a general transformation $t_{\boldsymbol{\kappa}}$.

FAIR REPRESENTATIONS The paradigm of learning fair representations has been established as an effective approach to obtain data representations that preserve fairness while maintaining utility for a variety of downstream

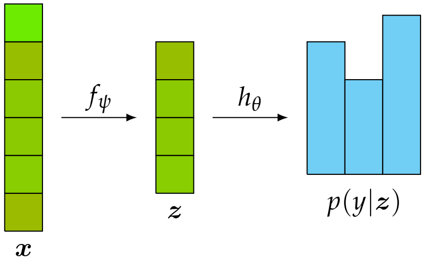


FIGURE 2.2: The concept of learning fair representations. The inputs $\boldsymbol{x}$ are first encoded into a new representation $\boldsymbol{z}$ through an encoder $f_\psi$. The representation $\boldsymbol{z}$ is given as an input to the classifier $h_\theta$ and at the output we obtain probability distribution over labels.

tasks [67, 68]. In this setting, illustrated in Fig. 2.2, a machine learning model $M$ is composed of two parts: an encoder $f_\psi$, provided by the *data producer*, and a classifier $h_\theta$, provided by the *data consumer*. The key advantage of this approach is that the data producer can encode the input data $\boldsymbol{x}$ into a new representation $\boldsymbol{z} = f_\psi(\boldsymbol{x})$, ensure that any classifier $h_\theta$ built on top of this encoded data is fair (according to the chosen fairness definition), and then send the encoded data $\boldsymbol{z}$ to the data consumer. Fair representations can be learned with a variety of different approaches, including variational autoencoders [69, 70], adversarial training [68, 71, 72, 73, 74, 75, 76, 77], and disentanglement [78, 79].

INDIVIDUAL AND GROUP FAIRNESS WITH GUARANTEES In this thesis, we propose new methods to learn fair representations that have provable fairness guarantees, and we consider two fairness notions: *individual* and *group* fairness. Individual fairness requires that two individuals $\boldsymbol{x}$ and $\boldsymbol{x}'$ who are similar (according to some well-defined criterion) receive similar classification outcomes $h_\theta(f_\psi(\boldsymbol{x}))$ and $h_\theta(f_\psi(\boldsymbol{x}'))$. Group fairness requires average classification outcome to be the same for two different groups (where groups are typically determined based on a sensitive attribute). We discuss learning fair representations in the context of individual fairness in Chapter 5, and group fairness in Chapter 6.

# 3

# CERTIFYING GEOMETRIC ROBUSTNESS OF NEURAL NETWORKS

In Chapter 1 we introduced the setting of robust machine learning and described what does it mean for a neural network to be provably robust. Typically, most of the research has focused on creating more precise relaxations for the operations in the network [37, 80, 81] in order to certify robustness to $l_\infty$ noise. However, Engstrom et al. [82] have shown that neural networks are also not robust to geometric transformations such as rotations or translations, which are arguably even more important in practice as they can often naturally occur when camera recording the input is rotated or shifted. This raises an important question: how to guarantee that networks are robust against such geometric transformations?

KEY CHALLENGE FOR CERTIFYING GEOMETRIC ROBUSTNESS The key challenge in certifying geometric robustness is to compute as tight as possible convex set $\mathbb{C}_{\text{in}}(\boldsymbol{x})$ which contains all possible inputs $\boldsymbol{x}' = t_{\boldsymbol{\kappa}}(\boldsymbol{x})$ obtainable by transforming the original input $\boldsymbol{x}$ through a geometric transformation $t_{\boldsymbol{\kappa}}$ parametrized by $\boldsymbol{\kappa}$ (e.g. shifts $\delta_x$ and $\delta_y$ for translation). This set can then be passed to one of the existing neural network verifiers to obtain a certificate that the network is robust for the given input $\boldsymbol{x}$.

PRIOR WORK Certifying such geometric transformations can be performed by overapproximating the input region with an interval shape [56]. Its key idea is summarized in Fig. 3.1: Here, the goal is to prove that any image obtained by translating the original image by some $\delta_x, \delta_y \in [-4, 4]$ is classified to label 3. To accomplish this task, Singh et al. [56] propagate the image and the parameters $\delta_x, \delta_y$ through every component of the transformation using interval bound propagation. The output region $I$ is a convex shape capturing all images that can be obtained by translating the original image between $-4$ and 4 pixels. Finally, $I$ is fed to a standard neural network verifier which tries to prove that all images in $I$ classify to the label 3. This method can also be improved using tighter relaxation based on Polyhedra [83]. Unfortunately, as we show later, bound propagation is not satisfactory. The core issue is that any approach based on bound propagation inherently

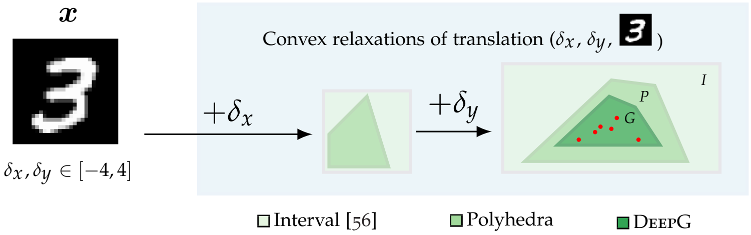

$\boldsymbol{x}$
$\delta_x, \delta_y \in [-4, 4]$
Convex relaxations of translation ($\delta_x$, $\delta_y$, 3)
$+\delta_x$
$+\delta_y$
$I$
$P$
$G$
☐ Interval [56]
🟩 Polyhedra
🟩 DeepG

Figure 3.1: End-to-end certification of geometric robustness using different convex relaxations. Here, the original image $\boldsymbol{x}$ (left) goes through a geometric transformation $t_{\boldsymbol{\kappa}}$ which is translation parametrized by $\boldsymbol{\kappa} = [\delta_x, \delta_y]$. Shapes on the right, produced by Interval, Polyhedra and DeepG, are different candidates for the shape $\mathbb{C}_{\text{in}}(\boldsymbol{x})$ which over-approximates all possible images that could be obtained through the geometric transformation $t_{\boldsymbol{\kappa}}$.

accumulates loss for every intermediate result, often producing regions that are too coarse to allow the neural network verification to succeed.

THIS CHAPTER Instead of sequentially computing bounds for each intermediate operation, we propose a new method based on sampling and optimization which computes a convex relaxation for the *entire composition* of transformations. The key idea of our method is to sample the parameters of the transformation (e.g., $\delta_x, \delta_y$), obtaining sampled points at the output (red dots in Fig. 3.1), and to then compute sound and asymptotically optimal linear constraints around these points (shape $G$). We implemented our method in a system called DeepG and showed that it is significantly more precise than bound propagation (using Interval or Polyhedra relaxation) on a wide range of geometric transformations.

MAIN CONTRIBUTIONS Our main contributions are:

- A novel method, combining sampling and optimization, to compute asymptotically optimal linear constraints bounding the set of geometrically transformed images. We demonstrate that these constraints enable significantly more precise certification compared to prior work.
- A complete implementation of our certification in a system called DeepG. Our results show substantial benefits over the state-of-the-art across a range of geometric transformations. We make DeepG publicly available at `https://github.com/eth-sri/deepg/`.

## 3.1 RELATED WORK

We now discuss some of the closely related work in certification of the neural networks and their robustness to geometric transformations.

CERTIFICATION OF NEURAL NETWORKS Recently, a wide range of methods have been proposed to certify robustness of neural networks against adversarial examples. Those methods are typically based on abstract interpretation [36, 37], linear relaxation [38, 39, 40], duality [41], SMT solving [42, 43, 44], mixed integer programming [45], symbolic intervals [46], Lipschitz optimization [47], semi-definite relaxations [48] and combining approximations with solvers [49, 50]. Certification procedures can also be extended into end-to-end training of neural networks to be provably robust against adversarial examples [51, 52]. Recent line of work [53, 54, 55] proposes to construct a classifier based on the smoothed neural network which comes with probabilistic guarantees on the robustness against $L_2$ perturbations. None of these works except [56] consider geometric transformations, while [56] only verifies robustness against rotation. The work of [57] also generates linear relaxations of non-linear specifications, but they do not handle geometric transformations. We remark that [58] considers a much more restricted (discrete) setting leading to a finite number of images. This means that certification can be performed by brute-force enumeration of this finite set of transformed images. In our setting, as we will see, this is not possible, as we are dealing with an uncountable set of transformed images.

NEURAL NETWORKS AND GEOMETRIC TRANSFORMATIONS There has been considerable research in empirical quantification of geometric robustness of neural networks [82, 84, 85, 86, 87, 88]. Another line of work focuses on the design of architectures which possess an inherent ability to learn to be more robust against such transformations [89, 90]. However, all of these approaches offer only empirical evidence of robustness. Instead, our focus is to provide formal guarantees.

CERTIFICATION OF GEOMETRIC TRANSFORMATIONS Prior work [56] introduced a method for analyzing rotations using the interval propagation and performed certification using the state-of-the-art verifier DeepPoly. It is straightforward to generalize their interval approach to handle more complex transformations beyond rotation. However, as we show experimentally, interval propagation loses precision which is why certification often does not succeed.

To capture relationships between pixel values and transformations, one would ideally use the Polyhedra relaxation [83] instead of intervals. While Polyhedra offers higher precision, its worst-case running time is exponential in the number of variables [91]. Hence, it does not scale to geometric transformations, where every pixel introduces a new variable. Thus, we extended the recent DeepPoly relaxation [56] (a restricted Polyhedra) with custom approximations for the operations used in several geometric transformations (e.g., translation, scaling). Our experimental results show that even though this approach significantly improves over intervals, it is not precise enough to certify robustness of most images in our dataset. In turn, this motivates the method introduced in this chapter.

HEURISTIC ADVERSARIAL DEFENSES After the first introduction of adversarial examples [28, 29], defense mechanisms to train robust neural networks were built based on the inclusion of adversarial examples to the training set [30, 31]. Models trained using adversarial training with projected gradient descent (PGD) [32] were shown to be robust against the strongest known attacks [33]. This is in contrast to other defense mechanisms which have been broken by new attack techniques [34]. While models trained using adversarial training achieve robustness against strong adversaries, there are no guarantees that model is robust against any kind of adversarial attack under the threat model considered.

PROVABLE ADVERSARIAL DEFENSES There has also been considerable amount of work on methods to train classifiers with robustness guarantees. These approaches are typically based on Lipschitz regularization [59], linear [60] or semidefinite [61, 62] relaxations, hybrid zonotope [63] or interval bound propagation [64]. While these approaches obtain robustness guarantees, accuracy of these networks is relatively small and limits practical use of these methods.

There has also been recent work on certification of general neural networks, not necessarily trained in a special way. These methods are based on SMT solvers [44], abstract interpretation [36], mixed-integer linear programs [92], linear relaxations [38, 39, 80] or combinations of those [93, 94].

Another line of work proposes to replace neural networks with a randomized classifier [55, 65, 66] which comes with probabilistic guarantees on its robustness. While these approaches scale to larger datasets such as

ImageNet (although with probabilistic instead of exact guarantees), their bounds come from the relationship between $L_2$ robustness and Gaussian distribution. In this chapter, we consider general verification problem where input is not necessarily limited to an $L_p$ ball, but arbitrary convex set, as explained in Section 3.2.

## 3.2 BACKGROUND

Our goal is to certify the robustness of a neural network against adversarial examples generated using parameterized geometric transformations. In this section we formulate this problem statement, introduce the notation of transformations and provide a running example which we use throughout the chapter to illustrate key concepts.

GEOMETRIC TRANSFORMATIONS We assume that $t_{\boldsymbol{\kappa}}$ is a geometric transformation parametrized by $\boldsymbol{\kappa} \in \mathbb{R}^k$, transforming an input image into another image. For example, if $t_{\boldsymbol{\kappa}}$ is a rotation, then $\boldsymbol{\kappa}$ is a single element vector consisting of a rotation angle, and in the case of translation it is two element vector consisting of vertical and horizontal offsets. More formally, we consider $t_{\boldsymbol{\kappa}}$ to be a composition of a geometric transformation (determining movement of each pixel), interpolation (computing pixel values on a grid) and single-pixel transformations (brightness and contrast).

LINEAR CONVEX RELAXATION Given an original image $\boldsymbol{x}$, the goal of our method is to produce a convex relaxation $\mathcal{C}_{\text{in}}(\boldsymbol{x})$ that contains all images $\boldsymbol{x}'$, for which there exists some $\boldsymbol{\kappa} \in \mathcal{H}$ such that the following linear bounds hold:

$$\mathcal{C}_{\text{in}}(\boldsymbol{x}) = \{\boldsymbol{x}' \mid \exists \boldsymbol{\kappa} \in \mathcal{H}, \boldsymbol{\kappa}^T \boldsymbol{W}_{ij}^{(l)} + b_{ij}^{(l)} \leq x'_{ij} \leq \boldsymbol{\kappa}^T \boldsymbol{W}_{ij}^{(u)} + b_{ij}^{(u)}\}, \tag{3.1}$$

where $\boldsymbol{W}^{(l)}, \boldsymbol{W}^{(u)} \in \mathbb{R}^{n \times n \times k}$ and $\boldsymbol{b}^{(l)}, \boldsymbol{b}^{(u)} \in \mathbb{R}^{n \times n}$ are linear coefficients which determine the convex shape. We only consider *sound* convex relaxation where $\mathcal{S}_0(\boldsymbol{x}) \subseteq \mathcal{C}_{\text{in}}(\boldsymbol{x})$, or more precisely those for which

$$t_{\boldsymbol{\kappa}}(\boldsymbol{x}) \in \mathcal{C}_{\text{in}}(\boldsymbol{x}), \forall (\boldsymbol{\kappa}, \boldsymbol{x}) \in \mathcal{H} \times \mathcal{X} \tag{3.2}$$

This means that all images that could be obtained by transforming the original image with some parameters $\boldsymbol{\kappa}$ are certainly contained inside of the relaxed shape $\mathcal{C}_{\text{in}}(\boldsymbol{x})$.

GEOMETRIC IMAGE TRANSFORMATIONS From now on in this chapter we are going to assume that the input image $\boldsymbol{x}$ is fixed. Moreover, we will decompose the geometric transformation $t_{\boldsymbol{\kappa}}$ into the following components:

- A parameterized spatial transformation $\mathcal{T}_{\boldsymbol{\mu}} : \mathbb{R}^2 \to \mathbb{R}^2$, mapping pixel coordinates to the new coordinates
- An interpolation $I : \mathbb{R}^2 \to \mathbb{R}$ which maps pixel coordinates to the real value and ensuring the result can be represented on a discrete pixel grid
- Parameterized changes in brightness and contrast $\mathcal{P}_{\alpha,\beta} : \mathbb{R} \to \mathbb{R}$ which change the pixel value

Here, transformation parameters $\boldsymbol{\kappa}$ are decomposed into spatial transformation parameters $\boldsymbol{\mu}$, and brightness and contrast parameters $\alpha$ and $\beta$. We assume $\mathcal{T}_{\boldsymbol{\mu}}$ is a composition of bijective transformations such as rotation, translation, shearing and scaling. While our approach also applies to other interpolation methods, in this chapter we focus on the case where $I$ is the commonly-used bilinear interpolation.

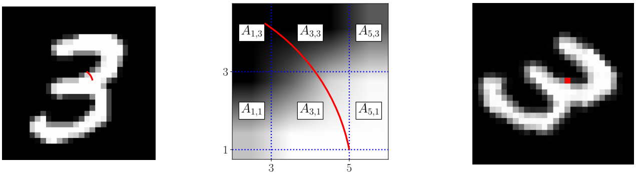
To ease presentation, we assume the image (with integer coordinates) consists of an even number of rows and columns, is centered around $(0, 0)$, and its coordinates are odd integers. We note that all results hold in the general case (without the assumption).

INTERPOLATION The bilinear interpolation $I \colon \mathbb{R}^2 \to [0, 1]$ evaluated on a coordinate $(p_x, p_y) \in \mathbb{R}^2$ is a polynomial of degree 2 given by

$$I(p_x, p_y) := \frac{1}{4} \sum_{\delta_i, \delta_j \in \{0,2\}} x_{i+\delta_i, j+\delta_j} (2 - |i + \delta_i - p_x|)(2 - |j + \delta_j - p_y|). \quad (3.3)$$

Here, $(i, j)$ is the lower-left corner of an *interpolation region* $A_{i,j} := [i, i+2] \times [j, j+2]$ which contains pixel $(p_x, p_y)$. Matrix $\boldsymbol{x}$ consists of pixel values at corresponding coordinates in the original image. The function $I$ is continuous on $\mathbb{R}^2$ and smooth on the interior of every interpolation region. These interpolation regions are depicted with the blue horizontal and vertical dotted lines in Fig. 3.2b.

The pixel value $\tilde{x}_{p_x,p_y}$ of the transformed image can be obtained by (i) calculating the preimage of the coordinate $(p_x, p_y)$ under $\mathcal{T}_{\boldsymbol{\mu}}$, (ii) interpolating the resulting coordinate using $I$ to obtain a value $\xi$, and (iii) applying



(a) Original image

(b) Interpolation

(c) Rotated image

FIGURE 3.2: Image rotated by $-\frac{\pi}{4}$ degrees. Here, (a) shows the original image, while (b) shows part of (a) with a focus on relevant interpolation regions. Finally, (c) shows the resulting rotated image.

the changes in contrast and brightness via $P_{\alpha,\beta}(\xi) = \alpha\xi + \beta$, to obtain the final pixel value $\tilde{x}_{p_x,p_y} = \mathcal{I}_{\alpha,\beta,\boldsymbol{\mu}}(p_x, p_y)$. These three steps are captured by

$$\mathcal{I}_{\alpha,\beta,\boldsymbol{\mu}}(p_x, p_y) := \mathcal{P}_{\alpha,\beta} \circ I \circ \mathcal{T}_{\boldsymbol{\mu}}^{-1}(p_x, p_y). \tag{3.4}$$

RUNNING EXAMPLE To illustrate key concepts introduced throughout the chapter, we use the running example of an MNIST image [95] shown in Fig. 3.2. On this image, we will apply a rotation $R_\phi$ with an angle $\phi$. For our running example, we set $\mathcal{P}_{\alpha,\beta}$ to be the identity.

Consider the pixel $\tilde{p}_{5,1}$ in the transformed image shown in Fig. 3.2c (the pixel is marked with a red dot). The transformed image is obtained by rotating the original image in Fig. 3.2a by an angle $\phi = -\frac{\pi}{4}$. This results in the pixel value

$$\tilde{p}_{5,1} = I \circ R^{-1}_{-\frac{\pi}{4}}(5, 1) = I(2\sqrt{2}, 3\sqrt{2}) \approx 0.30$$

Here, the preimage of point $(5, 1)$ under $R_{-\frac{\pi}{4}}$ produces the point $(2\sqrt{2}, 3\sqrt{2})$ with non-integer coordinates. This point belongs to the interpolation region $A_{1,3}$ and by applying $I(2\sqrt{2}, 3\sqrt{2})$ to the original image in Fig. 3.2a, we obtain the final pixel value $\approx 0.30$ for pixel $(5, 1)$ in the rotated image.

NEURAL NETWORK CERTIFICATION To certify robustness of a neural network with respect to a geometric transformation, we rely on the state-of-the-art verifier DeepPoly [56]. For complex properties such as geometric transformations, the verifier needs to receive a convex relaxation of all possible inputs to the network. If this relaxation is too coarse, the verifier will not be able to certify the property.

PROBLEM STATEMENT To guarantee robustness, our goal is to compute a convex relaxation of all possible images obtainable via the transformation $\mathcal{I}_{\alpha,\beta,\boldsymbol{\mu}}$. This relaxation can then be provided as an input to a neural network verifier (e.g., DeepPoly). If the verifier proves that the neural network classification is correct for all images in this relaxation, then geometric robustness is proven.

## 3.3 ASYMPTOTICALLY OPTIMAL LINEAR CONSTRAINTS

We now present our method for computing the optimal linear approximation (in terms of volume).

MOTIVATION As mentioned earlier, designing custom transformers for every operation incurs precision loss at every step in the sequence of transformations. Our key insight is to define an optimization problem in a way where its solution is the *optimal* (in terms of volume) lower and upper linear constraint for the entire sequence. To solve this optimization problem, we propose a method based on sampling and linear programming. Our method produces, for every pixel, asymptotically optimal lower and upper linear constraints for the *entire composition* of transformations (including interpolation). Such an optimization problem is generally difficult to solve, however, we find that with geometric transformations, our approach is scalable and contributes only a small portion to the entire end-to-end certification running time.

OPTIMIZATION PROBLEM To compute linear constraints for every pixel value, we split the hyperrectangle $\mathcal{H}$ representing the set of possible parameters into $s$ splits $\{h_k\}_{k\in[s]}$. To ease the notation, we consider a fixed pixel $(p_x, p_y)$ for which we want to compute lower and upper linear constraints and denote these as $\boldsymbol{w}_l = \boldsymbol{W}^{(l)}_{p_x p_y}$ and $b_l = b^{(l)}_{p_x p_y}$, and analogously for upper constraints $\boldsymbol{w}_u$ and $b_u$. Our goal will be to compute *sound* lower and upper linear constraints for the pixel value $\mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y)$ for a given pixel $(p_x, p_y)$. Both of these constraints will be linear in the parameters $\boldsymbol{\kappa} = (\alpha, \beta, \boldsymbol{\mu}) \in h_k$. We define *optimal* and *sound* linear (lower and upper) constraints for $\mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y)$ to be a pair of hyperplanes fulfilling

$$\boldsymbol{w}_l^T \boldsymbol{\kappa} + b_l \leq \mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y) \qquad \forall \boldsymbol{\kappa} \in h_k \tag{3.5}$$

$$\boldsymbol{w}_u^T \boldsymbol{\kappa} + b_u \geq \mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y) \qquad \forall \boldsymbol{\kappa} \in h_k, \tag{3.6}$$

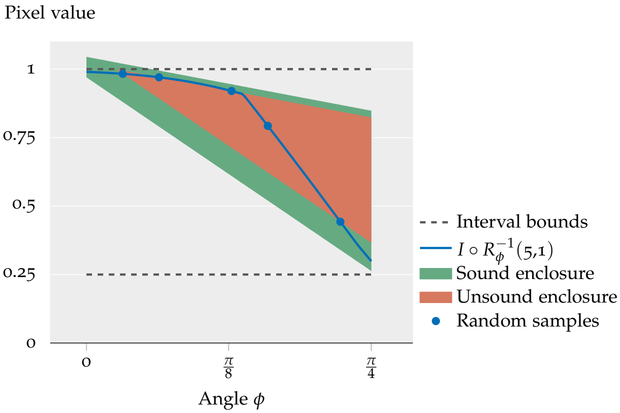


FIGURE 3.3: Unsound (Step 1) and sound (Step 3) enclosures for $I \circ R_\phi^{-1}(5,1)$, with respect to random sampling from $\phi \in [0, \frac{\pi}{4}]$, in comparison to the interval bounds from prior work [56]. Note that $I \circ R_\phi^{-1}(5,1)$ is not piecewise linear, because bilinear interpolation is a polynomial of degree 2.

while minimizing

$$L(\boldsymbol{w}_l, b_l) = \frac{1}{V} \int_{\boldsymbol{\kappa} \in h_k} \Big(\mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y) - (b_l + \boldsymbol{w}_l^T \boldsymbol{\kappa}\Big)) \mathrm{d}\boldsymbol{\kappa} \tag{3.7}$$

$$U(\boldsymbol{w}_u, b_u) = \frac{1}{V} \int_{\boldsymbol{\kappa} \in h_k} \Big((b_u + \boldsymbol{w}_u^T \boldsymbol{\kappa}) - \mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y)\Big) \, \mathrm{d}\boldsymbol{\kappa}. \tag{3.8}$$

Here $V$ denotes the normalization constant equal to the volume of $h_k$. Intuitively, the optimal constraints should result in a convex relaxation of minimum volume. This formulation also allows independent computation for every pixel, facilitating parallelization across pixels. Next, we describe how we obtain lower constraints (upper constraints are computed analogously).

STEP 1: COMPUTE A POTENTIALLY UNSOUND CONSTRAINT To generate a reasonable but a potentially unsound linear constraint, we sample parameters $\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_N$ from $h_k$, approximate the integral in Eq. (3.7) by its

Monte Carlo estimate $L_N$ and enforce the constraints only at the sampled points:

$$\begin{aligned}
&\min_{(\boldsymbol{w}_l, b_l)\in W} L_N(\boldsymbol{w}_l, b_l) = \min_{(\boldsymbol{w}_l, b_l)\in W} \frac{1}{N}\sum_{i=1}^{N}\Big(\mathcal{I}_{\boldsymbol{\kappa}_i}(p_x, p_y) - (b_l + \boldsymbol{w}_l^T\boldsymbol{\kappa}_i)\Big), \\
&b_l + \boldsymbol{w}_l^T\boldsymbol{\kappa}_i \le \mathcal{I}_{\boldsymbol{\kappa}_i}(p_x, p_y) \qquad \forall i \in [N].
\end{aligned} \tag{3.9}$$

This problem can be solved exactly using linear programming (LP). The solution is a potentially unsound constraint $b_l' + \boldsymbol{w}_l'^T\boldsymbol{\kappa}$ (it may violate the constraint at non-sampled points). For our running example, the region bounded by these potentially unsound lower and upper linear constraints is shown as orange in Fig. 3.3.

STEP 2: BOUNDING THE MAXIMUM VIOLATION Our next step is to compute an upper bound on the violation of Eq. (3.5) induced by our potentially unsound constraint from Step 1. This violation is equal to the maximum of the function $f(\boldsymbol{\kappa}) = b_l' + \boldsymbol{w}_l'^T\boldsymbol{\kappa} - \mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y)$ over the hyperrectangle $h_k$. It can be shown that the function $f$ is Lipschitz continuous which enables application of standard global optimization techniques with guarantees [96]. We remark that such methods have already been applied for optimization over inputs to neural network [47, 97].

We show a high level description of this optimization procedure in Algorithm 1. Throughout the optimization, we maintain a partition of the domain of function $f$ into hyperrectangles $h$. The hyperrectangles are stored in a priority queue $q$ sorted by an upper bound $f_h^{\text{bound}}$ of the maximum value the function can take inside of the hyperrectangle. At every step, shown in Line 6, the hyperrectangle with the highest upper bound is further refined into smaller hyperrectangles $h_1', \ldots, h_k'$ and their upper bounds are recomputed. This procedure finishes when the difference between every upper bound and the maximal value at one of the hyperrectangle centers is at most $\epsilon$. Finally, maximum upper bound of the elements in the queue is returned as a result of the optimization.

The two most important aspects of the algorithm, which determine the speed of convergence, are (i) computation of an upper bound, and (ii) choosing an edge along which to refine the hyperrectangle. To compute an upper bound inside of a hyperrectangle spanned between points $\boldsymbol{h}_l$ and $\boldsymbol{h}_u$, we use:

$$f(\boldsymbol{\kappa}) \le f(\tfrac{\boldsymbol{h}^u + \boldsymbol{h}^l}{2}) + \frac{1}{2}\left|\nabla^{\boldsymbol{h}} f\right|^T(\boldsymbol{h}^u - \boldsymbol{h}^l). \tag{3.10}$$

**Algorithm 1** Lipschitz Optimization with Bound Refinement

```
Input: $f, h, k \geq 2$
$f_{\max} := f(\text{center}(h))$
$f_h^{\text{bound}} := f^{\text{bound}}(h, \nabla f)$
$q := [(h, f_h^{\text{bound}})]$
repeat
    pop $(h', f_{h'}^{\text{bound}})$ from $q$ with maximum $f_{h'}^{\text{bound}}$
    $h'_1, \dots, h'_k :=$ partition$(h', \nabla f)$
    for $i := 1$ to $k$ do
        $f_{\max} := \max(f(\text{center}(h'_i)), f_{\max})$
        $f_{h'_i}^{\text{bound}} := f^{\text{bound}}(h'_i, \nabla f)$
        if $f_{h'_i}^{\text{bound}} > f_{\max} + \epsilon$ then
            add $(h'_i, f_{h'_i}^{\text{bound}})$ to $q$
        end if
    end for
until a maximal $f_{h'}^{\text{bound}}$ in $q$ is lower than $f_{\max} + \epsilon$
return $f_{\max} + \epsilon$
```

Here $\left|\nabla^{\boldsymbol{h}} f\right|$ can be any upper bound on the true gradient which satisfies $|\partial_i f(\boldsymbol{\kappa})| \leq \left|\nabla^h f\right|_i$ for every dimension $i$. To compute such a bound, we perform automatic differentiation of the function $f$ using interval propagation. As an added benefit, results of our analysis can be used for pruning of hyperrectangles. We reduce a hyperrectangle to one of its lower-dimensional faces along dimensions for which analysis on gradients proves that the respective partial derivative has a constant sign within the entire hyperrectangle. We also improve on standard refinement heuristics — instead of refining along the largest edge, we additionally weight edge length by an upper bound on the partial derivative of that dimension. In our experiments, we find that these insights speed up convergence compared to simply applying the method out of the box.

Let $v_l$ be the result of the above Lipschitz optimization algorithm. It is then guaranteed that

$$v_l \leq \max_{\boldsymbol{\kappa} \in h_k} \left( b'_l + \boldsymbol{w}'^T_l \boldsymbol{\kappa} - \mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y) \right) \leq v_l + \epsilon.$$

STEP 3: COMPUTE A SOUND LINEAR CONSTRAINT In the previous step we obtained a bound on the maximum violation of Eq. (3.5). Using this bound, in this step, we update our linear constraints $b_l = b'_l - v_l - \epsilon$ and $\boldsymbol{w}_l = \boldsymbol{w}'_l$ to obtain a sound lower linear constraint (it satisfies Eq. (3.5)). The region bounded by the sound lower and upper linear constraints is shown as green in Fig. 3.3. It is easy to check that our constraint is sound:

$$b_l + \boldsymbol{w}_{\boldsymbol{l}}^T \boldsymbol{\kappa} = b'_l - v_l - \epsilon + \boldsymbol{w}_l'^T \boldsymbol{\kappa} \leq \mathcal{I}_{\boldsymbol{\kappa}}(p_x, p_y) \qquad \forall \boldsymbol{\kappa} \in h_k.$$

RUNNING EXAMPLE As in Section 3.2, we focus on the pixel $(5, 1)$, choose $s = 2$ splits for $[0, \frac{\pi}{2}]$, and focus our attention on the split $[0, \frac{\pi}{4}]$. In Step 1, we sample random points {0.1, 0.2, 0.4, 0.5, 0.7} for our parameter $\phi \in [0, \frac{\pi}{4}]$ and evaluate $I \circ R_{\phi}^{-1}(5, 1)$ on these points, obtaining $\{0.98, 0.97, 0.92, 0.79, 0.44\}$.

These points correspond to the blue dots in Fig. 3.3. Solving the LP in Eq. (3.9) yields $b'_l = 1.07$ and $w'_l = -0.90$. Similarly, we compute a potentially unsound upper constraint. Together, these two constraints form the orange enclosure shown in Fig. 3.3. This enclosure is in fact unsound, as some points on the blue line (those not sampled above) are not included in the region.

In Step 2, using Lipschitz optimization for the function $1.07 - 0.9\phi - I \circ R_{\phi}^{-1}$ with $\epsilon = 0.02$ over $\phi \in [0, \frac{\pi}{4}]$ we obtain $v_l = 0.08$ resulting in the sound linear lower constraint $0.97 - 0.9\phi$. This, together with the similarly obtained sound upper constraint, forms the sound (green) enclosure in Fig. 3.3. In the figure we also show the black dashed lines corresponding to the interval bounds from prior work [56] which enclose much larger volume than our linear constraints.

ASYMPTOTICALLY OPTIMAL CONSTRAINTS While our constraints may not be optimal, one can show they are asymptotically optimal as we increase the number of samples:

**Theorem 1.** *Let $N$ be the number of points sampled in our algorithm and $\epsilon$ the tolerance used in the Lipschitz optimization. Let $(\boldsymbol{w}_l, b_l)$ be our lower constraint and let $(\boldsymbol{w}^*, b^*)$ be the minimum of $L$. For every $\delta > 0$ there exists $N_\delta$ such that $|L(\boldsymbol{w}_l, b_l) - L(\boldsymbol{w}^*, b^*)| < \delta + \epsilon$ for every $N > N_\delta$, with high probability. Analogous result holds for upper constraint $(\boldsymbol{w}_u, b_u)$ and function $U$.*

## 3.4 EXPERIMENTAL EVALUATION

We implemented our certification method in a system called DeepG. First, we demonstrate that DeepG can certify robustness to significantly more complex transformations than both prior work and traditional bound propagation approaches based on relational abstractions. Second, we experimentally show that our method requires relatively small number of samples to converge to the optimal linear constraints. Third, we investigate the effectiveness of a variety of training methods to train a network provably robust to geometric transformations. Finally, we demonstrate that DeepG is scalable and can certify geometric robustness for large networks. We provide networks and code to reproduce the experiments in this chapter at `https://github.com/eth-sri/deepg/`.

EXPERIMENTAL SETUP We evaluate on image recognition datasets: MNIST [95], Fashion-MNIST [98] and CIFAR-10 [99]. For each dataset, we randomly select 100 images from the test set to certify. Among these 100 images, we discard all images that are misclassified even without any transformation. In all experiments for MNIST and Fashion-MNIST we evaluate a 3-layer convolutional neural network with 9 618 neurons, while for the more challenging CIFAR-10 dataset we consider a 4-layer convolutional network with 45 216 neurons. We certify robustness to composition of transformations such as rotation, translation, scaling, shearing and changes in brightness and contrast. All experiments except the one with large networks were performed on a desktop PC with 2 GeForce RTX 2080 Ti GPU-s and 16-core Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz.

COMPARISON WITH PRIOR WORK In the first set of experiments we certify robustness to geometric transformations and compare our results to prior work [56]. While they considered only rotations, we implemented their approach for other transformations and their compositions. This generalization is shown as Interval in Table 3.1. For each dataset and geometric transformation, we train a neural network using data augmentation with the transformation that we are certifying. Additionally, we use PGD adversarial training to obtain a network robust to noise which we later show significantly increases verifiability of the network.

We first measure the success of a randomized attack which samples 100 transformed images uniformly at random [82]. Then, we generate linear constraints using DeepG, as described in Section 3.3. Constraints produced

| | | Accuracy (%) | Attacked (%) | Certified (%) | |
|---|---|---|---|---|---|
| | | | | Interval [56] | DeepG |
| MNIST | R(30) | 99.1 | 0.0 | 7.1 | **87.8** |
| | T(2, 2) | 99.1 | 1.0 | 0.0 | **77.0** |
| | Sc(5), R(5), B(5, 0.01) | 99.3 | 0.0 | 0.0 | **34.0** |
| | Sh(2), R(2), Sc(2), B(2, 0.001) | 99.2 | 0.0 | 1.0 | **72.0** |
| Fashion-MNIST | Sc(20) | 91.4 | 11.2 | 19.1 | **70.8** |
| | R(10), B(2, 0.01) | 87.7 | 3.6 | 0.0 | **71.4** |
| | Sc(3), R(3), Sh(2) | 87.2 | 3.5 | 3.5 | **56.6** |
| CIFAR-10 | R(10) | 71.2 | 10.8 | 28.4 | **87.8** |
| | R(2), Sh(2) | 68.5 | 5.6 | 0.0 | **54.2** |
| | Sc(1), R(1), B(1, 0.001) | 73.2 | 3.8 | 0.0 | **54.4** |

Table 3.1: Comparison of DeepG which uses linear constraints with the baseline based on interval bound propagation. Here, R($\phi$) corresponds to rotations with angles between $\pm\phi$; T($x, y$), to translations between $\pm x$ pixels horizontally and between $\pm y$ pixels vertically; Sc($p$), to scaling the image between $\pm p$%; Sh($m$), to shearing with a shearing factor between $\pm m$%; and B($\alpha, \beta$), to changes in contrast between $\pm\alpha$% and brightness between $\pm\beta$.

| | Images certified (%) | | |
|---|---|---|---|
| | Interval | Polyhedra | DeepG |
| T(0.25) | 0 | 14 | **90** |
| Sc(4) | 0 | 23 | **75** |
| Sh(10) | 0 | 12 | **38** |

Table 3.2: Certification success rates of interval propagation [56], Polyhedra and DeepG (for translation, shearing and scaling).

by both our method and the interval baseline are given as an input to the state-of-the-art neural network verifier DeepPoly [56]. We invoke both methods for every split separately, with the same set of splits. In order to make results fully comparable, both methods are parallelized over pixels in the same way and the refinement parameter $k$ of interval propagation is chosen such that its runtime is roughly equal to the one of DeepG. Table 3.1 shows the results of our evaluation.

While interval propagation used in prior work can prove robustness for simpler transformations, it fails for more complex geometric transformations. For example, already for translation which has two parameters it does not succeed at certifying a single image. This shows that, in order to certify complex transformations, one has to capture relationships between

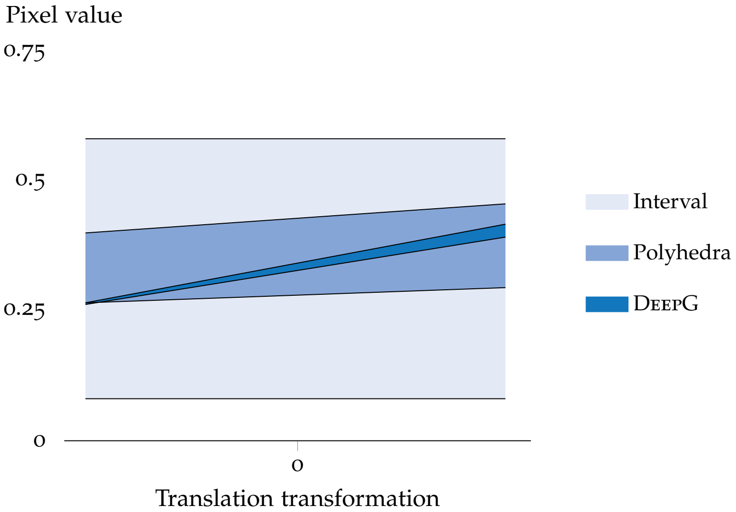


Figure 3.4: Translation transformation approximated using interval propagation, polyhedra and DeepG, for a representative pixel.

pixel values and transformation parameters using a relational abstraction. Linear constraints computed by DeepG provide a significant increase in certification precision, justifying the more involved method to compute the constraints.

COMPARISON WITH CUSTOM TRANSFORMERS To understand the benefits of our method further, we decided to construct a more advanced baseline than the interval propagation. In particular, we crafted specialized transformers for DeepPoly [56], which is a restriction of Polyhedra, to handle the operations used in geometric transformations. These kind of transformers have brought significant benefits over the interval propagation in certifying robustness to noise perturbations, and thus, we wanted to see what the benefits would be in our setting of geometric transformations. Concretely, we designed Polyhedra transformers for addition and multiplication which enables handling of geometric operations. Fig. 3.4 shows that relaxation with these transformers is significantly tighter than intervals. This also translates to higher certification rates compared to intervals, shown in Table 3.2. However, this method still fails to certify many images on which DeepG succeeds. This experiment shows that generating constraints for the entire composition of transformations as in DeepG is (expectedly)

| $n$ | $\epsilon$ | Approximation error | Certified(%) | Runtime(s) |
|---|---|---|---|---|
| 100 | 0.1 | 0.032 | 54.8 | 1.1 |
| 100 | 0.01 | 0.010 | 96.5 | 1.2 |
| 1000 | 0.001 | 0.006 | 97.8 | 4.9 |
| 10000 | 0.00001 | 0.005 | 98.2 | 46.1 |

Table 3.3: Speed of convergence of DeepG towards optimal linear bounds.

more effective than crafting transformers for individual operations of the transformation.

CONVERGENCE TOWARDS OPTIMAL BOUNDS While Thm. 1 shows that DeepG obtains optimal linear constraints in the limit, we also experimentally check how quickly our method converges in practice. For this experiment, we consider rotation between $-2^\circ$ and $2^\circ$, composed with scaling between $-5\%$ and $5\%$. We run DeepG while varying the number of samples used for the LP solver ($n$) and tolerance in Lipschitz optimization ($\epsilon$). In Table 3.3 we show the approximation error (average distance between lower and upper linear constraint), certification rate and time taken to compute the constraints. For instance, even with only 100 samples and $\epsilon = 0.01$ DeepG can certify almost every image in 1.2 seconds. While higher number of samples and smaller tolerance are necessary to obtain more precise bounds, they do not bring significant increase in certification rates.

COMPARISON OF DIFFERENT TRAINING METHODS Naturally, we would like to know how to train a neural network which is certifiably robust against geometric transformations. In this experiment, we evaluate effectiveness of a wide range of training methods to train a network certifiably robust to the translation up to 2 pixels in both $x$ and $y$ direction. While [82] train with adversarial examples and show that this leads to lower empirical success of an adversary, we are interested in a different question: can we train neural networks to be provably robust against geometric transformations?

We first train the same MNIST network as before, in a standard fashion, without any kind of defense. As expected, the resulting network shown in the first row of Table 3.4 is not robust at all – random attack can find many translations which cause misclassification of the network. To alleviate this problem, we incorporate data augmentation into training by randomly translating every image in a batch between -4 and 4 pixels before passing it

| | | Accuracy (%) | Attack success (%) | Certified (%) | |
|---|---|---|---|---|---|
| | | | | Interval [56] | DeepG |
| MNIST | Standard | 98.7 | 52.0 | 0.0 | 12.0 |
| | Augmented | 99.0 | 4.0 | 0.0 | 46.5 |
| | $L_\infty$-PGD | 98.9 | 45.5 | 0.0 | 20.2 |
| | $L_\infty$-DiffAI | 98.4 | 51.0 | 1.0 | 17.0 |
| | $L_\infty$-PGD + Augmented | **99.1** | **1.0** | 0.0 | **77.0** |
| | $L_\infty$-DiffAI + Augmented | 98.0 | 6.0 | **42.0** | 66.0 |

Table 3.4: Certification using DeepG for neural networks trained using different training techniques.

through the network. As a result, the network is significantly more robust against the attack, however there are still many images that we fail to certify. To make the network more amenable to certification, we consider two additional techniques. They are both based on the observation that convex relaxation of geometric transformations can be viewed as noise in the pixel values. To train a network which is robust to noise we consider adversarial training with projected gradient descent (PGD) [32] and provable defense based on DiffAI [51]. We also consider combination of these techniques with data augmentation.

Based on the results shown in Table 3.4, we conclude that training with PGD coupled with data augmentation achieves both highest accuracy and highest number of certified images. Training with DiffAI significantly increases certification rate for interval bound propagation, but has the drawback of significantly lower accuracy than other methods.

evaluation on large networks We evaluated whether DeepG can certify robustness of large CIFAR-10 networks with residual connections. We certified ResNet-Tiny and ResNet-18 from [100] with 312k and 558k neurons, respectively. As certifying these networks is challenging, we consider relatively small rotation between -2 and 2 degrees. As before, we generated constraints using both DeepG and interval bound propagation. This experiment was performed on a Tesla V100 GPU.

ResNet-Tiny was trained using PGD adversarial training and has standard accuracy 83.8%. Using the constraints from DeepG, we certify 91.1% of images while interval constraints allow us to certify only 1.3%. Average time for the verifier to certify or report failure is 528 seconds per image.

ResNet-18 was trained using DiffAI [100] and has standard accuracy 40.2%. In this case, using constraints from DeepG, the verifier certifies 82.2% of images. Similarly as before (see Table 3.4), training the network with

DiffAI also enables a high certification rate of 77.8% even using interval constraints. However, the drawback is that this network has low accuracy of only 40.2% compared to ResNet-Tiny trained with PGD which has 83.8% accuracy. On average, the verifier takes 1652 seconds per image. Here, generating the constraints for both networks took 25 seconds on average.

## 3.5 DISCUSSION

In this chapter, we tackled the problem of certifying robustness of neural networks to geometric transformations. This is an important problem as such class of transformations often appears naturally (e.g. with camera movement). While prior work considered a simple baseline where the outcome of the transformation is captured with interval constraints, our approach results in more precise convex relaxation. We introduced a new method for computing (optimal at the limit) linear constraints on geometric image transformations combining Lipschitz optimization and linear programming. We implemented the method in a system called DeepG and showed that it leads to significantly better certification precision in proving robustness against geometric perturbations than prior work, on both defended and undefended networks.

IMPACT Since the publication of our work on which this chapter is based, there has been much of further research in the area. Several papers have considered certifying geometric transformations using convex relaxations [101] or randomized smoothing [102, 103]. In our own work we also later showed that the techniques developed in this chapter generalize and can be used to obtain convex relaxations in other settings: spatial transformations [5], audio transformations [7] and point cloud transformations [8], ultimately leading to certifying robustness for a wider range of specifications.

# 4

# ADVERSARIAL TRAINING AND PROVABLE DEFENSES: BRIDGING THE GAP

Ever since the discovery of adversarial examples [29, 31], training robust neural networks has been one of the central challenges in machine learning. The initial training methods were based on including these examples in training, as part of so-called *adversarial training* [31, 32, 104]. Models trained with adversarial training were shown to be empirically robust against later developed attacks [35], but this is not fully satisfactory as we cannot *guarantee* that the model is indeed robust against *all* future attacks. This has motivated the development of *provable defenses* [60, 63] which provide such guarantees, but come with the drawback of lower accuracy. In this chapter, we address this problem by proposing a new provable defense based on the insights from adversarial training which leads to training networks with high accuracy and provable robustness.

KEY CHALLENGE The key challenge in this chapter is to carefully combine advantages of adversarial and provable training to create a new method for training highly accurate neural networks that are provably robust to adversarial examples.

PRIOR WORK Prior work has proposed two families of methods, *adversarial training* and *provable defenses*. Adversarial training [30, 31] provides a framework to augment the training procedure with adversarial inputs produced by an adversarial attack. Madry et al. [32] instantiated adversarial training using a strong iterative adversary and showed that their approach can train models which are highly robust against the strongest known adversarial attacks such as [33]. This method has also been able to train robust ImageNet models [105]. While promising, the main drawback of the method is that when instantiated in practice, via an approximation of an otherwise intractable optimization problem, it provides no guarantees – it does not produce a certificate that there are no possible adversarial attacks which could potentially break the model. To address this lack of guarantees, recent line of work on provable defenses [60, 61, 63] has proposed to train neural networks that are certifiably robust to a specific attacker threat model. However, these guarantees come at the cost of a significantly lower standard accuracy than models trained using adversarial training. This setting raises a natural question: can we leverage ideas from both, adversarial training techniques and provable defense methods, so to obtain models with high accuracy and certified robustness?

THIS CHAPTER: COMBINING ADVERSARIAL AND PROVABLE DEFENSES In this chapter, we take a step towards addressing this challenge. We show that it is possible to train more accurate and provably robust neural networks using the *same* convex relaxations as those used in existing, state-of-the-art provable defense methods, but with a new, different optimization procedure inspired by adversarial training. Our optimization works as follows: (i) to certify a property (e.g., robustness) of the network, the verifier produces a convex relaxation of all possible intermediate vector outputs in the neural network, then (ii) an adversary now searches over this (intermediate) convex region in order to find, what we refer to as a *latent adversarial example* – a concrete intermediate input contained in the relaxation that when propagated through the network causes a misclassification which prevents verification, and finally (iii) the resulting latent adversarial examples are now incorporated into our training scheme using adversarial training. Overall, we can see this method as bridging the gap between adversarial training and provable defenses (it can conceptually be instantiated with any convex relaxation). We experimentally show that the method is promising and results in a neural network with state-of-the-art 78.4% accuracy and 60.5% certified robustness on the challenging CIFAR-10 dataset with 2/255 $L_\infty$ perturbation (the best known existing results are 71.5% accuracy and 54.0% certified robustness [106]).

MAIN CONTRIBUTIONS Our key contributions are:

- A new method, which we refer to as *convex layerwise adversarial training* (COLT), that can train provably robust neural networks and conceptually bridges the gap between adversarial training and existing provable defense methods.

- Instantiation of convex layerwise adversarial training using linear convex relaxations used in prior work (accomplished by introducing a projection operator).

- Experimental results showing convex layerwise adversarial training can train neural network models which achieve both, state-of-the-art accuracy and certified robustness on CIFAR-10 with 2/255 $L_\infty$ perturbation.

- Complete implementation of our training and certification methods in a system which we release at `https://github.com/eth-sri/colt`.

Overall, we believe the method presented in this chapter is a promising step towards training models that enjoy both, higher accuracy and higher certification guarantees. An interesting item for future work would be to explore instantiations of the method with other convex relaxations than the one considered here.

## 4.1 RELATED WORK

We now discuss some of the closely related work on robustness of neural networks.

HEURISTIC ADVERSARIAL DEFENSES After the first introduction of adversarial examples [28, 29], defense mechanisms to train robust neural networks were built based on the inclusion of adversarial examples to the training set [30, 31]. Models trained using adversarial training with projected gradient descent (PGD) [32] were shown to be robust against the strongest known attacks [33]. This is in contrast to other defense mechanisms which have been broken by new attack techniques [34]. While models trained using adversarial training achieve robustness against strong adversaries, there are no guarantees that model is robust against any kind of adversarial attack under the threat model considered.

PROVABLE ADVERSARIAL DEFENSES There has also been considerable amount of work on methods to train classifiers with robustness guarantees. These approaches are typically based on Lipschitz regularization [59], linear [60] or semidefinite [61, 62] relaxations, hybrid zonotope [63] or interval bound propagation [64]. While these approaches obtain robustness guarantees, accuracy of these networks is relatively small and limits practical use of these methods.

There has also been recent work on certification of general neural networks, not necessarily trained in a special way. These methods are based on SMT solvers [44], abstract interpretation [36], mixed-integer linear programs [92], linear relaxations [38, 39, 80] or combinations of those [93, 94].

Another line of work proposes to replace neural networks with a randomized classifier [55, 65, 66] which comes with probabilistic guarantees on its robustness. While these approaches scale to larger datasets such as ImageNet (although with probabilistic instead of exact guarantees), their bounds come from the relationship between $L_2$ robustness and Gaussian distribution. In this chapter, we consider general verification problem where input is not necessarily limited to an $L_p$ ball, but arbitrary convex set, as explained in Section 4.2.

## 4.2 BACKGROUND

In this section, we introduce the background for this chapter.

THREAT MODEL While the method we propose in this chapter can be applied to any transformation $t_{\boldsymbol{\kappa}}$, to be comparable with prior work, we focus on the threat model with $l_\infty$ robustness. This threat model is instantiated as $\boldsymbol{x}' = t_{\boldsymbol{\kappa}}(\boldsymbol{x})$ where $\boldsymbol{x}'_{ij} = \boldsymbol{x}_{ij} + \kappa_{ij}$ and each $\kappa_{ij} \in [-\epsilon, \epsilon]$. Clearly, the resulting shape $\mathbb{S}_0$ is already defined with the linear constraints, as follows:

$$\mathbb{S}_0(\boldsymbol{x}) = \{\boldsymbol{x}' \mid \exists \boldsymbol{\kappa} \in \mathcal{H}, x_{ij} - \kappa_{ij} \leq x'_{ij} \leq x_{ij} + \kappa_{ij}\}, \tag{4.1}$$

so we can set $\mathbb{C}_{\text{in}}(\boldsymbol{x}) = \mathbb{S}_0(\boldsymbol{x})$.

PROVABLE DEFENSES As explained in Chapter 2, the key idea to train provably robust neural network is to replace the inner maximum in Eq. (2.2) with an upper bound. This results in an optimization as follows:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},y)\sim D} \overline{\mathcal{L}}(\boldsymbol{x}, y)$$

where $\max_{\boldsymbol{\kappa} \in \mathcal{H}} \mathcal{L}(h_\theta(t_{\boldsymbol{\kappa}}(\boldsymbol{x})), y) \leq \overline{\mathcal{L}}(\boldsymbol{x}, y)$.

This family of methods to train certified neural networks is based on the computation of an upper bound to the inner loss, as opposed to a lower bound computed for adversarial training. These methods are typically referred to as provable defenses as they provide guarantees on the robustness of the resulting network, under any kind of attack inside the threat model. An upper bound is typically computed using linear relaxations [60], interval propagation [64] or methods combining interval bounds and linear relaxations [63, 106]. However, these methods suffer from two disadvantages.

First, due to the convex relaxations, an upper bound on the loss is typically not tight and can be quite loose. However, we believe this is less of an issue due to the fact that interval relaxations were shown to experimentally be able to train more provably robust models than methods based on linear relaxations (which usually produce tighter bounds than intervals). For example, Mirman, Gehr, and Vechev [63] and Zhang et al. [106] report $\sim$ 28% robust accuracy using pure interval training on CIFAR-10 with perturbation 8/255 while [107] achieve 21% using linear relaxations.

Second, the way these methods construct the loss makes the relationship between the loss and the network parameters significantly more complex than in standard training. We investigated this further in [9] and showed that the complexity of the loss computation makes the objective function discontinuous and sensitive, resulting in a difficult optimization problem for training the network, meaning these training methods often converge to a suboptimal solution. Our experimental results confirm this – we substantially outperform existing methods both in terms of accuracy and certified robustness using the *same* linear relaxation, but a different optimization procedure.

CERTIFICATION VIA CONVEX RELAXATIONS A neural network consisting of $k$ layers and parameters $\theta$ is represented as a function $h_\theta$ where $h_\theta = h_\theta^k \circ h_\theta^{k-1} \cdots \circ h_\theta^1$ and $h_\theta^i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ denotes a transformation applied at hidden layer $i$. We also denote the function representing part of the neural network from layer $i$ to the final layer $k$ as $h_\theta^{i:k} = h_\theta^k \circ h_\theta^{k-1} \cdots \circ h_\theta^i$.

Our goal will be to prove a property on the output of the neural network, encoded via a linear constraint:

$$\boldsymbol{c}^T h_\theta(\boldsymbol{x}') + d < 0, \forall \boldsymbol{x}' \in \mathbb{S}_0(\boldsymbol{x}) \tag{4.2}$$

where $\boldsymbol{c}$ and $d$ are property specific vector and scalar values, respectively. This formulation is general enough to capture many interesting safety properties [57, 108], including robustness to $L_p$ perturbations. Note that in this chapter we will sometimes denote $\mathbb{C}_0(\boldsymbol{x}) = \mathbb{C}_{\text{in}}(\boldsymbol{x})$.

We now formally describe how provable defenses perform certification. We denote the set of possible intermediate concrete vectors at layer $i$ that can be obtained by propagating vector $\boldsymbol{x}' \in \mathbb{S}_0(\boldsymbol{x})$ through the network as $\mathbb{S}_i(\boldsymbol{x}) = h_\theta^i(\mathbb{S}_{i-1}(\boldsymbol{x})) \subseteq \mathbb{R}^{d_i}$. As it is difficult to explicitly compute the set $\mathbb{S}_i(\boldsymbol{x})$, a standard approach is to approximate it via a convex relaxation $\mathbb{C}_i(\boldsymbol{x})$. As the input set is already convex, there is no need to introduce a relaxation, and thus we set $\mathbb{C}_0(\boldsymbol{x}) = \mathbb{S}_0(\boldsymbol{x})$. Given a neural network layer $h^i_\theta$ which transforms one set of vectors into another, we represent its corresponding convex relaxation transformer as $g^i_\theta$. That is, $g^i_\theta$ will transform one convex set into another convex set. More formally, for any set $\mathbb{D} \subseteq \mathbb{R}^{d_{i-1}}$, $g^i_\theta(\mathbb{D})$ is convex and $h^i_\theta(\mathbb{D}) \subseteq g^i_\theta(\mathbb{D})$. Then, we recursively define the effect of $g^i_\theta$ on a convex relaxation as $\mathbb{C}_i(\boldsymbol{x}) = g^i_\theta(\mathbb{C}_{i-1}(\boldsymbol{x})) \subseteq \mathbb{R}^{d_i}$. Finally, to certify robustness using the obtained convex relaxation, it is enough to check whether all output vectors in $\mathbb{C}_k(\boldsymbol{x})$ satisfy the linear constraint in Equation 4.2. If this is true, then all output vectors in $\mathbb{S}_k(\boldsymbol{x})$ satisfy the constraint as well due to the fact that $\mathbb{S}_k(\boldsymbol{x}) \subseteq \mathbb{C}_k(\boldsymbol{x})$.

## 4.3 PROVABLE DEFENSE VIA CONVEX LAYERWISE ADVERSARIAL TRAINING

We now describe our convex layerwise adversarial training approach which yields a provable defense that bridges the gap between standard adversarial training and existing provable defenses.

**MOTIVATION: LATENT ADVERSARIAL EXAMPLES** Consider an already trained neural network model $h_\theta$ which we would like to certify using convex relaxations. A fundamental issue here is that certification methods based on convex relaxations can struggle to prove the target property (e.g., robustness) due to the iterative loss of precision introduced by the relaxation. More precisely, assume the neural network actually satisfies the property from Equation 4.2 for an input $\boldsymbol{x}$, meaning that $\boldsymbol{c}^T h_\theta(\boldsymbol{x}') + d < 0, \forall \boldsymbol{x}' \in \mathbb{S}_0(\boldsymbol{x})$. Naturally, this also implies that the neural network behaves correctly in the latent space of its first hidden layer in the region $\mathbb{S}_1(\boldsymbol{x})$. Formally, this means that $\boldsymbol{c}^T h^{2:k}_\theta(\boldsymbol{x}'_1) + d < 0, \forall \boldsymbol{x}'_1 \in \mathbb{S}_1(\boldsymbol{x})$. However, if one would use a certification method which replaces the region $\mathbb{S}_1(\boldsymbol{x})$ by its convex relaxation $\mathbb{C}_1(\boldsymbol{x})$, then it is possible that we would fail to certify our desired property. This is due to the fact that there may exist an input $\boldsymbol{x}'_1 \in \mathbb{C}_1(\boldsymbol{x}) \setminus \mathbb{S}_1(\boldsymbol{x})$ such that $\boldsymbol{c}^T h^{2:k}_\theta(\boldsymbol{x}'_1) + d \geq 0$. Of course, we could repeat the above thought experiment and possibly find more violating latent inputs in the set $\mathbb{C}_i(\boldsymbol{x}) \setminus \mathbb{S}_i(\boldsymbol{x})$ of any hidden layer $i$. The existence of points found in the difference between a convex relaxation and the true region is a fundamental reason for the failure of certification methods based on convex relaxations. For convenience, we refer to such points as *latent adversarial examples*. Next, we describe a method which trains the neural network in a way that aims to minimize the number of latent adversarial examples.

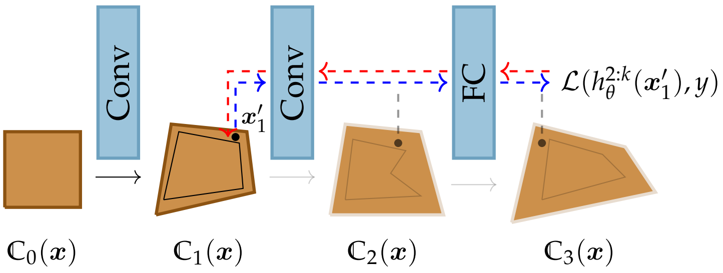

Conv
Conv
FC
$\boldsymbol{x}'_1$
$\mathcal{L}(h_\theta^{2:k}(\boldsymbol{x}'_1), y)$
$\mathbb{C}_0(\boldsymbol{x})$
$\mathbb{C}_1(\boldsymbol{x})$
$\mathbb{C}_2(\boldsymbol{x})$
$\mathbb{C}_3(\boldsymbol{x})$

FIGURE 4.1: An iteration of convex layerwise adversarial training. Latent adversarial example $\boldsymbol{x}'_1$ is found in the convex region $\mathbb{C}_1(\boldsymbol{x})$ and propagated through the rest of the layers in a forward pass, shown with the blue line. During backward pass, gradients are propagated through the same layers, shown with the red line. Note that the first convolutional layer does not receive any gradients.

**LAYERWISE PROVABLE OPTIMIZATION VIA CONVEX RELAXATIONS** Our key observation is that the two families of defense methods described earlier are in fact different ends of the same spectrum: methods based on *adversarial training* maximize the cross-entropy loss in the first convex region $\mathbb{C}_0(\boldsymbol{x})$ while *provable defenses* maximize the same loss, but in the last convex region $\mathbb{C}_k(\boldsymbol{x})$. Both methods then backpropagate the loss through the network and update the parameters using SGD. However, as explained previously, certification methods may fail even before the last layer due to the presence of latent adversarial examples in the difference of the regions $\mathbb{C}_i(\boldsymbol{x})$ and $\mathbb{S}_i(\boldsymbol{x})$. A natural question then is – can we leverage adversarial training so to eliminate latent adversarial examples from hidden layers and obtain a provable network?

To this end, we propose adversarial training in layerwise fashion. The initial phase of training is equivalent to adversarial training as used by Madry et al. [32]. In this phase in the inner loop we repeatedly find an input in $\mathbb{C}_0(\boldsymbol{x})$ which maximizes the cross-entropy loss and update the parameters of the neural network so to minimize this loss using SGD. Note that the outcome of this phase is a model which is highly robust against strong multi-step adversaries. However, certification of this fact often fails due to the previously mentioned loss of precision in the particular convex relaxation being used, which then leads to the existence of latent adversarial examples in the hidden layers.

The next step of our training method is visually illustrated in Figure 4.1. Here, we propagate the initial convex region through the first layer of the

**Algorithm 2** Convex layerwise adversarial training via convex relaxations

```
Input: $k$-layer network $h_\theta$, training set $(\mathcal{X}, \mathcal{Y})$, learning rate $\eta$, step size $\alpha$, inner steps $n$
for $l = 1$ to $k$ do
    for $i = 1$ to $n_{\text{epochs}}$ do
        Sample mini-batch $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_b, y_b)\} \sim (\mathcal{X}, \mathcal{Y})$
        Compute convex relaxations $\mathbb{C}_l(\boldsymbol{x}_1), \mathbb{C}_l(\boldsymbol{x}_2), ..., \mathbb{C}_l(\boldsymbol{x}_b)$
        Initialize $\boldsymbol{x}'_1 \sim \mathbb{C}_l(\boldsymbol{x}_1), \boldsymbol{x}'_2 \sim \mathbb{C}_l(\boldsymbol{x}_2), ..., \boldsymbol{x}'_b \sim \mathbb{C}_l(\boldsymbol{x}_b)$
        for $j = 1$ to $b$ do
            for step $= 1$ to $n$ do
                $\boldsymbol{x}'_j \leftarrow \Pi_{\mathbb{C}_l(\boldsymbol{x}_j)}(\boldsymbol{x}'_j + \alpha \nabla_{\boldsymbol{x}'_j} \mathcal{L}(h_\theta^{l+1:k}(\boldsymbol{x}'_j), y_j))$
            end for
        end for
        Update parameters $\theta \leftarrow \theta - \eta \cdot \frac{1}{b} \sum_{j=1}^{b} \nabla_\theta \mathcal{L}(h_\theta^{l+1:k}(\boldsymbol{x}'_j), y_j)$
    end for
    Freeze parameters $\theta_{l+1}$ of layer function $h_\theta^{l+1}$
end for
```

network and obtain the convex region $\mathbb{C}_1(\boldsymbol{x})$. We then solve the optimization problem to find a *concrete point* $\boldsymbol{x}'_1$ inside of $\mathbb{C}_1(\boldsymbol{x})$ which produces the maximum loss when this point is propagated further through the network (this forward pass is shown with the blue line). Finally, we backpropagate the final loss (red line) and update the parameters of the network so to minimize the loss. Critically, we do not backpropagate through the convex relaxation in the first layer as standard provable defenses do [60, 63, 64]. We instead freeze the first layer and stop backpropagation after the update of the second layer. Because of this, our optimization problem is significantly easier – the neural network only has to learn to behave well on the *concrete points* that were found in the convex region $\mathbb{C}_l(\boldsymbol{x})$. This can be viewed as an extension of the robust optimization method that Madry et al. [32] found to work well in practice.

We then proceed with the above process for later layers. Formally, this training process amounts to (approximately) solving the following min-max optimization problem at the $l$-th step:

$$\min_{\theta^{l+1:k}} \mathbb{E}_{(\boldsymbol{x},y)\sim D} \max_{\boldsymbol{x}'_l \in \mathbb{C}_l(\boldsymbol{x})} \mathcal{L}(h_\theta^{l+1:k}(\boldsymbol{x}'_l), y, \theta) \tag{4.3}$$

Note that for $l = 0$ this formulation is equivalent to the standard min-max formulation from Chapter 2 because $\mathbb{C}_0(\boldsymbol{x}) = \mathbb{S}_0(\boldsymbol{x})$. Our approach to solve this min-max optimization problem for every layer $l$ is shown in Algorithm 2. We initialize every batch by random sampling from the corresponding convex region. Then, in every iteration we use projected gradient descent (PGD) to maximize the inner loss in Equation 4.3. We first update $\boldsymbol{x}'_j$ in the direction of the gradient of the loss and then project it back to $\mathbb{C}_l(\boldsymbol{x}_j)$ using the projection operator $\Pi$. Note that this approach assumes the existence of an efficient projection method to the particular convex relaxation the method is instantiated with. In the next section, we show how to instantiate the training algorithm described above to a particular convex relaxation which is generally tighter than a hyperrectangle and where we derive an efficient projection operation.

## 4.4 CONVEX LAYERWISE ADVERSARIAL TRAINING USING LINEAR RELAXATIONS

So far we have described the general approach of convex layerwise adversarial training. Now we show how to instantiate it for a particular convex relaxation based on linear approximations. If instead one would use interval approximation [63, 64] as the convex relaxation, then all regions $\mathbb{C}_l(\boldsymbol{x})$ would be hyperrectangles and projection to these sets is fast and simple. However, the interval relaxation provides a coarse approximation which motivates the need to train with relaxations that provide tighter bounds. Thus, we consider linear relaxations which are generally tighter than those based on intervals.

In particular we leverage the same relaxation which was previously proposed in Weng et al. [39], Wong and Kolter [60], and Singh et al. [109] as an effective way to certify neural networks. Here, each convex region is represented as a set $\mathbb{C}_l(\boldsymbol{x}) = \{\boldsymbol{a}_l + \boldsymbol{A}_l\boldsymbol{e} \mid \boldsymbol{e} \in [-1, 1]^{m_l}\}$. Vector $\boldsymbol{a}_l$ represents the center of the set and the matrix $\boldsymbol{A}_l$ represents the affine transformation of the hypercube $[-1, 1]^{m_l}$. This representation is also known as zonotope abstraction [110]. The initial convex region $\mathbb{C}_0(\boldsymbol{x})$ is represented using $\boldsymbol{a}_0 = \boldsymbol{x}$ and $\boldsymbol{A}_0 = \epsilon \boldsymbol{I}_{d_0}$ is a diagonal matrix. Propagation of these convex regions through the network is out of the scope of this thesis – a full description can be found in [60] or [109]. At a high level, the convolutional and fully connected layers are handled by multiplying $\boldsymbol{A}_l$ and $\boldsymbol{a}_l$ by appropriate matrices. To handle the ReLU activation, we apply a convex relaxation that amounts to multiplying $\boldsymbol{A}_l$ and $\boldsymbol{a}_l$ by appropriately chosen diagonal matrices which depend on whether the ReLU is activated or not. Using this relaxation of ReLU, we recursively obtain all convex regions $\mathbb{C}_l(\boldsymbol{x})$. In practice, the term $\boldsymbol{A}_l\boldsymbol{e}$ can be computed without explicitly constructing matrix $\boldsymbol{A}_l$ because $\boldsymbol{A}_l\boldsymbol{e} = \boldsymbol{W}_l\boldsymbol{\Lambda}_{l-1}\boldsymbol{W}_{l-2}\cdots\boldsymbol{M}_0\boldsymbol{e}$. Due to the associativity of matrix multiplication, we can compute $\boldsymbol{A}_l\boldsymbol{e}$ by performing a chain of matrix-vector multiplications from right to left (instead of more expensive chain of matrix-matrix multiplications from left to right) and obtain vector $\boldsymbol{A}_l\boldsymbol{e}$.

PROJECTION TO LINEAR CONVEX REGIONS To use our training method we now need to instantiate Algorithm 2 with a suitable projection operator $\Pi_{\mathbb{C}_l(\boldsymbol{x})}$. The key insight here is that the vector $\boldsymbol{x}' \in \mathcal{C}_l(\boldsymbol{x})$ is uniquely determined by auxiliary vector $\boldsymbol{e} \in [-1,1]^{m_l}$ where $\boldsymbol{x}' = \boldsymbol{a}_l + \boldsymbol{A}_l\boldsymbol{e}$. Then instead of directly solving for $\boldsymbol{x}'$ which requires projecting to $\mathbb{C}_l(\boldsymbol{x})$, we can solve for $\boldsymbol{e}$ instead, which would uniquely determine $\boldsymbol{x}'$. Crucially, the domain of $\boldsymbol{e}$ is a hyperrectangle $[-1,1]^{m_l}$ which is easy to project to. To visualize this further we provide an example in Figure 4.2. The goal is to project the red point $\boldsymbol{x}'$ in the right picture to the convex region $\mathbb{C}_l(\boldsymbol{x})$. To project, we first perform change of variables to substitute $\boldsymbol{x}'$ with $\boldsymbol{e}$ and then project $\boldsymbol{e}$ to the square $[-1,1] \times [-1,1]$ to obtain the blue point $\Pi(\boldsymbol{e})$ on the left. Then, we again perform change of variables to obtain the blue point $\Pi(\boldsymbol{x}')$ on the right, which is the projection of $\boldsymbol{x}'$ we were looking for.

Based on these observations, we modify Line 7 of Algorithm 2 to first update the coefficients $\boldsymbol{e}_j$ using the following update rule: $\boldsymbol{e}_j \leftarrow clip(\boldsymbol{e}_j + \alpha\boldsymbol{A}_l^T\nabla_{\boldsymbol{x}_j'}\mathcal{L}(\boldsymbol{x}_j', y_j), -1, 1)$. Here $clip$ is function which thresholds its argument between -1 and 1, formally $clip(t, -1, 1) = \min(\max(t, -1), 1)$. This is followed by an update to $\boldsymbol{x}_j'$ via $\boldsymbol{x}_j' \leftarrow \boldsymbol{a}_l + \boldsymbol{A}_l\boldsymbol{e}_j$, completing the update step.

EFFICIENT COMPUTATION OF CONVEX REGIONS While our representation of convex regions with matrix $\boldsymbol{A}_l$ and vector $\boldsymbol{a}_l$ has clean mathematical properties, in practice, a possible issue is that the matrix $\boldsymbol{A}_l$ can grow to be quite large. Because of this, propagating it through the network can be memory intensive and prohibit the use of larger batches. To overcome this difficulty, we propose two methods.

First method is based on the observation that $\boldsymbol{A}_l$ is quite sparse. We start with a very sparse, diagonal matrix $\boldsymbol{A}_0$ at the input. After each convolution, an element of matrix $\boldsymbol{A}_{l+1}$ is non-zero only if there is a non-zero element inside of its convolutional kernel in matrix $\boldsymbol{A}_l$. We can leverage this obser-

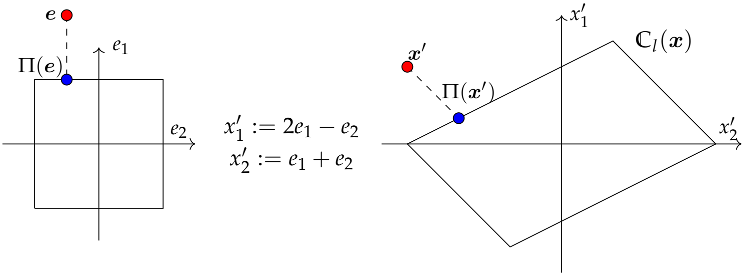


Figure 4.2: Projection to a region based on linear relaxation using change of variables.

vation to precompute positions of all non-zero elements in matrix $\boldsymbol{A}_{l+1}$ and compute their values using matrix multiplication.

Second method is based on the idea from [107]. Their key insight is that convex regions $\mathbb{C}_l(\boldsymbol{x})$ can be computed approximately during training. They propose to use random projections from [111] to estimate lower and upper bound for each neuron. While they operate in dual framework, we apply the same approach in our primal view. In our experiments, we use this method as it is more efficient than working with sparse representation. After training, during certification, we compute regions $\mathbb{C}_l(\boldsymbol{x})$ exactly without the estimation.

This optimization is critical to enabling training to take place altogether. An interesting item for future work is further optimizing the current relaxation (via a specialized GPU implementation) or developing more memory friendly relaxations, so to scale the training to larger networks.

REGULARIZATION One issue with the method presented in Section 4.3 is that there is no explicit mechanism that tries to make convex relaxation $\mathbb{C}_l(\boldsymbol{x})$ as close as possible to the exact region $\mathbb{S}_l(\boldsymbol{x})$ in order to avoid downstream loss of precision during later stages of the training, when layer $l$ is already frozen. To address this, we also incorporate additional regularizers similar to [112]: $L_1$ regularization and ReLU stability regularization. The purpose of $L_1$ regularization is to learn sparse weight matrices and the goal of ReLU stability regularization is to have fewer crossing ReLU units, both of which are beneficial for precision. Our ReLU stability is different from [112] since we directly minimize the area induced by our linear convex relaxation.

## 4.5 CERTIFICATION OF NEURAL NETWORKS

After training a neural network via convex layerwise adversarial training, our goal is to certify the target property (e.g., robustness). Here we leverage several recent advances in certification techniques which are not fast enough to be incorporated into the training procedure, but which can significantly speed up the certification or increase its precision.

REFINEMENT OF THE LINEAR APPROXIMATION The linear relaxation of ReLU, which we are using, is parameterized by slopes $\boldsymbol{\lambda}$ of the linear relaxation. Prior work which employed this relaxation [39, 60, 109] has chosen these slopes greedily by minimizing the area of the relaxation. During training we also choose $\boldsymbol{\lambda}$ in the same way. However, during certification, we can also optimize for the values of $\boldsymbol{\lambda}$ that give rise to the convex region inside of which the maximum loss is minimized. This optimization problem can be written as:

$$\min_{\boldsymbol{\lambda} \in [0,1]^{d_l}} \max_{\boldsymbol{x}' \in \mathbb{C}_l(\boldsymbol{x}; \boldsymbol{\lambda})} \mathcal{L}(h_\theta^{l+1:k}(\boldsymbol{x}'), y)$$

Solving this is computationally too expensive inside the training loop, but during certification it is feasible to approximate the solution. We solve for $\boldsymbol{\lambda}$ using the Adam optimizer and clipping the elements between 0 and 1 after each update. We remark that the idea of learning the slope is similar to [108] who propose to optimize dual variables in a dual formulation, however here we stay in the primal formulation.

CONVEX RELAXATIONS AND EXACT BOUND PROPAGATION During convex layerwise adversarial training we essentially train the network to be certified on all regions $\mathbb{C}_0(\boldsymbol{x}), \ldots, \mathbb{C}_k(\boldsymbol{x})$. While computing exact regions $\mathbb{S}_l(\boldsymbol{x}) \subseteq \mathbb{C}_l(\boldsymbol{x})$ is not feasible during training, we can afford it to some extent during certification. The idea is to first propagate the bounds using convex relaxations until one of the hidden layers $l$ and obtain a region $\mathbb{C}_l(\boldsymbol{x})$. If training was successful, there should not exist a concrete point $\boldsymbol{x}'_l \in \mathbb{C}_l(\boldsymbol{x})$ which, if propagated through the network, violates the correctness property in Equation 4.2. We can encode both, the property and the propagation of the *exact bounds* $\mathbb{S}_l(\boldsymbol{x})$ using a Mixed-Integer Linear Programming (MILP) solver. Note that we can achieve this because we represent the region $\mathbb{C}_l(\boldsymbol{x})$ using a set of linear constraints, which may not be possible for general convex shapes. We perform the MILP encoding using the formulation from

Tjeng, Xiao, and Tedrake [92]. It is usually possible to encode only the last two layers using MILP due to the poor scalability of these solvers for realistic network sizes. One further improvement we also include is to tighten the convex regions $\mathbb{C}_l(\boldsymbol{x})$ using refinement via linear programming as described in [93]. We remark that this combination of convex relaxation and exact bound propagation does not fall under the convex barrier to certification [81].

## 4.6 EXPERIMENTAL EVALUATION

We now present an evaluation of our training method on the challenging CIFAR-10 dataset. All of our code, datasets, trained models and scripts to reproduce the experiments can be found at `https://github.com/eth-sri/colt`.

EXPERIMENTAL SETUP We perform all experiments on a desktop PC using a single GeForce RTX 2080 Ti GPU and 16-core Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz. We implemented training and certification in PyTorch [113] and used Gurobi 9.0 [114] as a MILP solver.

NEURAL NETWORK ARCHITECTURE We evaluate on two architectures. First architecture is a 4-layer convolutional network: first 3 layers are convolutional layers with filter sizes 32, 32, 128, kernel sizes 3, 4, 4 and strides 1, 2, 2, respectively. Second architecture is a 3-layer convolutional network: first 2 layers are convolutional layers with filter sizes 32 and 128, kernel sizes 5 and 4, strides 2 and 2, respectively. In both architectures, convolutional layers are followed by a fully connected layer consisting of 250 hidden units. After each layer there is a ReLU activation. Final layer is a fully connected layer with 10 output neurons.

TRAINING During layerwise training we start with $\epsilon$ perturbation which is higher than the one we certify, and then decrease it by a certain factor when the training progresses to the next layer. In each stage of the training, we train for 200 epochs, starting from the same loss as in the previous stage and gradually annealing it to the loss of the current stage during first 60 epochs. We optimize using SGD with the initial learning rate 0.03 and after the initial 60 epochs we multiply the learning rate by 0.5 every 10 epochs. To find the best performing hyperparameters for training, we created a validation set consisting of random 5000 images from the training set and

| Method | Accuracy(%) | Certified Robustness(%) |
|---|---|---|
| COLT (this chapter) | **78.4** | **60.5** |
| CROWN-IBP [106] | 71.5 | 54.0 |
| Wong et al. [107] | 68.3 | 53.9 |
| IBP [64] | 70.2 | 50.0 |
| StableReLU [112] | 61.1 | 45.9 |
| DiffAI [100] | 62.3 | 45.5 |

TABLE 4.1: Evaluation on CIFAR-10 dataset with $L_\infty$ perturbation 2/255

used it to tune the hyperparameters with SigOpt [115]. We tuned batch size, initial $\epsilon$, factor to decrease $\epsilon$ after each layer, $L_1$ regularization and ReLU stability factors.

CERTIFICATION After training completes, we perform certification as follows: for every image, we first try to certify it using only linear relaxations (with the improvement of learned slopes, Section 4.5). If this fails, we encode the last layer as MILP and try again. Finally, if this fails we encode the ReLU activation after the last convolution using additional up to 50 binary variables and the rest using the triangle formulation [116]. We consider an image to be not certifiable if we fail to certify it using these methods. We always certify the full test set of 10 000 images.

COMPARISON TO PRIOR WORK We first train a robust network using our method for the $L_\infty$ perturbation 2/255. In this experiment, we used larger architecture with 3 convolutional and 1 fully connected layer. We perform convex layerwise adversarial training in 4 stages, for 200 epochs per stage, for a total of 800 epochs. The training takes 53, 164, 228, 250 seconds per epoch in the four respective stages of the training. It takes roughly 2 days to certify 10 000 images on a single GPU. Results are shown in Table 4.1. We always compare to the best reported and reproducible results in the literature on *any* architecture. We do not compare to smoothing-based approaches [55], as these provide probabilistic instead of exact guarantees. Extensions to [55] such as [66] also use additional existing techniques such as pre-training on ImageNet and unlabeled data which are orthogonal. We also do not compare to using cascades from [107], as this improvement is also orthogonal to the method here. Thus, we only consider their best *single* network architecture (inline with prior work Zhang et al. [106] which compares to a single architecture). We believe all methods listed in Table 4.1,

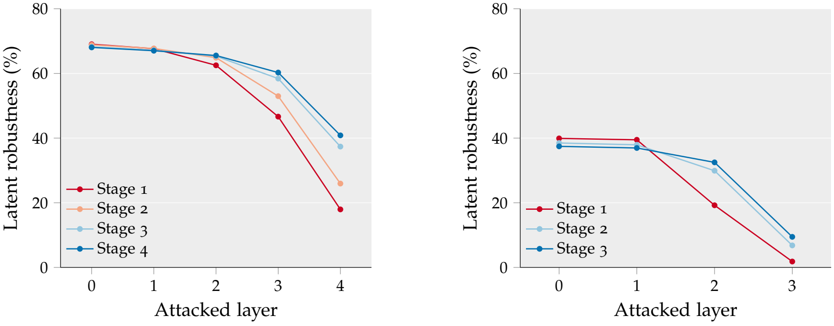
| Method | Accuracy(%) | Certified Robustness(%) |
|---|---|---|
| COLT (this chapter) | 51.7 | 27.5 |
| CROWN-IBP [106] | **54.5** | **30.5** |
| DiffAI [100] | 46.2 | 27.2 |
| Wong et al. [107] | 28.7 | 21.8 |
| StableReLU [112] | 40.5 | 20.3 |

Table 4.2: Evaluation on CIFAR-10 dataset with $L_\infty$ perturbation 8/255

including ours, would benefit from additional techniques such as cascades, pre-training and leveraging unlabeled data. Experimentally, we find that the neural network trained using our method substantially outperforms all existing approaches, both in terms of standard accuracy and certified robustness for 2/255. Note that here we are using the same linear relaxation as Wong et al. [107], but our optimization procedure is different and shows significant improvements over the one used in their work. We also remark that concurrent work Zhang et al. [106] reports 59.7% robustness against PGD which implies that even *empirical* robustness of their best model is lower than *certified* robustness of our network.

We also run the same experiment for $L_\infty$ perturbation 8/255 and present the results in Table 4.2. In this experiment, we used smaller architecture with 2 convolutional and 1 fully connected layer. Here we perform convex layerwise adversarial training in 3 stages, again for 200 epochs per stage, for a total of 600 epochs. The training takes 20, 67, 87 seconds per epoch in the three respective stages of the training. Here we do not include comparison with Gowal et al. [64] as their results were found to be not reproducible [100, 117, 118], and the best reproducible results for this method can be found in [106]. Here we substantially outperform all existing approaches except for the concurrent work of [106] whose method is based on a combination of interval and linear relaxation. We suspect that here the main issue is that our 3-layer network lacks capacity to solve this task, and capacity was found to be one of the key components necessary to obtain a robust classifier [32]. Due to promising results for 2/255, we believe achieving state-of-the-art results for 8/255 is very likely an issue of instantiating our method with a convex relaxation that is more memory efficient, which we believe is an interesting item for future work.

ANALYSIS Next, we analyze the effect of our convex layerwise adversarial training, also on CIFAR-10 with 2/255 and 8/255 $L_\infty$ perturbations. Recall



(a) CIFAR-10, 2/255 (b) CIFAR-10, 8/255

FIGURE 4.3: Effect of proposed convex layerwise adversarial training. After each stage of the training, we attack the model with a latent adversarial attack on each of the layers. Note that layer 0 represents standard PGD attack (attack in the input space).

that in the first stage, our training is equivalent to PGD [32] and in each of the successive stages, we freeze the current layer and retrain the rest of the network. In this experiment, we are interested in robustness of a model against latent adversarial attacks during each stage of the training. To perform this experiment, after each stage of the training we stored the intermediate model and ran latent adversarial attack on these models, on each of the layers. For latent adversarial attack, we perform PGD in the latent space, with 150 steps and step size of 0.01. Note that final models correspond to the models reported in Table 4.1 and Table 4.2.

The results are shown in Figure 4.3a and Figure 4.3b, for 2/255 and 8/255 perturbations, respectively. Each point represents success rate of latent adversarial attack on a trained model, where each model is shown in a different color. Red line shows the model trained after the first stage, which is equivalent to PGD training from Madry et al. [32]. As expected, this model is robust against the standard adversarial attack in the input space which is denoted as attack on layer 0. However, this Stage 1 model lacks robustness in the deeper layers which prevents us from certifying the robustness using convex relaxations. Using convex layerwise adversarial training, the model progressively becomes robust to perturbations in the deeper layers. For example, Figure 4.3a shows that final model, after Stage 4, has 41% robustness in layer 4 which is significant improvement over Stage 1 model which has only 18% robustness in layer 4. Note that this means we can certify 41% using only linear relaxation, and 60.5% when learning the slopes and encoding part of the network as MILP, which is explained in Section 4.5. We observe similar results for 8/255 perturbation shown in Figure 4.3b.

OTHER DATASETS To further evaluate our method, we also experimented with other datasets: Street House View Numbers (SVHN) and MNIST Handwritten Digits. On SVHN with $L_\infty$ perturbation 0.01 we also achieve state-of-the-art accuracy and certified robustness. On MNIST, we evaluated with perturbations 0.1 and 0.3. With $L_\infty$ perturbation 0.1 we achieve results comparable with best results from prior work, while with perturbation 0.3 our certified robustness is lower than the one achieved by approaches based on interval bound propagation [106]. We believe that, because of large perturbation of 0.3, random projections are imprecise and one would need to use the exact bounds which introduces much higher cost at runtime. This is also reflected in the poor performance of Wong et al. [107] on this benchmark, as we use the same random projections as their work. We believe that instantiating our method with a convex relaxation that is more memory friendly than what we used would likely yield better results in this experiment. These full results can be found in our paper [2].

## 4.7 DISCUSSION

In this chapter, we presented a new method to train certified neural networks. The key concept was to combine techniques from provable defenses using convex relaxations with those of adversarial training. Our method, named convex layerwise adversarial training (COLT), achieves state-of-the-art 78.4% accuracy and 60.5% certified robustness on CIFAR-10 with a 2/255 $L_\infty$ perturbation, significantly outperforming prior work when considering a single network (it also achieves competitive results on 8/255 $L_\infty$). The method is general and can be instantiated with *any* convex relaxation.

IMPACT There has been plenty of follow-up work on certified training. First, in Jovanović et al. [9] we closely investigated the underlying optimization problem induced by different convex relaxations. We showed that interval bounds, unlike more precise convex relaxations, induce continuous and non-sensitive objective function, indeed resulting in easier optimization problem. In another work [6] we showed that one can use similar techniques as in this chapter to boost certified robustness of any other network, while losing only a small amount of accuracy. Since the publication of our work, there has been significant progress in training methods with higher certified and natural accuracy [21, 22, 23]. Many of these new ideas are based on similar insights as this chapter, namely connecting heuristic and provable defenses.

# 5

# LEARNING CERTIFIED INDIVIDUALLY FAIR REPRESENTATIONS

This chapter is based on an observation that *individual fairness* can be phrased as a special form of robustness. Namely, individual fairness requires that similar individuals are classified similarly which is technically very related to robustness considered in Chapter 3 and Chapter 4. This is an important problem as machine learning is often used in sensitive domains (e.g., crime risk assessment [119], ad targeting [120], and credit scoring [121]) where making sure that models do not reinforce human bias, discriminate, or lack fairness [122, 123, 124] is of utmost importance. Moreover, as fairness appears in a variety of settings (typically split into pre-processing, in-processing, post-processing), here we focus on *fair representation learning* which allows pre-processing data into a new representation that can be used for a variety of tasks.

KEY CHALLENGE Formally, we consider a *fair representation learning* setting where machine learning model $M \colon \mathbb{R}^n \to \mathbb{R}^o$ is composed of two parts: an encoder $f_\psi \colon \mathbb{R}^n \to \mathbb{R}^k$, provided by the data producer, and a classifier $h_\theta \colon \mathbb{R}^k \to \mathbb{R}^o$, provided by the data consumer, with $\mathbb{R}^k$ denoting the latent space. A central challenge then is to enforce and prove individual fairness in that setting. That is, to both learn an individually fair representation and to certify that individual fairness is actually satisfied across the end-to-end model $M$ without compromising the independence of the data producer and the data consumer.

THIS CHAPTER In this chapter, we propose the first method for addressing the above challenge. At a high level, our approach is based on the two key ingredients: (i) recent advances in training machine learning models with logical constraints [125], (ii) new methods for proving that constraints are satisfied [45], similar to the techniques presented in Chapter 3 and Chapter 4. Together, these ideas open the possibility for learning certified individually fair models.

Concretely, we identify a practical class of individual fairness definitions captured via declarative fairness constraints. Such a fairness constraint is a binary similarity function $\phi \colon \mathbb{R}^n \times \mathbb{R}^n \to \{0, 1\}$, where $\phi(\boldsymbol{x}, \boldsymbol{x}')$ evaluates to

1 if and only if two individuals $\boldsymbol{x}$ and $\boldsymbol{x}'$ are similar (e.g., if all their attributes except for race are the same). By working with declarative constraints, data regulators can now express interpretable, domain-specific notions of similarity, a problem known to be challenging [67, 126, 127, 128, 129, 130]. Similarly as in robustness where we considered neighborhood $\mathbb{S}_0(\boldsymbol{x})$, here the neighborhood of all similar individuals is determined through the constraint $\phi$ and denoted as $\mathbb{S}_0^{\phi}(\boldsymbol{x})$.

Given the fairness constraint $\phi$, we can now train an individually fair representation and use it to obtain a certificate of individual fairness for the end-to-end model. For training, the data producer can employ our framework to learn an encoder $f_{\psi}$ with the goal that two individuals satisfying $\phi$ should be mapped close together in $\ell_\infty$-distance in latent space. As a consequence, individual fairness can then be certified for a data point in two steps: first, the data producer computes a convex relaxation of the latent set of similar individuals and passes it to the data consumer. Second, the data consumer certifies individual fairness by proving local robustness within the convex relaxation. Importantly, the data consumer can now perform *modular* certification: it does not need to know the fairness constraint $\phi$ and the concrete data point $x$.

Our experimental evaluation on several datasets and fairness constraints shows a substantial increase (up to 72.6%) of certified individuals (unseen during training) when compared to standard representation learning.

MAIN CONTRIBUTIONS Our key contributions are:

- A practical family of similarity notions for individual fairness defined via interpretable logical constraints.
- A method to learn individually fair representations (defined in an expressive logical fragment), which comes with provable certificates.
- An end-to-end implementation of our method in an open-source tool called LCIFR, together with an extensive evaluation on several datasets, constraints, and architectures. We make LCIFR publicly available at `https://github.com/eth-sri/lcifr`.

## 5.1 OVERVIEW

This section provides a high-level overview of our approach, with the overall flow shown in Fig. 5.1.

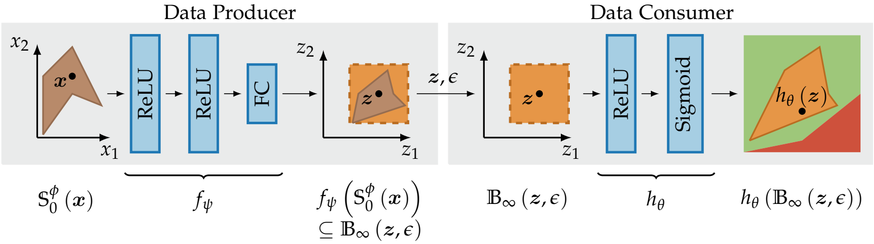

Data Producer
Data Consumer
$x_2$
$x_1$
$\boldsymbol{x}$
ReLU
ReLU
FC
$z_2$
$z_1$
$\boldsymbol{z}$
$\boldsymbol{z}, \epsilon$
ReLU
Sigmoid
$h_\theta(\boldsymbol{z})$
$\mathbb{S}_0^\phi(\boldsymbol{x})$
$f_\psi$
$f_\psi\left(\mathbb{S}_0^\phi(\boldsymbol{x})\right) \subseteq \mathbb{B}_\infty(\boldsymbol{z}, \epsilon)$
$\mathbb{B}_\infty(\boldsymbol{z}, \epsilon)$
$h_\theta$
$h_\theta(\mathbb{B}_\infty(\boldsymbol{z}, \epsilon))$

FIGURE 5.1: Conceptual overview of our framework. The left side shows the component corresponding to the data producer who learns an encoder $f_\psi$ which maps the entire set of individuals $\mathbb{S}_0^\phi(\boldsymbol{x})$ that are similar to individual $\boldsymbol{x}$, according to the similarity notion $\phi$, to points near $f_\psi(\boldsymbol{x})$ in the latent space. The data producer then computes an $\ell_\infty$-bounding box $\mathbb{B}_\infty$ around the latent set of similar individuals $f_\psi(\mathbb{S}_0^\phi(\boldsymbol{x}))$ with center $\boldsymbol{z} = f_\psi(\boldsymbol{x})$ and radius $\epsilon$ and passes it to the data consumer. The data consumer receives the latent representation $\boldsymbol{z}$ and radius $\epsilon$, trains a classifier $h_\theta$, and certifies that the entire $\ell_\infty$-ball centered around $\boldsymbol{z}$ with radius $\epsilon$ is classified the same (green color shows fair output region).

As introduced earlier, our setting consists of three parties. The first party is a data regulator who defines similarity measures for the input and the output denoted as $\phi$ and $\mu$, respectively. The properties $\phi$ and $\mu$ are problem-specific and can be expressed in a rich logical fragment which we describe later in §5.3. For example, for classification tasks $\mu$ could denote equal classification (i.e. $\mu(M(\boldsymbol{x}), M(\boldsymbol{x}')) = 1 \iff M(\boldsymbol{x}) = M(\boldsymbol{x}')$) or classifying $M(\boldsymbol{x})$ and $M(\boldsymbol{x}')$ to the same label group; for regressions tasks $\mu$ could evaluate to 1 if $\|M(\boldsymbol{x}) - M(\boldsymbol{x}')\| \leq 0.1$ and $0$ otherwise. We focus on equal classification in the classification setting for the remainder of this chapter.

The goal of treating similar individuals as similarly as possible can then be formulated as finding a classifier $M$ which maximizes

$$\mathbb{E}_{\boldsymbol{x} \sim D}\left[\forall \boldsymbol{x}' \in \mathbb{R}^n : \phi(\boldsymbol{x}, \boldsymbol{x}') \implies \mu(M(\boldsymbol{x}), M(\boldsymbol{x}'))\right], \tag{5.1}$$

where $D$ is the underlying data distribution (we assume a logical expression evaluates to 1 if it is true and to 0 otherwise). As usual in machine learning, we approximate this quantity with the empirical risk, by computing the percentage of individuals $\boldsymbol{x}$ from the test set for which we can certify that

$$\forall \boldsymbol{x}' \in \mathbb{S}_0^{\phi}(\boldsymbol{x}) : \mu(M(\boldsymbol{x}), M(\boldsymbol{x}')), \tag{5.2}$$

where $\mathbb{S}_0^{\phi}(\boldsymbol{x}) = \{\boldsymbol{x}' \in \mathbb{R}^n \mid \phi(\boldsymbol{x}, \boldsymbol{x}')\}$ denotes the set of all points similar to $\boldsymbol{x}$. Note that $\mathbb{S}_0^{\phi}(\boldsymbol{x})$ generally contains an infinite number of individuals. In Fig. 5.1, $\mathbb{S}_0^{\phi}(\boldsymbol{x})$ is represented as a brown shape, and $\boldsymbol{x}$ is shown as a single point inside of $\mathbb{S}_0^{\phi}(\boldsymbol{x})$.

The key idea of our approach is to train the encoder $f_\psi$ to map point $\boldsymbol{x}$ and all points $\boldsymbol{x}' \in \mathbb{S}_0^{\phi}(\boldsymbol{x})$ close to one another in the latent space with respect to $\ell_\infty$-distance, specified as

$$\phi\left(\boldsymbol{x}, \boldsymbol{x}'\right) \implies ||f_\psi(\boldsymbol{x}') - f_\psi(\boldsymbol{x})||_\infty \leq \delta, \tag{5.3}$$

where $\delta$ is a tunable parameter of the method, determined in agreement between producer and consumer (we could also use another $\ell_p$-norm). If the encoder indeed satisfies Eq. (5.3), the data consumer, potentially indifferent to the fairness constraint, can then train a classifier $h_\psi$ independently of the similarity notion $\phi$. Namely, the data consumer only has to train $h_\psi$ to be robust to perturbations up to $\delta$ in $\ell_\infty$-norm, which can be solved via standard min-max optimization, discussed in §5.3.

We now explain our end-to-end inference with provable certificates for encoder $f_\psi$ and classifier $h_\theta$.

PROCESSING THE PRODUCER MODEL Given a data point $\boldsymbol{x}$, we first propagate both $\boldsymbol{x}$ and its set of similar points $\mathbb{S}_0^{\phi}(\boldsymbol{x})$ through the encoder, as shown in Fig. 5.1, to obtain the latent representations $\boldsymbol{z} = f_\psi(\boldsymbol{x})$ and $f_\psi(\mathbb{S}_0^{\phi}(\boldsymbol{x}))$. As Eq. (5.3) may not hold for the particular $\boldsymbol{x}$ and $\delta$ due to the stochastic nature of training, we compute the smallest $\ell_\infty$-bounding box of radius $\epsilon$ such that $f_\psi(\mathbb{S}_0^{\phi}(\boldsymbol{x})) \subseteq \mathbb{B}_\infty(\boldsymbol{z}, \epsilon) := \{\boldsymbol{z}' \mid \|\boldsymbol{z} - \boldsymbol{z}'\|_\infty \leq \epsilon\}$. This $\ell_\infty$-bounding box with center $\boldsymbol{z}$ and radius $\epsilon$ is shown as orange in Fig. 5.1.

PROCESSING THE CONSUMER MODEL Next, we provide the latent representation $\boldsymbol{z}$ and the radius $\epsilon$ to the data consumer. The data consumer then knows that all points similar to $\boldsymbol{x}$ are in the $\ell_\infty$-ball of radius $\epsilon$, but does not need to know the similarity constraint $\phi$ nor the particular shape $f_\psi(\mathbb{S}_0^{\phi}(\boldsymbol{x}))$. The key observation is the following: if the data consumer can prove its classifier $h_\theta$ is robust to $\ell_\infty$-perturbations up to $\epsilon$ around $\boldsymbol{z}$, then the end-to-end classifier $M = h_\theta \circ f_\psi$ satisfies individual fairness at $\boldsymbol{x}$ with respect to the similarity rule $\phi$ imposed by the data regulator.

There are two central technical challenges we need to address. The first challenge is how to train an encoder to satisfy Eq. (5.3), while not making any domain-specific assumptions about the point $\boldsymbol{x}$ or the similarity constraint $\phi$. The second challenge is how to provide a certificate of individual fairness for $\boldsymbol{x}$, which requires both computing the smallest radius $\epsilon$ such that $f_\psi(\mathbb{S}_0^\phi(\boldsymbol{x})) \subseteq \mathbb{B}_\infty(\boldsymbol{z}, \epsilon)$, as well as certifying $\ell_\infty$-robustness of the classifier $h_\theta$.

To train an encoder, we build on DL2 [125], which provides a translation from logical constraints $\phi$ to a differentiable loss function. The training of the encoder network can then be formulated as a min-max optimization problem, which alternates between (i) searching for counterexamples $\boldsymbol{x}' \in \mathbb{S}_0^\phi(\boldsymbol{x})$ that violate Eq. (5.3), and (ii) training $f_\psi$ on the counterexamples. We employ gradient descent to minimize a joint objective composed of a classification loss and the constraint loss obtained from translating Eq. (5.3). Once no more counterexamples are found, we can conclude the encoder empirically satisfies Eq. (5.3). We discuss the detailed procedure in §5.3.

We compute a certificate for individual fairness in two steps. First, to provide guarantees on the latent representation generated by the encoder $f_\psi$, we solve the optimization problem

$$\epsilon = \max_{\boldsymbol{x}' \in \mathbb{S}_0^\phi(\boldsymbol{x})} ||\boldsymbol{z} - f_\psi(\boldsymbol{x}')||_\infty.$$

Recall that the set $\mathbb{S}_0^\phi(\boldsymbol{x})$ generally contains an infinite number of individuals $\boldsymbol{x}'$, and thus this optimization problem cannot be solved by simple enumeration. In §5.4 we show how this optimization problem can be encoded as a mixed-integer linear program (MILP) and solved using off-the-shelf MILP solvers. After obtaining $\epsilon$, we certify local robustness of the classifier $h_\theta$ around $\boldsymbol{z} = f_\psi(\boldsymbol{x})$ by proving (using MILP) that for each $\boldsymbol{z}'$ where $||\boldsymbol{z}' - \boldsymbol{z}|| \leq \epsilon$, the classification results of $h_\theta(\boldsymbol{z}')$ and $h_\theta(\boldsymbol{z})$ coincide. Altogether, this implies the overall model $M = h_\theta \circ f_\psi$ satisfies individual fairness for $\boldsymbol{x}$. Finally, note that since the bounding box $\mathbb{B}\left(\boldsymbol{z}, \epsilon\right)$ is a convex relaxation of the latent set of similar individuals $f_\psi(\mathbb{S}_0^\phi(\boldsymbol{x}))$, the number of individuals for which we can obtain a certificate is generally lower than the number of individuals that actually satisfy Eq. (5.2).

## 5.2 RELATED WORK

We now discuss additional related work that was not discussed in Chapter 2, more specifically the works on learning individually fair representations and individual fairness in other contexts.

LEARNING INDIVIDUALLY FAIR REPRESENTATIONS Most work so far focuses on learning representations that satisfy statistical notions of fairness, but there has also been some recent work on learning individually fair representations. These works learn fair representations with alternative definitions of individual fairness based on Wasserstein distance [77, 128], fairness graphs [127], or distance measures [126]. A different line of work has investigated leaning the fairness metric from data [128, 129, 130, 131]. In contrast, we define individual fairness via interpretable logical constraints. Finally, recent works [132, 133, 134] studied the task of learning representations that are robust to (adversarial) perturbations, i.e., all similar individuals in our case, however not in the context of fairness. Many of the above methods for learning (individually) fair representations employ nonlinear components [67, 126], graphs [127], or sampling [70, 78] and can thus not be efficiently certified, unlike the neural networks that we consider in our work.

INDIVIDUAL FAIRNESS IN OTHER CONTEXTS While we focus on learning fair representations, other lines of work have investigated individual fairness in the context of clustering [135, 136], causal inference [137, 138, 139, 140], composition of individually fair classifiers [141, 142], and differential privacy (DP) [143, 144, 145]. The close relationship between individual fairness and DP has been discussed in previous work (see, e.g., [143]). However, DP crucially differs from our work in that it obtains a probabilistic fairness guarantee, similar to McNamara, Ong, and Williamson [146] mentioned above, whereas we compute absolute fairness guarantees for every data point. The most natural way to employ DP for a representation learning approach, like LCIFR, would be to make the data producer model $f_\theta$ differentially private for a neighborhood that encodes $S_\phi$, by adding noise inside the computation of $f_\theta$. If one can achieve DP for the neighborhood $S_\phi$ (a non-trivial challenge), the data consumer model can then be seen as a post-processing step, which with the right robustness certificate yields a probabilistic guarantee of Eq. (5.2).

## 5.3 LEARNING INDIVIDUALLY FAIR REPRESENTATIONS

We now present our method for learning individually fair representations with respect to the property $\phi$. To illustrate our method, we consider the case where the regulator proposes the following similarity constraint:

$$\phi(\boldsymbol{x}, \boldsymbol{x}') := \bigwedge_{i \in \mathrm{Cat} \setminus \{\mathrm{race}\}} (x_i = x'_i) \bigwedge_{j \in \mathrm{Num}} |x_j - x'_j| \leq \alpha.$$

According to $\phi$, individual $\boldsymbol{x}'$ is considered similar to $\boldsymbol{x}$ if: (i) all categorical attributes except for race are equal to those of $\boldsymbol{x}$, and (ii) all numerical attributes (e.g., income) of $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ by at most $\alpha$. Thus, under $\phi$, the similarity of individuals $\boldsymbol{x}$ and $\boldsymbol{x}'$ does not depend on their respective races. Note that, since $\phi$ is binary, $\boldsymbol{x}$ and $\boldsymbol{x}'$ are either considered similar or not which is in line with the typical use-case in classification where two individuals are either classified to the same label or not. Moreover, such logical formulas (of reasonable size) are generally considered humanly readable and are thus investigated in the interpretable machine learning community (e.g., for decision trees [147]).

ENFORCING INDIVIDUAL FAIRNESS To learn a representation that satisfies $\phi$, we build on the recent work DL2 [125]. Concretely, we aim to enforce the following constraint on the encoder $f_\psi$ used by the data producer:

$$\phi(\boldsymbol{x}, \boldsymbol{x}') \implies \| f_\psi(\boldsymbol{x}) - f_\psi(\boldsymbol{x}') \|_\infty \leq \delta, \tag{5.4}$$

where $\delta$ is a tunable constant, determined in agreement between the data producer and the data consumer. With DL2, this implication can be translated into a non-negative, differentiable loss $\mathcal{L}(\phi)$ such that $\mathcal{L}(\phi)(\boldsymbol{x}, \boldsymbol{x}') = 0$ if and only if the implication is satisfied. Here, we denote $\omega(\boldsymbol{x}, \boldsymbol{x}') := \| f_\psi(\boldsymbol{x}) - f_\psi(\boldsymbol{x}') \|_\infty \leq \delta$ and translate the constraint in Eq. (5.4) as

$$\mathcal{L}(\phi \implies \omega) = \mathcal{L}(\neg\phi \vee \omega) = \mathcal{L}(\neg\phi) \cdot \mathcal{L}(\omega),$$

where negations are propagated through constraints via standard logic. Moreover, we have

$$\begin{aligned} \mathcal{L}(\omega)(\boldsymbol{x}, \boldsymbol{x}') &= \mathcal{L}\left(\| f_\psi(\boldsymbol{x}) - f_\psi(\boldsymbol{x}') \|_\infty \leq \delta\right) \\ &= \max\left\{\| f_\psi(\boldsymbol{x}) - f_\psi(\boldsymbol{x}') \|_\infty - \delta, 0\right\}. \end{aligned}$$

Similarly, conjunctions $\mathcal{L}\left(\phi' \wedge \phi''\right)$ would be translated as $\mathcal{L}\left(\phi'\right) + \mathcal{L}\left(\phi''\right)$, and we refer interested readers to the original work [125] for further details on the translation.

Using this differentiable loss, the data producer can now approximate the problem of finding an encoder $f_\psi$ that maximizes the probability that the constraint $\phi \implies \omega$ is satisfied for all individuals via the following min-max optimization problem (defined in two steps): First, we find a counterexample

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}' \in \mathbb{S}_0^{\phi}(\boldsymbol{x})}{\arg\min}\, \mathcal{L}\left(\neg\left(\phi \implies \omega\right)\right)\left(\boldsymbol{x}, \boldsymbol{x}'\right).$$

Recall that $\mathbb{S}_0^{\phi}\left(\boldsymbol{x}\right) = \left\{\boldsymbol{x}' \in \mathbb{R}^n \mid \phi\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right\}$ denotes the set of all individuals similar to $\boldsymbol{x}$ according to $\phi$. Then, in the second step, we find the parameters $\psi$ that minimize the constraint loss at $\boldsymbol{x}^*$:

$$\underset{\psi}{\arg\min}\, \mathbb{E}_{\boldsymbol{x} \sim D}\left[\mathcal{L}\left(\phi \implies \omega\right)\left(\boldsymbol{x}, \boldsymbol{x}^*\right)\right].$$

Note that in the outer loop, we are finding parameters $\psi$ that minimize the loss of the original constraint from Eq. (5.4), while in the inner loop, we are finding a counterexample $\boldsymbol{x}^*$ by minimizing the loss corresponding to the negation of this constraint. We use Adam [148] for optimizing the outer problem. For the inner minimization problem, Fischer et al. [125] further refine the loss by excluding constraints that have closed-form analytical solutions, e.g., $\max\left\{\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_\infty - \delta, 0\right\}$ which can be minimized by projecting $\boldsymbol{x}'$ onto the $\ell_\infty$-ball of radius $\delta$ around $\boldsymbol{x}$. The resulting objective is thus

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}' \in \mathbb{C}}{\arg\min}\, \mathcal{L}\left(\rho\right)\left(\boldsymbol{x}, \boldsymbol{x}'\right),$$

where $\mathbb{C}$ is the convex set and $\rho$ is $\neg\left(\phi \implies \omega\right)$ without the respective constraints. It has been shown [32] that such an objective can be efficiently solved with Projected Gradient Descent (PGD).

DL2 does not provide a meaningful translation for categorical constraints, which are essential to fairness, which we derive separately.

PREDICTIVE UTILITY OF THE REPRESENTATION Recall that our method is modular in the sense that the data producer and the data consumer models are learned separately. Thus, the data producer needs to ensure that the latent representation remains informative for downstream applications (represented by the data consumer model $h_\theta$). To that end, the data producer additionally trains a classifier $q\colon \mathbb{R}^k \to \mathbb{R}^o$ that tries to predict the target label $y$ from the latent representation $\boldsymbol{z} = f_\psi(\boldsymbol{x})$. Thus, the data producer seeks to jointly train the encoder $f_\psi$ and classifier $q$ to minimize the combined objective

$$\underset{f_\psi, q}{\arg\min}\, \mathbb{E}_{\boldsymbol{x},y} \left[ \mathcal{L}_C \left( q \left( f_\psi \left( \boldsymbol{x} \right) \right), y \right) + \gamma \mathcal{L}_F \left( \boldsymbol{x}, f_\psi(\boldsymbol{x}) \right) \right], \tag{5.5}$$

where $\mathcal{L}_C$ is any suitable classification loss (e.g., cross-entropy), $\mathcal{L}_F$ is the fairness constraint loss obtained via DL2, and the hyperparameter $\gamma$ balances the two objectives.

TRAINING ROBUST CLASSIFIER $h_\theta$ We assume the encoder $f_\psi$ has been trained to maintain predictive utility and satisfy Eq. (5.4). Recall that, given this assumption, the data consumer who wants to ensure classifier $h_\theta$ is individually fair, only needs to ensure local robustness of the classifier for perturbations up to $\delta$ in $l_\infty$-norm. This is a standard problem in robust machine learning [149] and can be solved via min-max optimization, recently found to work well for neural network models [32]:

$$\min_{\theta} \mathbb{E}_{\boldsymbol{z} \sim \mathcal{D}_{\boldsymbol{z}}} \left[ \max_{\boldsymbol{\pi} \in [\pm\delta]^k} \mathcal{L}_C \left( h_\theta(\boldsymbol{z} + \boldsymbol{\pi}), y \right) \right],$$

where $D_{\boldsymbol{z}}$ is the latent distribution obtained by sampling from $\mathcal{D}$ and applying the encoder $f_\psi$, and $\mathcal{L}_C$ is a suitable classification loss. The optimization alternates between: (i) trying to find $\boldsymbol{\pi} \in [-\delta, \delta]^k$ that maximizes $\mathcal{L}_C \left( h_\theta(\boldsymbol{z} + \boldsymbol{\pi}), y \right)$, and (ii) updating $\theta$ to minimize $\mathcal{L}_C \left( h_\theta(\boldsymbol{z} + \boldsymbol{\pi}), y \right)$ under such worst-case perturbations $\boldsymbol{\pi}$.

## 5.4 CERTIFYING INDIVIDUAL FAIRNESS

In this section we discuss how the data consumer can compute a certificate of individual fairness for their model $h_\theta$ trained on the latent representation (as described in §5.3 above). We split this process into two steps: (i) the data producer propagates a data point $\boldsymbol{x}$ through the encoder to obtain $\boldsymbol{z} = f_\psi(\boldsymbol{x})$ and computes the radius $\epsilon$ of the smallest $\ell_\infty$-ball around $\boldsymbol{z}$ that contains the latent representations of all similar individuals $f_\psi\left(\mathbb{S}_0^\phi(\boldsymbol{x})\right)$, i.e., $f_\psi\left(\mathbb{S}_0^\phi(\boldsymbol{x})\right) \subseteq \mathbb{B}_\infty(\boldsymbol{z}, \epsilon)$, and (ii) the data consumer checks whether all points in the latent space that differ by at most $\epsilon$ from $\boldsymbol{z}$ are classified to the same label, i.e., $h_\theta(\boldsymbol{z}) = h_\theta(\boldsymbol{z}')$ for all $\boldsymbol{z}' \in \mathbb{B}_\infty(\boldsymbol{z}, \epsilon)$. We now discuss both of these steps.

### 5.4.1 *Certifying Latent Similarity*

To compute the minimum $\epsilon$ which ensures that $f_\psi\left(\mathbb{S}_0^\phi(\boldsymbol{x})\right) \subseteq \mathbb{B}_\infty(\boldsymbol{z}, \epsilon)$, the data producer models the set of similar individuals $\mathbb{S}_0^\phi(\boldsymbol{x})$ and the encoder $f_\psi$ as a mixed-integer linear program (MILP).

MODELING $\mathbb{S}_0^\phi$ AS MILP We use an example to demonstrate the encoding of logical constraints with MILP. Consider an individual $\boldsymbol{x}$ that has two categorical features $x_1 = [1, 0, \dots, 0]$ and $x_2 = [0, \dots, 0, 1]$ and one numerical feature $x_3$, with the following constraint for similarity:

$$\phi(\boldsymbol{x}, \boldsymbol{x}') := (x_1 = x_1') \wedge (|x_3 - x_3'| \leq \alpha).$$

Here $\boldsymbol{x}$ is an individual from the test dataset and can be treated as constant, while $\boldsymbol{x}'$ is encoded using mixed-integer variables. For every categorical feature $x_i'$ we introduce $k$ binary variables $v_i^l$ with $l = 1, \dots, k$, where $k$ is the number of distinct values this categorical feature can take. For the fixed categorical feature $x_1'$, which is equal to $x_1$, we add the constraints $v_1^1 = 1$ and $v_1^l = 0$ for $l = 2, \dots, k$. To model the free categorical feature $x_2'$ we add the constraint $\sum_l v_2^l = 1$ thereby enforcing it to take on exactly one of the $k$ potential values. Finally, the numerical attribute $x_3'$ can be modeled by adding a corresponding variable $v_3$ with the two constraints: $v_3 \geq x_3 - \alpha$ and $v_3 \leq x_3 + \alpha$. It can be easily verified that our encoding of $S_\phi$ is exact.

Consider now a fairness constraint including disjunctions, i.e., $\phi := \phi_1 \vee \phi_2$. To model such a disjunction we introduce two auxiliary binary variables $v_1$ and $v_2$ with the constraints $v_i = 1 \iff \phi_i(\boldsymbol{x}, \boldsymbol{x}') = 1$ for $i = 1, 2$ and $v_1 + v_2 \geq 1$. Note that the encodings demonstrated on these two examples can be applied for general constraints $\phi$.

MODELING $f_\psi$ AS MILP To model the encoder we employ the method from Tjeng, Xiao, and Tedrake [45] which is exact for neural networks with ReLU activations. We recall that a ReLU performs $\max\{t, 0\}$ for some input $t$. Given an upper and lower bound on $t$, i.e., $t \in [l, u]$ we can encode the output of ReLU exactly via case distinction: (i) if $u \leq 0$ add a variable with upper and lower bound 0 to MILP, (ii) if $l \geq 0$ add a variable with upper and lower bounds $u$ and $l$ respectively to MILP, and (iii) if $l < 0 < u$, add a variable $v$ and a binary indicator $i$ to MILP in addition to the following constraints:

$$\begin{gathered} 0 \leq v \leq t \cdot i, \\ t \leq v \leq t - l \cdot (1 - i), \\ i = 1 \iff 0 \leq t. \end{gathered}$$

Finally, given the MILP formulation of $\mathbb{S}_0^{\phi}$ and $f_\psi$ we can compute $\epsilon$ by solving the following $k$ MILP instances (where $k$ is the dimension of the latent space):

$$\hat{\epsilon}_j = \max_{\boldsymbol{x}' \in \mathbb{S}_0^{\phi}(\boldsymbol{x})} |\left[f_\psi\left(\boldsymbol{x}\right)\right]_j - \left[f_\psi\left(\boldsymbol{x}'\right)\right]_j|.$$

We compute the final result as $\epsilon = \max\{\hat{\epsilon}_1, \hat{\epsilon}_2, \ldots \hat{\epsilon}_k\}$.

### 5.4.2 *Certifying Local Robustness*

The data consumer obtains a point in latent space $\boldsymbol{z}$ and a radius $\epsilon$. To obtain a fairness certificate, the data consumer certifies that all points in the latent space at $\ell_\infty$-distance at most $\epsilon$ from $\boldsymbol{z}$ are mapped to the same label as $\boldsymbol{z}$. This amounts to solving the following MILP optimization problem for each logit $[h_\theta(\boldsymbol{z}')]_{y'}$ with label $y'$ different from the true label $y$:

$$\max_{\boldsymbol{z}' \in \mathbb{B}_\infty(\boldsymbol{z}, \epsilon)} [h_\theta(\boldsymbol{z}')]_{y'} - [h_\theta(\boldsymbol{z}')]_y.$$

If the solution of the above optimization problem is less than zero for each $y' \neq y$, then robustness of the classifier $h_\theta$ is provably established. Note that, the data consumer can employ same methods as the data producer to encode the classifier as MILP [45] and benefit from any corresponding advancements in solving MILP instances in the context of neural network certification, e.g., [150].

We now formalize our certificate, that allows the data consumer to prove individual fairness of $M$, once given $\boldsymbol{z}$ and $\epsilon$ by the data producer:

**Theorem 2.** *(Individual fairness certificate) Suppose $M = h_\theta \circ f_\psi$ with data point $\boldsymbol{x}$ and similarity notion $\phi$. Furthermore, let $\boldsymbol{z} = f_\psi(\boldsymbol{x})$ be encoded rep-*

*resentation,* $\mathbb{S}_0^\phi(\boldsymbol{x}) = \{\boldsymbol{x}' \in \mathbb{R}^n \mid \phi(\boldsymbol{x}, \boldsymbol{x}')\}$ *the set of similar points, and* $\epsilon = \max_{\boldsymbol{x}' \in \mathbb{S}_0^\phi(\boldsymbol{x})} ||\boldsymbol{z} - f_\psi(\boldsymbol{x}')||_\infty$ *a given radius. If*

$$\max_{\boldsymbol{z}' \in \mathbb{B}_\infty(\boldsymbol{z}, \epsilon)} [h_\theta(\boldsymbol{z}')]_{y'} - [h_\theta(\boldsymbol{z}')]_y < 0$$

*for all labels* $y'$ *different from the true label* $y$*, then for all* $\boldsymbol{x}' \in \mathbb{S}_0^\phi(\boldsymbol{x})$ *we have* $M(\boldsymbol{x}) = M(\boldsymbol{x}')$*.*

## 5.5 EXPERIMENTAL EVALUATION

We implement our method in a tool called LCIFR and present an extensive experimental evaluation. We consider a variety of different datasets — Adult [151], Compas [152], Crime [151], German [151], Health (`https://www.kaggle.com/c/hhp`), and Law School [153]. We perform the following preprocessing on all datasets: (i) normalize numerical attributes to zero mean and unit variance, (ii) one-hot encode categorical features, (iii) drop rows and columns with missing values, and (iv) split into train, test and validation sets. Although we only consider datasets with binary classification tasks, we note that our method straightforwardly extends to the multiclass case. We perform all experiments on a desktop PC using a single GeForce RTX 2080 Ti GPU and 16-core Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz. We make all code, datasets and preprocessing pipelines publicly available at `https://github.com/eth-sri/lcifr` to ensure reproducibility of our results.

EXPERIMENT SETUP We model the encoder $f_\psi$ as a neural network, and we use logistic regression as a classifier $h_\theta$. We perform a grid search over model architectures and loss balancing factors $\gamma$ which we evaluate on the validation set. As a result, we consider $f_\psi$ with 1 hidden layer of 20 neurons (except for Law School where we do not have a hidden layer) and a latent space of dimension 20. We fix $\gamma$ to 10 for Adult, Crime, and German, to 1 for Compas and Health, and to 0.1 for Law School.

FAIRNESS CONSTRAINTS We propose a range of different constraints for which we apply our method. These constraints define the similarity between two individuals based on their numerical attributes (NOISE), categorical attributes (CAT), or combinations thereof (CAT + NOISE). Furthermore, we consider more involved similarity notions based on disjunctions (ATTRIBUTE)

| CONSTRAINT | DATASET | ACCURACY (%) BASE | ACCURACY (%) LCIFR | CERTIFIED (%) BASE | CERTIFIED (%) LCIFR |
|---|---|---|---|---|---|
| NOISE | ADULT | 83.0 | 81.4 | 59.0 | 97.8 |
| | COMPAS | 65.8 | 63.4 | 32.1 | 79.0 |
| | CRIME | 84.4 | 83.1 | 7.4 | 66.9 |
| | GERMAN | 76.5 | 74.0 | 71.0 | 97.5 |
| | HEALTH | 80.8 | 81.1 | 75.4 | 97.8 |
| | LAW SCHOOL | 84.4 | 84.6 | 57.9 | 69.2 |
| CAT | ADULT | 83.3 | 83.1 | 79.9 | 100 |
| | COMPAS | 65.6 | 66.3 | 90.9 | 100 |
| | CRIME | 84.4 | 83.9 | 78.3 | 100 |
| | GERMAN | 76.0 | 75.5 | 88.5 | 100 |
| | HEALTH | 80.7 | 80.9 | 64.1 | 99.8 |
| | LAW SCHOOL | 84.4 | 84.4 | 25.6 | 51.1 |
| CAT + NOISE | ADULT | 83.3 | 81.3 | 47.5 | 97.6 |
| | COMPAS | 65.6 | 63.7 | 30.9 | 75.6 |
| | CRIME | 84.4 | 81.5 | 6.2 | 63.3 |
| | GERMAN | 76.0 | 70.0 | 68.0 | 95.5 |
| | HEALTH | 80.7 | 80.7 | 24.7 | 97.3 |
| | LAW SCHOOL | 84.4 | 84.5 | 11.6 | 28.9 |
| ATTRIBUTE | ADULT | 83.0 | 80.9 | 49.3 | 94.6 |
| | GERMAN | 76.5 | 73.5 | 65.0 | 96.5 |
| | LAW SCHOOL | 84.3 | 86.9 | 46.4 | 62.6 |
| QUANTILES | LAW SCHOOL | 84.2 | 84.2 | 56.5 | 76.9 |

TABLE 5.1: Accuracy and certified individual fairness. We compare the accuracy and percentage of certified individuals with a baseline obtained from setting the loss balancing factor $\gamma = 0$. LCIFR produces a drastic increase in certified individuals while only incurring minor decrease in accuracy.

and quantiles of certain attributes to counter subordination between social groups [127] (QUANTILES).

APPLYING OUR METHOD IN PRACTICE We assume that the data regulator has defined the above constraints. First, we act as the data producer and learn a representation that enforces the individual fairness constraints using our method from §5.3. After training, we compute $\epsilon$ for every individual data point in the test set and pass it to the data consumer along with the latent representation of the entire dataset as described in §5.4.1. Second, we act as data consumer and use our method from §5.3 to learn a locally-robust classifier from the latent representation. Finally, to obtain

| Task | Label | Accuracy (%) | Certified (%) |
|---|---|---|---|
| Original | charlson index | 73.8 | 96.9 |
| Transfer | msc2a3 | 73.7 | 86.1 |
| | metab3 | 75.4 | 93.6 |
| | arthspin | 75.4 | 93.7 |
| | neument | 73.8 | 97.1 |
| | respr4 | 72.4 | 98.4 |

Table 5.2: Accuracy and percentage of certified individuals for transferable representation learning on Health dataset with Cat + Noise constraint. The transfer labels are omitted during training and the data producer objective is augmented with a reconstruction loss. This allows the data consumer to achieve high accuracies and certification rates across a variety of (potentially unknown) tasks.

a certificate of individual fairness, we use $\epsilon$ to certify the classifier via our method from §5.4.2.

In Table 5.1 we compare the accuracy and percentage of certified individuals (i.e., the empirical approximation of a lower bound on Eq. (5.1)) with a baseline encoder and classifier obtained from standard representation learning (i.e., $\gamma = 0$). We do not compare with other approaches for learning individually fair representations since they either consider a different similarity metric or employ nonlinear components that cannot be efficiently certified. It can be observed that LCIFR drastically increases the percentage of certified individuals across all constraints and datasets. We would like to highlight the relatively low (albeit still significantly higher than baseline) certification rate for the Law School dataset. This is due to the relatively small loss balancing factor $\gamma = 0.1$ which only weakly enforces the individual fairness constraint during training. Finally, we report the following mean certification runtime per input, averaged over all constraints: 0.29s on Adult, 0.35s on Compas, 1.23s on Crime, 0.28s on German, 0.68s on Health, and 0.02s on Law School, showing that our method is computationally efficient.

FAIR TRANSFER LEARNING We follow Madras et al. [68] to demonstrate that our method is compatible with transferable representation learning. We also consider the Health dataset, for which the original task is to predict the Charlson Index. To demonstrate transferability, we omit the primary condition group labels from the set of features, and try to predict them from the latent representation without explicitly optimizing for the task. To that end, the data producer additionally learns a decoder $g\left(\boldsymbol{z}\right)$, which tries to predict the original attributes $\boldsymbol{x}$ from the latent representation, thereby not only retaining task-specific information on the Charlson Index. This amounts to adding a reconstruction loss $\mathcal{L}_R\left(\boldsymbol{x}, g\left(f_\psi\left(\boldsymbol{x}\right)\right)\right)$ (e.g., $\ell_2$) to the objective in Eq. (5.5). Assuming that our representations are in fact transferable, the data consumer is now free to choose any classification objective. We note that our certification method straightforwardly extends to all possible prediction tasks allowing the data consumer to obtain fairness certificates regardless of the objective. Here, we let the data consumer train classifiers for both the original task and to predict the 5 most common primary condition group labels. We display the accuracy and percentage of certified data points on all tasks in Table 5.2. The table shows that our learned representation transfers well across tasks while additionally providing provable individual fairness guarantees.

## 5.6 DISCUSSION

We introduced a novel end-to-end framework for learning representations with provable certificates of individual fairness. We demonstrated that our method is compatible with existing notions of fairness, such as transfer learning. Our evaluation across different datasets and fairness constraints demonstrates the practical effectiveness of our method.

IMPACT There has been extensive follow-up work on learning representations with individual fairness guarantees [10, 24] as well as other types of guarantees (e.g. group fairness). Recent work [154] certified different notions of individual fairness and in Peychev et al. [10] we extended the method for learning certified individually fair representations described in this chapter to vision models where individual fairness is defined in the latent space of a generative model. Another work [155] considered extending the specification of individual similarity from hard-coded logical formulas to those obtained via combination of language models and crowd-sourcing.

# 6

# LEARNING CERTIFIED GROUP FAIR REPRESENTATIONS

As discussed in Chapter 5, fair representation learning has become one of the most promising ways to encode data into new representations with high fairness and utility. While in the previous chapter the fairness was defined as individual fairness, in this chapter we focus on group fairness which is more widely used in practice due to its computational simplicity. More concretely, the goal is to ensure that representations have two properties: (i) they are informative for various prediction tasks of interest, (ii) sensitive attributes of the original data (e.g., race) cannot be recovered from the representations (which implies group fairness). Perhaps the most prominent approach for learning such fair representations is adversarial training [68, 71, 72, 73, 74], which jointly trains an encoder trying to transform data into a fair representation with an adversary attempting to recover sensitive attributes from the representation. However, several recent lines of work [69, 77, 156, 157, 158, 159] have noticed that these approaches do not produce truly fair representations: stronger adversaries *can* in fact recover sensitive attributes. Clearly, this could allow malicious or ignorant users to use the provided representations to discriminate.

KEY CHALLENGE This raises the following question: *Can we learn representations which provably guarantee that sensitive attributes cannot be recovered?* More concretely, can we learn an encoder $f$ that maps tuple of input and attribute $(\boldsymbol{x}, a)$ to a new representation $\boldsymbol{z}$ such that any classifier $h$ trained to predict the sensitive attribute $a$ from $\boldsymbol{z}$ provably (with high confidence) has accuracy below some threshold?

THIS CHAPTER Following prior work, we focus on tabular datasets used for tasks such as loan or insurance assessment where fairness is of high relevance. We assume that the original input data $\boldsymbol{x}$ comes from two probability distributions $p_0$ and $p_1$, representing groups with sensitive attributes $a = 0$ and $a = 1$, respectively. In the cases where distributions $p_0$ and $p_1$ are known, we will obtain provable fairness guarantees, and otherwise we perform density estimation and obtain guarantees with respect to the estimated distribution. In our experimental evaluation we confirm that the

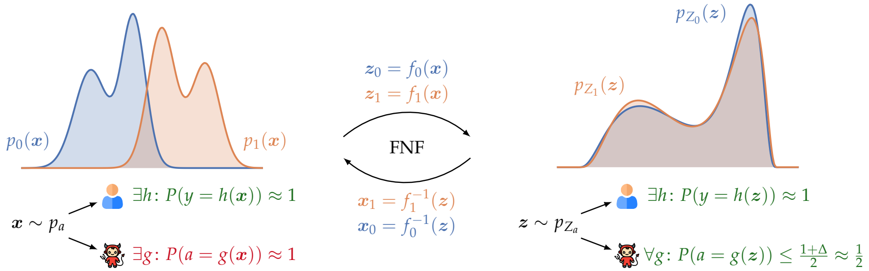

$p_0(\boldsymbol{x})$
$p_1(\boldsymbol{x})$
$\boldsymbol{z}_0 = f_0(\boldsymbol{x})$
$\boldsymbol{z}_1 = f_1(\boldsymbol{x})$
FNF
$\boldsymbol{x}_1 = f_1^{-1}(\boldsymbol{z})$
$\boldsymbol{x}_0 = f_0^{-1}(\boldsymbol{z})$
$p_{Z_0}(\boldsymbol{z})$
$p_{Z_1}(\boldsymbol{z})$
$\boldsymbol{x} \sim p_a$
$\exists h\colon P(y = h(\boldsymbol{x})) \approx 1$
$\exists g\colon P(a = g(\boldsymbol{x})) \approx 1$
$\boldsymbol{z} \sim p_{Z_a}$
$\exists h\colon P(y = h(\boldsymbol{z})) \approx 1$
$\forall g\colon P(a = g(\boldsymbol{z})) \leq \frac{1+\Delta}{2} \approx \frac{1}{2}$

FIGURE 6.1: Overview of Fair Normalizing Flows (FNF). There are two encoders, $f_0$ and $f_1$, that transform the two input distributions $p_0$ and $p_1$ into latent distributions $p_{Z_0}$ and $p_{Z_1}$ with a small statistical distance $\Delta \approx 0$. Without FNF, a strong adversary $g$ can easily recover sensitive attribute $a$ from the original input $\boldsymbol{x}$, but once inputs are passed through FNF, we are guaranteed that *any* adversary that tries to guess sensitive attributes from latent $\boldsymbol{z}$ cannot be significantly better than random chance. At the same time, we can ensure that any benevolent user $h$ maintains high utility.

bounds computed on the estimated distribution in practice also bound adversarial accuracy on the true distribution, meaning that density estimation works well for the setting we consider.

To address the above challenges, we propose Fair Normalizing Flows (FNF), a new method for learning fair representations with guarantees. In contrast to other approaches where encoders are standard feed-forward neural networks, we instead model the encoder as a normalizing flow [25]. Fig. 6.1 provides a high-level overview of FNF. As shown on the left in Fig. 6.1, using raw inputs $\boldsymbol{x}$ allows us to train high-utility classifiers $h$, but at the same time does not protect against the existence of a malicious adversary $g$ that can predict a sensitive attribute $a$ from the features in $\boldsymbol{x}$. Our architecture consists of two flow-based encoders $f_0$ and $f_1$, where flow $f_a$ transforms probability distribution $p_a(\boldsymbol{x})$ into $p_{Z_a}(\boldsymbol{z})$ by mapping $\boldsymbol{x}$ into $\boldsymbol{z} = f_a(\boldsymbol{x})$. The goal of the training procedure is to minimize the distance $\Delta$ between the resulting distributions $p_{Z_0}(\boldsymbol{z})$ and $p_{Z_1}(\boldsymbol{z})$ so that an adversary cannot distinguish between them. Intuitively, after training our encoder, each latent representation $\boldsymbol{z}$ can be inverted into original inputs $x_0 = f_0^{-1}(\boldsymbol{z})$ and $x_1 = f_1^{-1}(\boldsymbol{z})$ that should ideally have similar probability w.r.t. $p_0$ and $p_1$, meaning that even the optimal adversary cannot distinguish which of them actually produced latent $\boldsymbol{z}$. Crucially, as normalizing flows enable us to compute the exact likelihood in the latent space, for trained encoders we can upper bound the accuracy of *any* adversary with $\frac{1+\Delta}{2}$, which should be small if training was successful. Furthermore, the distance $\Delta$ provides a tight upper bound [68] on common fairness notions such as demographic parity [143] and equalized odds [160]. As shown on the right in Fig. 6.1, we can still train high-utility classifiers $h$ using our representations, but now we can actually *guarantee* that no adversary $g$ can recover sensitive attributes better than chance.

We empirically demonstrate that FNF can substantially increase provable fairness without significantly sacrificing accuracy on several common datasets. Additionally, we show that the invertibility of FNF enables algorithmic recourse, allowing us to examine how to reverse a negative decision outcome.

MAIN CONTRIBUTIONS Our key contributions are:

- A novel fair representation learning method, called Fair Normalizing Flows (FNF), which guarantees that the sensitive attributes cannot be recovered from the learned representations at the cost of a small decrease in classification accuracy.
- Experimental evaluation demonstrating that FNF can provably remove sensitive attributes from the representations, while keeping accuracy for the prediction task sufficiently high.
- Extensive investigation of algorithmic recourse and applications of FNF to transfer learning.

## 6.1 RELATED WORK

In this chapter, we focus on group fairness, which requires certain classification statistics to be equal across different groups of the population. Concretely, we consider demographic parity [143], equalized odds [160], and equality of opportunity [160], which are widely studied in the literature [67, 68, 71]. Algorithms enforcing such fairness notions target various stages of the machine learning pipeline: Pre-processing methods transform sensitive data into an unbiased representation [67, 146], in-processing methods modify training by incorporating fairness constraints [161, 162], and post-processing methods change the predictions of a pre-trained classifier [160]. Here, we consider fair representation learning [67], which computes data representations that hide sensitive information, e.g. group membership, while maintaining utility for downstream tasks and allowing transfer learning.

RECOVERING SENSITIVE ATTRIBUTES FROM REPRESENTATIONS Fair representations can be learned with a variety of different approaches, including variational autoencoders [69, 70], adversarial training [68, 71, 72, 73, 74, 75, 76, 77], and disentanglement [78, 79]. Adversarial training methods minimize a lower bound on demographic parity, namely an adversary's accuracy for predicting the sensitive attributes from the latent representation. However, since these methods only empirically evaluate worst-case unfairness, adversaries that are not considered during training can still recover sensitive attributes from the learned representations [69, 77, 156, 157, 158, 159]. These findings illustrate the necessity of learning representations with provable guarantees on the maximum recovery of sensitive information regardless of the adversary, which is precisely the goal of this chapter. Prior work makes first steps in this direction: Gupta et al. [158] upper bound a monotonically increasing function of demographic parity with the mutual information between the latent representation and sensitive attributes. However, the monotonic nature of this bound prevents computing guarantees on the reconstruction power of the optimal adversary. Feng et al. [77] minimize the Wasserstein distance between latent distributions of different protected groups, but only provide an upper bound on the performance of any Lipschitz continuous adversary. However, as we will show, the optimal adversary is generally discontinuous. Cerrato et al. [163] also learn fair representations using normalizing flows, but different to us, they do not use exact likelihood computation to provide theoretical fairness guarantees.

PROVABLE GROUP FAIRNESS GUARANTEES The ongoing development of guidelines on the fair usage of AI [164, 165, 166] has spurred interest in provably fair algorithms. Unlike this chapter, the majority of these efforts [146, 167, 168], including our approach in Chapter 5, focus on individual fairness. Individual fairness is also tightly linked to differential privacy [143, 169], which guarantees that an attacker cannot infer whether a given individual was present in the dataset or not, but these models can still admit reconstruction of sensitive attributes by leveraging population-level correlations [144]. Group fairness certification methods [170, 171, 172] generally only focus on certification and, unlike this chapter, do not learn representations that are provably fair.

## 6.2 BACKGROUND

We assume that the data $(\boldsymbol{x}, a) \in \mathbb{R}^d \times \mathcal{A}$ comes from a probability distribution $p$, where $\boldsymbol{x}$ represents the features and $a$ represents a sensitive attribute. In this chapter, we focus on the case where the sensitive attribute is binary, meaning $\mathcal{A} = \{0, 1\}$. Given $p$, we can define the conditional probabilities as $p_0(\boldsymbol{x}) = P(\boldsymbol{x} \mid a = 0)$ and $p_1(\boldsymbol{x}) = P(\boldsymbol{x} \mid a = 1)$. We are interested in classifying each sample $(\boldsymbol{x}, a)$ to a label $y \in \{0, 1\}$, which may or may not be correlated with the sensitive attribute $a$. Our goal is to build a classifier $\hat{y} = h(\boldsymbol{x})$ that tries to predict $y$ from the features $\boldsymbol{x}$, while satisfying certain notions of fairness. Next, we present several definitions of fairness relevant for this chapter.

FAIRNESS CRITERIA A classifier $h$ satisfies *demographic parity* if it assigns positive outcomes to both sensitive groups equally likely, i.e., $P(h(\boldsymbol{x}) = 1 \mid a = 0) = P(h(\boldsymbol{x}) = 1 \mid a = 1)$. If demographic parity cannot be satisfied, we consider demographic parity distance, defined as $|\mathbb{E}[h(\boldsymbol{x}) \mid a = 0] - \mathbb{E}[h(\boldsymbol{x}) \mid a = 1]|$. An issue with demographic parity occurs if the base rates differ among the attributes, i.e., $P(y = 1 \mid a = 0) \neq P(y = 1 \mid a = 1)$. In that case, even the ground truth label $y$ does not satisfy demographic parity. Thus, Hardt, Price, and Srebro [160] introduced *equalized odds*, which requires that $P(h(\boldsymbol{x}) = 1 \mid y = y_0, a = 0) = P(h(\boldsymbol{x}) = 1 \mid y = y_0, a = 1)$ for $y_0 \in \{0, 1\}$.

FAIR REPRESENTATIONS Instead of directly predicting $y$ from $\boldsymbol{x}$, Zemel et al. [67] introduced the idea of learning *fair representations* of data. The idea is that a data producer preprocesses the original data $\boldsymbol{x}$ to obtain a new representation $\boldsymbol{z} = f(\boldsymbol{x}, a)$. Then, any data consumer, who is using this data to solve a downstream task, can use $\boldsymbol{z}$ as an input to the classifier instead of the original data $\boldsymbol{x}$. Thus, if the data producer can ensure that data representation is fair (w.r.t. some fairness notion), then all classifiers employing this representation will automatically inherit the fairness property. However, due to inherent biases of the dataset, this fairness increase generally results in a small accuracy decrease.

NORMALIZING FLOWS Flow-based generative models [25, 173, 174, 175] provide an attractive framework for transforming any probability distribution $q$ into another distribution $\bar{q}$. Accordingly, they are often used to estimate densities from data using the *change of variables* formula on a sequence of invertible transformations, so-called normalizing flows [25]. In this chapter, however, we mainly leverage the fact that flow models sample a latent variable $\boldsymbol{z}$ from a density $\bar{q}(\boldsymbol{z})$ and apply an invertible function $f_{\boldsymbol{\theta}}$, parametrized by $\boldsymbol{\theta}$, to obtain datapoint $\boldsymbol{x} = f_{\boldsymbol{\theta}}^{-1}(\boldsymbol{z})$. Given a density $q(\boldsymbol{x})$, the exact log-likelihood is then obtained by applying the change of variables formula $\log q(\boldsymbol{x}) = \log \bar{q}(\boldsymbol{z}) + \log|\det(d\boldsymbol{z}/d\boldsymbol{x})|$. Thus, for $f_{\boldsymbol{\theta}} = f_1 \circ f_2 \circ \ldots \circ f_K$ with $\boldsymbol{r}_0 = \boldsymbol{x}$, $f_i(\boldsymbol{r}_{i-1}) = \boldsymbol{r}_i$, and $\boldsymbol{r}_K = \boldsymbol{z}$, we have

$$\log q(\boldsymbol{x}) = \log \bar{q}(\boldsymbol{z}) + \sum_{i=1}^{K} \log|\det(d\boldsymbol{r}_i/d\boldsymbol{r}_{i-1})|. \tag{6.1}$$

An appropriate choice of transformations $f_i$ [25, 173, 174] makes the computation of the log-determinant tractable, resulting in efficient training and sampling. Alternative generative models cannot compute the exact log-likelihood (e.g., VAEs [176], GANs [177]) or have inefficient sampling (e.g., autoregressive models). Our approach is also related to discrete flows [178, 179] and alignment flows [180, 181]. However, alignment flows jointly learn the density and the transformation, unlike the fairness setting where these are computed by different entities.

## 6.3 MOTIVATION

In this section, we motivate our approach by highlighting some key issues with fair representation learning based on adversarial training. Consider a distribution of samples $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ divided into two groups, shown as blue and orange in Fig. 6.2. The first group with a sensitive attribute $a = 0$ has a distribution $(x_1, x_2) \sim p_0$, where $p_0$ is a mixture of two Gaussians $\mathcal{N}([-3, 3], I)$ and $\mathcal{N}([3, 3], I)$. The second group with a sensitive attribute $a = 1$ has a distribution $(x_1, x_2) \sim p_1$, where $p_1$ is a mixture of two Gaussians $\mathcal{N}([-3, -3], I)$ and $\mathcal{N}([3, -3], I)$. The label of a point $(x_1, x_2)$ is defined by $y = 1$ if $\text{sign}(x_1) = \text{sign}(x_2)$ and $y = 0$ otherwise. Our goal is to learn a data representation $\boldsymbol{z} = f(\boldsymbol{x}, a)$ such that it is *impossible* to recover $a$ from $\boldsymbol{z}$, but still possible to predict target $y$ from $\boldsymbol{z}$. Note that such a representation exists for our task: simply setting $\boldsymbol{z} = f(\boldsymbol{x}, a) = (-1)^a \boldsymbol{x}$ makes it impossible to predict whether a particular $\boldsymbol{z}$ corresponds to $a = 0$ or $a = 1$, while still allowing us to train a classifier $h$ with essentially perfect accuracy (e.g., $h(\boldsymbol{z}) = 1_{\{z_1 > 0\}}$).

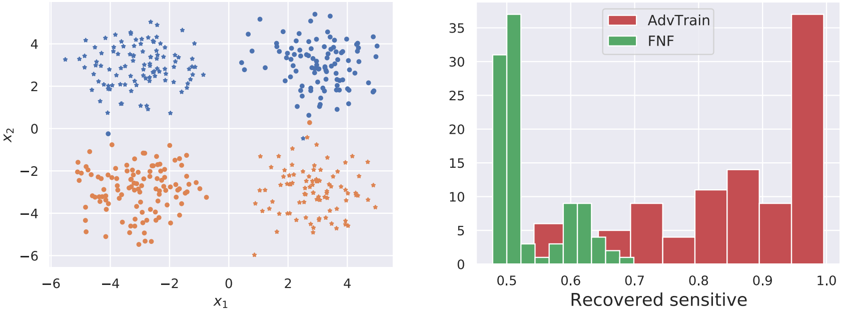


Figure 6.3: Sensitive attribute recovery rates for adversarial training and fair normalizing flows (FNF) with 100 different random seeds.

**ADVERSARIAL TRAINING FOR FAIR REPRESENTATIONS** Adversarial training [68, 71] is an approach that trains encoder $f$ and classifier $h$ jointly with an adversary $g$ trying to predict the sensitive attribute $a$. While the adversary tries to minimize its loss $\mathcal{L}_{adv}$, the encoder $f$ and classifier $h$ are trying to maximize $\mathcal{L}_{adv}$ and minimize the classification loss $\mathcal{L}_{clf}$ as

$$\min_{f,h} \max_{g \in \mathcal{G}} \mathbb{E}_{(\boldsymbol{x},a)\sim D} \left[ \mathcal{L}_{clf}(f(\boldsymbol{x},a),h) - \gamma \mathcal{L}_{adv}(f(\boldsymbol{x},a),g) \right], \tag{6.2}$$

where $\mathcal{G}$ denotes the model family of adversaries, e.g., neural networks, considered during training. Unfortunately, there are two key issues with adversarial training. First, it yields a non-convex optimization problem, which usually cannot be solved to optimality because of saddle points. Second, it assumes that the adversary $g$ comes from a fixed model family $\mathcal{G}$, which means that even if the optimal $g \in \mathcal{G}$ cannot recover the sensitive attribute $a$, adversaries from other model families can still do so as demonstrated in recent work [69, 77, 156, 157, 158]. To investigate these issues, we apply adversarial training to learn representations for our synthetic example, and measure how often the sensitive attributes can be recovered from learned representations. Our results, shown in Fig. 6.3, repeated 100 times with different seeds, demonstrate that adversarial training is unstable and rarely results in truly fair representations (where only 50% can be recovered). In §6.5 we follow up on recent work and show that several adversarial fair representation learning approaches do not work against adversaries from a different model familiy (e.g., larger networks). In Fig. 6.3 we show that our approach, introduced next, can reliably produce fair representations without affecting the utility.

## 6.4 FAIR NORMALIZING FLOWS

Throughout this section we will assume knowledge of prior distributions $p_0(\boldsymbol{x})$ and $p_1(\boldsymbol{x})$. At the end of the section, we discuss the required changes if we only work with estimates. Let $\mathcal{Z}_0$ and $\mathcal{Z}_1$ denote conditional distributions of $z = f(\boldsymbol{x}, a)$ for $a \in \{0, 1\}$, and let $p_{Z_0}$ and $p_{Z_1}$ denote their respective densities. Madras et al. [68] have shown that bounding the *statistical distance* $\Delta(p_{Z_0}, p_{Z_1})$ between $\mathcal{Z}_0$ and $\mathcal{Z}_1$ provides an upper bound on the unfairness of any classifier $h$ built on top of the representation encoded by $f$. The statistical distance between $\mathcal{Z}_0$ and $\mathcal{Z}_1$ is defined similarly to maximum mean discrepancy (MMD) [182] between the two distributions:

$$\Delta(p_{Z_0}, p_{Z_1}) \triangleq \sup_{\mu \in \mathcal{B}} |\mathbb{E}_{\boldsymbol{z} \sim \mathcal{Z}_0}[\mu(\boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z} \sim \mathcal{Z}_1}[\mu(\boldsymbol{z})]|, \tag{6.3}$$

where $\mu \colon \mathbb{R}^d \to \{0, 1\}$ is a function in a set of all binary classifiers $\mathcal{B}$ trying to discriminate between $\mathcal{Z}_0$ and $\mathcal{Z}_1$. If we can train an encoder to induce latent distributions $\mathcal{Z}_0$ and $\mathcal{Z}_1$ with statistical distance below some threshold, then we can both upper bound the maximum adversarial accuracy by $(1 + \Delta(p_{Z_0}, p_{Z_1}))/2$ and, using the bounds from Madras et al. [68], obtain guarantees for demographic parity and equalized odds of any downstream classifier $h$. Such guarantees are unattainble for adversarial training, which minimizes a lower bound of $\Delta(p_{Z_0}, p_{Z_1})$. In contrast, we learn fair representations that allow computing the optimal adversary $\mu^*$ attaining the supremum in Eq. (6.3) and thus enable exact evaluation of $\Delta(p_{Z_0}, p_{Z_1})$.

OPTIMAL ADVERSARY In the following lemma we state the form of an optimal adversary which attains the supremum in the definition of statistical distance in Eq. (6.3). The full proof can be found in Balunović, Ruoss, and Vechev [4].

**Lemma 1.** *The adversary $\mu^*$ attaining the supremum in the definition of $\Delta(p_{Z_0}, p_{Z_1})$ can be defined as $\mu^*(\boldsymbol{z}) = 1_{\{p_{Z_0}(\boldsymbol{z}) \leq p_{Z_1}(\boldsymbol{z})\}}$, namely it evaluates to $1$ if and only if $p_{Z_0}(\boldsymbol{z}) \leq p_{Z_1}(\boldsymbol{z})$.*

This intuitively makes sense – given some representation $\boldsymbol{z}$, the adversary computes likelihood under both distributions $\mathcal{Z}_0$ and $\mathcal{Z}_1$, and predicts the attribute with higher likelihood for that $\boldsymbol{z}$. Liao et al. [75] also observed that the optimal adversary can be phrased as $\arg\max_a p(a|\boldsymbol{z})$. So far, prior work mostly focused on mapping input $\boldsymbol{x}$ to the latent representation $\boldsymbol{z} = f_\theta(\boldsymbol{x}, a)$ via standard neural networks. However, for such models, given densities $p_0(\boldsymbol{x})$ and $p_1(\boldsymbol{x})$ over the input space, it is intractable to compute the densities $p_{Z_0}(\boldsymbol{z})$ and $p_{Z_1}(\boldsymbol{z})$ in the latent space as many inputs $\boldsymbol{x}$ can be mapped to the same latent $\boldsymbol{z}$ and we cannot use inverse function theorem. Consequently, adversarial training methods cannot compute the optimal adversary and thus resort to a lower bound.

ENCODING WITH NORMALIZING FLOWS Our approach, named Fair Normalizing Flows (FNF), consists of two models, $f_0$ and $f_1$, that encode inputs from the groups with sensitive attributes $a = 0$ and $a = 1$, respectively. We show a high-level overview of FNF in Fig. 6.1. Note that models $f_0$ and $f_1$ are parameterized by $\theta_0$ and $\theta_1$, but we do not write this explicitly to ease the notation. Given some input $\boldsymbol{x}_0 \sim p_0$, it is encoded to $\boldsymbol{z}_0 = f_0(\boldsymbol{x}_0)$, inducing a probability distribution $\mathcal{Z}_0$ with density $p_{Z_0}(\boldsymbol{z})$ over all possible latent representations $\boldsymbol{z}$. Similarly, inputs $\boldsymbol{x}_1 \sim p_1$ are encoded to $\boldsymbol{z}_1 = f_1(\boldsymbol{x}_1)$, inducing the probability distribution $\mathcal{Z}_1$ with density $p_{Z_1}(\boldsymbol{z})$. Clearly, if we can train $f_0$ and $f_1$ so that the resulting distributions $\mathcal{Z}_0$ and $\mathcal{Z}_1$ have small distance, then we can guarantee fairness of the representations using the bounds from Madras et al. [68]. As evaluating the statistical distance is intractable for most neural networks, we need a model family that allows us to compute this quantity.

We propose to use bijective encoders $f_0$ and $f_1$ based on normalizing flows [25] which allow us to compute the densities at $\boldsymbol{z}$ using the change of variables formula

$$\log p_{Z_a}(\boldsymbol{z}) = \log p_a(f_a^{-1}(\boldsymbol{z})) + \log \left| \det \frac{\partial f_a^{-1}(\boldsymbol{z})}{\partial \boldsymbol{z}} \right| \tag{6.4}$$

for $a \in \{0, 1\}$. Recall that Lemma 1 provides a form of the optimal adversary. To compute the statistical distance it remains to evaluate the expectations $\mathbb{E}_{\boldsymbol{z}\sim\mathcal{Z}_0}[\mu^*(\boldsymbol{z})]$ and $\mathbb{E}_{\boldsymbol{z}\sim\mathcal{Z}_1}[\mu^*(\boldsymbol{z})]$. Sampling from $\mathcal{Z}_0$ and $\mathcal{Z}_1$ is straightforward – we can sample $\boldsymbol{x}_0 \sim p_0$ and $\boldsymbol{x}_1 \sim p_1$, and then pass the samples $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ through the respective encoders $f_0$ and $f_1$ to obtain $\boldsymbol{z}_0 \sim \mathcal{Z}_0$ and $\boldsymbol{z}_1 \sim \mathcal{Z}_1$. Given that the outputs of $\mu^*$ are bounded between 0 and 1, we can then use Hoeffding inequality to compute the confidence intervals for our estimate using a finite number of samples.

**Algorithm 3** Learning Fair Normalizing Flows

```
Input: $N, B, \gamma, p_0, p_1$
Initialize $h, f_0, f_1$ with parameters $\theta_h, \theta_0, \theta_1$
for $i = 1$ to $N$ do
  for $j = 1$ to $B$ do
    Sample $\boldsymbol{x}_0^j \sim p_0, \boldsymbol{x}_1^j \sim p_1$
    $\boldsymbol{z}_0^j = f_0(\boldsymbol{x}_0^j)$
    $\boldsymbol{z}_1^j = f_1(\boldsymbol{x}_1^j)$
  end for
  $\mathcal{L}_0 = \frac{1}{B}\sum_{j=1}^{B}(\log p_{Z_0}(\boldsymbol{z}_0^j) - \log p_{Z_1}(\boldsymbol{z}_0^j))$
  $\mathcal{L}_1 = \frac{1}{B}\sum_{j=1}^{B}(\log p_{Z_1}(\boldsymbol{z}_1^j) - \log p_{Z_0}(\boldsymbol{z}_1^j))$
  $\mathcal{L} = \gamma(\mathcal{L}_0 + \mathcal{L}_1) + (1-\gamma)\mathcal{L}_{clf}$
  Update $\theta_a \leftarrow \theta_a - \alpha\nabla_{\theta_a}\mathcal{L}$, for $a \in \{0,1\}$
  Update $\theta_h \leftarrow \theta_h - \alpha\nabla_{\theta_h}\mathcal{L}$
end for
```

**Lemma 2.** *Given a finite number of samples* $\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, ..., \boldsymbol{x}_0^n \sim p_0$ *and* $\boldsymbol{x}_1^1, \boldsymbol{x}_1^2, ..., \boldsymbol{x}_1^n \sim p_1$, *denote as* $\boldsymbol{z}_0^i = f_0(\boldsymbol{x}_0^i)$ *and* $\boldsymbol{z}_1^i = f_1(\boldsymbol{x}_1^i)$ *and let* $\hat{\Delta}(p_{Z_0}, p_{Z_1}) := |\frac{1}{n}\sum_{i=1}^{n}\mu^*(\boldsymbol{z}_0^i) - \frac{1}{n}\sum_{i=1}^{n}\mu^*(\boldsymbol{z}_1^i)|$ *be an empirical estimate of the statistical distance* $\Delta(p_{Z_0}, p_{Z_1})$. *Then, for* $n \geq -2\log\left(\frac{1-\sqrt{1-\delta}}{2}\right)/\epsilon^2$ *we are guaranteed that* $\Delta(p_{Z_0}, p_{Z_1}) \leq \hat{\Delta}(p_{Z_0}, p_{Z_1}) + \epsilon$ *with probability at least* $1-\delta$.

TRAINING FLOW-BASED ENCODERS The next challenge is to design a training procedure for our newly proposed architecture. The main issue is that the statistical distance is not differentiable (as the classifier $\mu^*$ is binary), so we replace it with a differentiable proxy based on the symmetrized KL divergence, shown in Lemma 3 below (proof provided in Balunović, Ruoss, and Vechev [4]. We show a high-level description of our training procedure in Algorithm 3. In each step, we sample a batch of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ from the respective distributions and encode them to the representations $\boldsymbol{z}_0$ and $\boldsymbol{z}_1$. We then estimate the symmetrized KL divergence between distributions $\mathcal{Z}_0$ and $\mathcal{Z}_1$, denoted as $\mathcal{L}_0 + \mathcal{L}_1$, and combine it with a classification loss $\mathcal{L}_{clf}$ using tradeoff parameter $\gamma$, and perform a gradient descent step to minimize the joint loss. While we use a convex scalarization scheme to obtain the joint loss in Algorithm 3, our approach is independent of the concrete multi-objective optimization objective.

**Lemma 3.** *We can bound* $\Delta(p_{Z_0}, p_{Z_1})^2 \leq \frac{1}{4}(KL(p_{Z_0}, p_{Z_1}) + KL(p_{Z_1}, p_{Z_0}))$.

BIJECTIVE ENCODERS FOR CATEGORICAL DATA Many fairness datasets consist of categorical data, and often even continuous data is discretized before training. In this case, we will show that the optimal bijective representation can be easily computed. Consider the case of discrete samples $\boldsymbol{x}$ coming from a probability distribution $p(\boldsymbol{x})$ where each component $x_i$ takes a value from a finite set $\{1, 2, \ldots, d_i\}$. Similar to the continuous case, our goal is to find bijections $f_0$ and $f_1$ that minimize the statistical distance of the latent distributions. Intuitively, we want to pair together inputs that have similar probabilities in both $p_0$ and $p_1$. In Lemma 4 we show that the solution that minimizes the statistical distance is obtained by sorting the inputs according to their probabilities in $p_0$ and $p_1$, and then matching inputs at the corresponding indices in these two sorted arrays. As this can result in a bad classification accuracy when inputs with different target labels get matched together, we can obtain another representation by splitting inputs in two groups according to the predicted classification label and then matching inputs in each group using Lemma 4. We can trade off accuracy and fairness by randomly selecting one of the two mappings based on a parameter $\gamma$.

**Lemma 4.** *Let* $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$ *and bijections* $f_0, f_1 : \mathcal{X} \to \mathcal{X}$. *Denote* $i_1, i_2, ..., i_m$ *and* $j_1, ..., j_m$ *permutations of* $\{1, 2, ..., m\}$ *such that*
$p_0(\boldsymbol{x}_{i_1}) \leq p_0(\boldsymbol{x}_{i_2}) \leq ... \leq p_0(\boldsymbol{x}_{i_m})$ *and* $p_1(\boldsymbol{x}_{j_1}) \leq p_1(\boldsymbol{x}_{j_2}) \leq ... \leq p_1(\boldsymbol{x}_{j_m})$. *The encoders defined by mapping* $f_0(\boldsymbol{x}_k) = \boldsymbol{x}_k$ *and* $f_1(\boldsymbol{x}_{j_k}) = \boldsymbol{x}_{i_k}$ *are bijective representations with the smallest possible statistical distance.*

STATISTICAL DISTANCE OF TRUE ESTIMATED DENSITY In this chapter we assume access to a density of the inputs for both groups and we *provably guarantee* fairness with respect to *this* density. While it is sensible in the cases where the density estimate can be trusted (e.g., if it was provided by a regulatory agency), in many practical scenarios, and our experiments in §6.5, we only have an estimate $\hat{p}_0$ and $\hat{p}_1$ of the true densities $p_0$ and $p_1$. We now want to know how far off our guarantees are compared to the ones for the true density. The following theorem provides a way to theoretically bound the statistical distance between $p_{Z_0}$ and $p_{Z_1}$ using the statistical distance between $\hat{p}_{Z_0}$ and $\hat{p}_{Z_1}$.

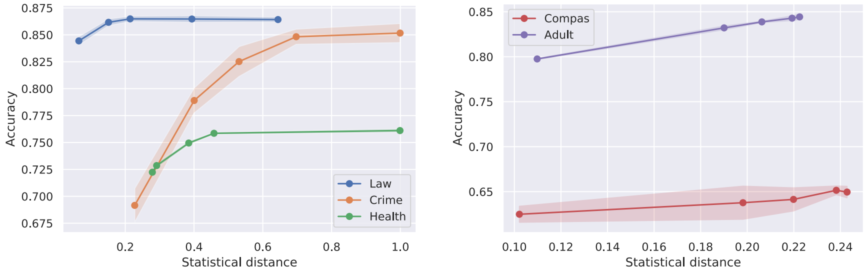
**Theorem 3.** *Let* $\hat{p}_0$ *and* $\hat{p}_1$ *be density estimates such that* $TV(\hat{p}_0, p_0) < \epsilon/2$ *and* $TV(\hat{p}_1, p_1) < \epsilon/2$, *where TV stands for the total variation between two* *distributions. If we denote the latent distributions* $f_0(\hat{p}_0)$ *and* $f_1(\hat{p}_1)$ *as* $\hat{p}_{Z_0}$ *and* $\hat{p}_{Z_1}$ *then* $\Delta(p_{Z_0}, p_{Z_1}) \leq \Delta(\hat{p}_{Z_0}, \hat{p}_{Z_1}) + \epsilon$*.*

This theorem can be combined with Lemma 2 to obtain a high probability upper bound on the statistical distance of the underlying true densities using estimated densities and a finite number of samples. Computing exact constants for the theorem is often not tractable, but as we will show experimentally, in practice the bounds computed on the estimated distribution in fact bound adversarial accuracy on the true distribution. Moreover, for low-dimensional data relevant to fairness, obtaining good estimates can be provably done for models such as Gaussian Mixture Models [183] and Kernel Density Estimation [184]. We can thus leverage the rich literature on density estimation [25, 174, 185, 186, 187] to estimate $\hat{p}_0$ and $\hat{p}_1$. Importantly, FNF is agnostic to the density estimation method, and can benefit from future advances in the field. Finally, we note that density estimation has already been applied in a variety of security-critical areas such as fairness [73], adversarial robustness [188], and anomaly detection [189].

## 6.5 EXPERIMENTAL EVALUATION

In this section, we evaluate Fair Normalizing Flows (FNF) on several standard datasets from the fairness literature. We consider UCI Adult and Crime [151], Compas [152], Law School [153], and the Health Heritage dataset. We preprocess Compas and Adult into categorical datasets by discretizing continuous features, and we keep the other datasets as continuous. Moreover, we preprocess the datasets by dropping uninformative features, facilitating the learning of a good density estimate, while keeping accuracy high. We make all of our code publicly available at `https://github.com/eth-sri/fnf`.

**EVALUATING FAIR NORMALIZING FLOWS** We first evaluate FNF's effectiveness in learning fair representations by training different FNF models with different values for the utility fairness tradeoff parameter $\gamma$. We estimate input densities using RealNVP [174] for Health, MADE [190] for Adult and Compas, and Gaussian Mixture Models (GMMs) for the rest (we also experiment with other density estimation methods [4]). For continuous datasets we use RealNVP as encoder, while for categorical datasets we compute the optimal bijective representations using Lemma 4. Fig. 6.4 shows our results, each point representing a single model, with models on the right focusing on classification accuracy, and models on the left



(a) Continuous datasets (b) Categorical datasets

FIGURE 6.4: Fair Normalizing Flows (FNF) on continuous and categorical data. The points show different accuracy vs. statistical distance tradeoffs (with 95% confidence intervals from varied random seeds), demonstrating that FNF significantly reduces statistical distance while retaining high accuracy.

gradually increasing their fairness focus. The results in Fig. 6.4, averaged over 5 random seeds, indicate that FNF successfully reduces the statistical distance between representations of sensitive groups while maintaining high accuracy. We observe that for some datasets (e.g., Law School) enforcing fairness only slightly degrades accuracy, while for others there is a substantial drop (e.g., Crime). In such datasets where the label and sensitive attribute are highly correlated we cannot achieve fairness and high accuracy simultaneously [191, 192]. Overall, we see that FNF is generally insensitive to the random seed and can reliably enforce fairness. Recall that we have focused on minimizing statistical distance of learned representations because, as mentioned earlier, Madras et al. [68] have shown that fairness metrics such as demographic parity, equalized odds and equal opportunity can all be bounded by statistical distance. For example, FNF reduces the demographic parity distance of a classifier on Health from 0.39 to 0.08 with an accuracy drop of 3.9% (we also show that FNF has good performance for equalized odds and equality of opportunity [4]).

BOUNDING ADVERSARIAL ACCURACY Recall that the guarantees provided by FNF hold for estimated densities $\hat{p}_0$ and $\hat{p}_1$. Namely, the maximum adversarial accuracy for predicting whether the latent representation $z$ originates from distribution $\hat{\mathcal{Z}}_0$ or $\hat{\mathcal{Z}}_1$ is bounded by $(1 + \Delta(\hat{p}_{Z_0}, \hat{p}_{Z_1}))/2$. In this experiment, we investigate how well these guarantees transfer to the underlying distributions $\mathcal{Z}_0$ and $\mathcal{Z}_1$. In Fig. 6.5 we show our upper bound

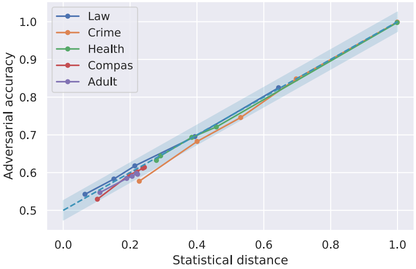


Figure 6.5: Bounding adversarial accuracy.

| | Acc | Adv Acc | | Max Adv Acc |
|---|---|---|---|---|
| | | $g \in \mathcal{G}$ | $g \notin \mathcal{G}$ | |
| Adv Forgetting [76] | 85.99 | 66.68 | 74.50 | ✗ |
| MaxEnt-ARL [74] | 85.90 | 50.00 | 85.18 | ✗ |
| LAFTR [68] | **86.09** | 72.05 | 84.58 | ✗ |
| FNF (our work) | 84.43 | N/A | **59.56** | **61.12** |

Table 6.1: Adversarial fair representation learning methods are only fair w.r.t. adversaries from a training family $\mathcal{G}$ while FNF provides a provable upper bound on the maximum accuracy of *any* adversary.

on the adversarial accuracy computed from the statistical distance using the estimated densities (diagonal dashed line), together with adversarial accuracies obtained by training an adversary, a multilayer perceptron (MLP) with two hidden layers of 50 neurons, for each model from Fig. 6.4. We also show 95% confidence intervals obtained using the Hoeffding bound from Lemma 2. We observe that our upper bound from the estimated densities $\hat{p}_0$ and $\hat{p}_1$ provides a tight upper bound on the adversarial accuracy for the true distributions $\mathcal{Z}_0$ and $\mathcal{Z}_1$. This demonstrates that, even though the exact constants from Thm. 3 are intractable, our density estimate is good enough in practice, and our bounds hold for adversaries on the true distribution.

COMPARISON WITH ADVERSARIAL TRAINING We now compare FNF with adversarial fair representation learning methods on Adult dataset: LAFTR-DP ($\gamma = 2$) [68], MaxEnt-ARL ($\alpha = 10$) [74], and Adversarial Forgetting ($\rho = 0.001, \delta = 1, \lambda = 0.1$) [76]. We train with a family of adversaries $\mathcal{G}$ trying to predict the sensitive attribute from the latent representation. Here, the families $\mathcal{G}$ are MLPs with 1 hidden layer of 8 neurons for LAFTR-DP, and 2 hidden layers with 64 neurons and 50 neurons for MaxEnt-ARL and Adversarial Forgetting, respectively. In Table 6.1 we show that these methods generally prevent adversaries from $\mathcal{G}$ to predict the sensitive attributes. However, we can still attack these representations using either larger MLPs (3 layers of 200 neurons for LAFTR-DP) or simple preprocessing steps (for MaxEnt-ARL and Adversarial Forgetting) as proposed by Gupta et al. [158] (essentially reproducing their results). Our results confirm findings from prior work [77, 157, 158]: adversarial training provides no guarantees against adversaries outside $\mathcal{G}$. In contrast, FNF computes a provable upper bound on the accuracy of *any* adversary for the estimated input distribution, and Table 6.1 shows that this extends to the true distribution. FNF thus learns representations with significantly lower adversarial accuracy with only minor decrease in task accuracy.

ALGORITHMIC RECOURSE WITH FNF We next experiment with FNF's bijectivity to perform recourse, i.e., reverse an unfavorable outcome, which is considered to be fundamental to explainable algorithmic decision-making [193]. To that end, we apply FNF with $\gamma = 1$ to the Law School dataset with three features: LSAT score, GPA, and the college to which the student applied (ordered decreasingly in admission rate). For all rejected applicants, i.e., $\boldsymbol{x}$ such that $h(f_a(\boldsymbol{x})) = h(\boldsymbol{z}) = 0$, we compute the closest $\tilde{\boldsymbol{z}}$ (corresponding to a point $\tilde{\boldsymbol{x}}$ from the dataset) w.r.t. the $\ell_2$-distance in latent space such that $h(\tilde{\boldsymbol{z}}) = 1$. We then linearly interpolate between $\boldsymbol{z}$ and $\tilde{\boldsymbol{z}}$ to find a (potentially crude) approximation of the closest point to $\boldsymbol{z}$ in latent space with positive prediction. Using the bijectivity of our encoders, we can compute the corresponding average feature change in the original space that would have caused a positive decision: increasing LSAT by 4.2 (non-whites) and 7.7 (whites), and increasing GPA by 0.7 (non-whites) and 0.6 (whites), where we only report recourse in the cases where the college does not change since this may not be actionable advice for certain applicants [194, 195, 196].

FLOW ARCHITECTURES In the next experiment we compare the Real-NVP encoder with an alternative encoder based on the Neural Spline Flows architecture [197] for the Crime dataset. In Table 6.2 we show the statistical distance and accuracy for models obtained using different values for the tradeoff parameter $\gamma$. We can observe that both flows offer similar performance. Note that FNF will benefit from future advances in normalizing

| | RealNVP | | NSF | |
|---|---|---|---|---|
| $\gamma$ | $\Delta$ | Acc | $\Delta$ | Acc |
| 0.00 | 1.00 | 0.85 | 1.00 | 0.84 |
| 0.02 | 0.70 | 0.85 | 0.71 | 0.85 |
| 0.10 | 0.53 | 0.83 | 0.54 | 0.83 |
| 0.90 | 0.23 | 0.69 | 0.24 | 0.69 |

Table 6.2: FNF performance with different flow encoder architectures.

flows research, as it is orthogonal to the concrete flow architecture that is used for training.

TRANSFER LEARNING Unlike prior work, transfer learning with FNF requires no additional reconstruction loss since both encoders are invertible and thus preserve all information about the input data. To demonstrate this, we follow the setup from Madras et al. [68] and train a model to predict the Charlson Index for the Health Heritage Prize dataset. We then transfer the learned encoder and train a classifier for the task of predicting the primary condition group. Our encoder reduces the statistical distance from 0.99 to 0.31 (this is independent of the label). For the primary condition group MSC2a3 we retain the accuracy at 73.8%, while for METAB3 it slightly decreases from 75.4% to 73.1%.

## 6.6 DISCUSSION

In this chapter, we introduced Fair Normalizing Flows (FNF), a method for learning representations ensuring that no adversary can predict sensitive attributes at the cost of a small accuracy decrease. This guarantee is stronger than prior work which only considers adversaries from a restricted model family. The key idea is to use an encoder based on normalizing flows which allows computing the exact likelihood in the latent space, given an estimate of the input density. Our experimental evaluation on several datasets showed that FNF effectively enforces fairness without significantly sacrificing utility, while simultaneously allowing interpretation of the representations and transferring to unseen tasks.

IMPACT The major step forward of FNF was the introduction of the concept of high confidence fairness guarantees for *any* classifier trained on top of the learned representations. The main drawback was that the guarantees only hold when we know the input distribution, which is rarely the case in practice. We successfully addressed this problem in our follow-up work FARE [11] where the key insight was to restrict the representation space of the encoder (using e.g. decision trees) which allows us to obtain high confidence guarantees via sampling, independent of the actual input distribution. There has also been other work [198] in this direction, further going into the direction of obtaining representations with high-confidence guarantees. Finally, the general problem of inferring sensitive attributes from processed data has also been relevant in the context of privacy. In Staab et al. [26], we developed a new method for minimizing the granularity of collected data which can be seen as a form of representation learning. We also investigated [27] capabilities of large language models to infer such personal attributes from text, and the question of learning new representations of text which hide such information remains open.

# 7

# CONCLUSION AND OUTLOOK

In this thesis we presented novel approach for achieving and certifying robustness and fairness in the context of AI Safety using formal and statistical methods. The first part of the thesis deals with the problem of certifying and training provably robust neural networks. In Chapter 3, our contribution was a method for certifying robustness of neural networks to geometric transformations using a method of sampling and optimization. Then, in Chapter 4 we proposed a novel training algorithm that can train highly accurate and certifiably robust neural networks. In the second part of the thesis we switched to fairness. Chapter 5 and Chapter 6 proposed new learning methods for representations which are fair with respect to individual and group fairness notions, respectively. In each chapter, we also outlined its impact on future work and possible next research steps. As more and more tasks which were typically solved by different models, are now being performed using a single foundation model, one major research direction is to make sure that such foundation model is robust and fair. Moreover, the scope of vulnerabilities of these foundation models is far wider than the specifications described in this work. Instead of $l_\infty$ or geometric transformations, we need to consider all possible input text modifications, and fairness is often difficult to describe simply as individual or group fairness. We believe that techniques described in this thesis can be seen as valuable guidelines to tackle these novel frontier problems.

# BIBLIOGRAPHY

[1] Mislav Balunović, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin Vechev. "Certifying Geometric Robustness of Neural Networks". In: *Advances in Neural Information Processing Systems*. 2019.

[2] Mislav Balunović and Martin T. Vechev. "Adversarial Training and Provable Defenses: Bridging the Gap". In: *ICLR*. OpenReview.net, 2020.

[3] Anian Ruoss, Mislav Balunović, Marc Fischer, and Martin T. Vechev. "Learning Certified Individually Fair Representations". In: *Advances in Neural Information Processing Systems 33*. 2020.

[4] Mislav Balunović, Anian Ruoss, and Martin T. Vechev. "Fair Normalizing Flows". In: *ICLR*. OpenReview.net, 2022.

[5] Anian Ruoss, Maximilian Baader, Mislav Balunović, and Martin T. Vechev. "Efficient Certification of Spatial Robustness". In: *AAAI*. AAAI Press, 2021, 2504.

[6] Mark Niklas Müller, Mislav Balunović, and Martin T. Vechev. "Certify or Predict: Boosting Certified Robustness with Compositional Architectures". In: *ICLR*. OpenReview.net, 2021.

[7] Wonryong Ryou, Jiayu Chen, Mislav Balunović, Gagandeep Singh, Andrei Marian Dan, and Martin T. Vechev. "Scalable Polyhedral Verification of Recurrent Neural Networks". In: *CAV (1)*. Vol. 12759. Lecture Notes in Computer Science. Springer, 2021, 225.

[8] Tobias Lorenz, Anian Ruoss, Mislav Balunović, Gagandeep Singh, and Martin T. Vechev. "Robustness Certification for Point Cloud Models". In: *ICCV*. IEEE, 2021, 7588.

[9] Nikola Jovanović, Mislav Balunović, Maximilian Baader, and Martin T. Vechev. "On the Paradox of Certified Training". In: *Trans. Mach. Learn. Res.* 2022 (2022).

[10] Momchil Peychev, Anian Ruoss, Mislav Balunović, Maximilian Baader, and Martin T. Vechev. "Latent Space Smoothing for Individually Fair Representations". In: *ECCV (13)*. Vol. 13673. Lecture Notes in Computer Science. Springer, 2022, 535.

[11] Nikola Jovanovic, Mislav Balunović, Dimitar Iliev Dimitrov, and Martin T. Vechev. “FARE: Provably Fair Representation Learning with Practical Certificates”. In: *ICML*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, 15401.

[12] Mislav Balunović, Dimitar Iliev Dimitrov, Robin Staab, and Martin T. Vechev. “Bayesian Framework for Gradient Leakage”. In: *ICLR*. OpenReview.net, 2022.

[13] Dimitar Iliev Dimitrov, Mislav Balunović, Nikola Konstantinov, and Martin T. Vechev. “Data Leakage in Federated Averaging”. In: *Trans. Mach. Learn. Res.* 2022 (2022).

[14] Mislav Balunović, Dimitar I. Dimitrov, Nikola Jovanovic, and Martin T. Vechev. “LAMP: Extracting Text from Gradients with Language Model Priors”. In: *NeurIPS*. 2022.

[15] Mark Vero, Mislav Balunović, Dimitar Iliev Dimitrov, and Martin T. Vechev. “TabLeak: Tabular Data Leakage in Federated Learning”. In: *ICML*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, 35051.

[16] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. “Graph convolutional neural networks for web-scale recommender systems”. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, 974.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NIPS*. 2012, 1106.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. “Mask R-CNN”. In: *ICCV*. IEEE Computer Society, 2017, 2980.

[19] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015).

[20] OpenAI. “GPT-4 Technical Report”. In: *CoRR* abs/2303.08774 (2023).

[21] Bohang Zhang, Du Jiang, Di He, and Liwei Wang. “Rethinking Lipschitz Neural Networks and Certified Robustness: A Boolean Function Perspective”. In: *NeurIPS*. 2022.

[22] Mark Niklas Müller, Franziska Eckert, Marc Fischer, and Martin T. Vechev. “Certified Training: Small Boxes are All You Need”. In: *ICLR*. OpenReview.net, 2023.

[23] Yuhao Mao, Mark Niklas Müller, Marc Fischer, and Martin T. Vechev. "TAPS: Connecting Certified and Adversarial Training". In: *CoRR* abs/2305.04574 (2023).

[24] Elias Benussi, Andrea Patanè, Matthew Wicker, Luca Laurenti, and Marta Kwiatkowska. "Individual Fairness Guarantees for Neural Networks". In: *IJCAI*. ijcai.org, 2022, 651.

[25] Danilo Jimenez Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[26] Robin Staab, Nikola Jovanovic, Mislav Balunović, and Martin T. Vechev. "From Principle to Practice: Vertical Data Minimization for Machine Learning". In: *SP*. IEEE, 2024.

[27] Robin Staab, Mark Vero, Mislav Balunović, and Martin T. Vechev. "Beyond Memorization: Violating Privacy Via Inference with Large Language Models". In: *CoRR* abs/2310.07298 (2023).

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

[29] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. "Evasion Attacks against Machine Learning at Test Time". In: *European Conference on Machine Learning and Knowledge Discovery in Databases, (ECML/PKDD)*. 2013.

[30] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale". In: *International Conference on Learning Representations* (2017).

[31] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations*. 2015.

[32] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *6th International Conference on Learning Representations*. 2018.

[33] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 IEEE Symposium on Security and Privacy (S&P)*. 2017.

[34] Anish Athalye, Nicholas Carlini, and David A. Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

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

[36] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation". In: *IEEE Symposium on Security and Privacy*. 2018.

[37] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. "Fast and Effective Robustness Certification". In: *Advances in Neural Information Processing Systems, (NeurIPS)*. 2018.

[38] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. "Efficient Neural Network Robustness Certification with General Activation Functions". In: *Advances in Neural Information Processing Systems, (NeurIPS)*. 2018.

[39] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. "Towards Fast Computation of Certified Robustness for ReLU Networks". In: *International Conference on Machine Learning, (ICML)*. 2018.

[40] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. "A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks". In: *Advances in Neural Information Processing Systems, (NeurIPS)*. 2019.

[41] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli. "A Dual Approach to Scalable Verification of Deep Networks". In: *Uncertainty in Artificial Intelligence, (UAI)*. 2018.

[42] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and Pawan Kumar Mudigonda. "A Unified View of Piecewise Linear Neural Network Verification". In: *Advances in Neural Information Processing Systems, (NeurIPS)*. 2018.

[43] Rüdiger Ehlers. "Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks". In: *Automated Technology for Verification and Analysis, (ATVA)*. 2017.

[44] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks". In: *Computer Aided Verification - 29th International Conference*. 2017.

[45] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming". In: *7th International Conference on Learning Representations*. 2019.

[46] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. "Formal Security Analysis of Neural Networks using Symbolic Intervals". In: *USENIX Security Symposium*. 2018.

[47] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. "Reachability Analysis of Deep Neural Networks with Provable Guarantees". In: *International Joint Conference on Artificial Intelligence, (IJCAI)*. 2018.

[48] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. "Semidefinite relaxations for certifying robustness to adversarial examples". In: *Advances in Neural Information Processing Systems, (NeurIPS)*. 2018.

[49] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. "Efficient Formal Safety Analysis of Neural Networks". In: *Advances in Neural Information Processing Systems, (NeurIPS)*. 2018.

[50] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. "Boosting Robustness Certification of Neural Networks". In: *International Conference on Learning Representations, (ICLR)*. 2019.

[51] Matthew Mirman, Timon Gehr, and Martin T. Vechev. "Differentiable Abstract Interpretation for Provably Robust Neural Networks". In: *International Conference on Machine Learning, (ICML)*. 2018.

[52] Eric Wong and J. Zico Kolter. "Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope". In: *International Conference on Machine Learning, (ICML)*. 2018.

[53] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. "Certified Robustness to Adversarial Examples with Differential Privacy". In: *IEEE Symposium on Security and Privacy, (SP)*. 2019.

[54] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. "Second-order adversarial attack and certifiable robustness". In: *arXiv preprint arXiv:1809.03113* (2018).

[55] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified Adversarial Robustness via Randomized Smoothing". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[56] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. "An abstract domain for certifying neural networks". In: *Proc. ACM Program. Lang.* (2019).

[57] Chongli Qin, Krishnamurthy (Dj) Dvijotham, Brendan O'Donoghue, Rudy Bunel, Robert Stanforth, Sven Gowal, Jonathan Uesato, Grzegorz Swirszcz, and Pushmeet Kohli. "Verification of Non-Linear Specifications for Neural Networks". In: *International Conference on Learning Representations, (ICLR)*. 2019.

[58] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. "Towards practical verification of machine learning: The case of computer vision systems". In: *arXiv preprint arXiv:1712.01785* (2017).

[59] Matthias Hein and Maksym Andriushchenko. "Formal guarantees on the robustness of a classifier against adversarial manipulation". In: *Advances in Neural Information Processing Systems*. 2017.

[60] Eric Wong and Zico Kolter. "Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope". In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

[61] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified Defenses against Adversarial Examples". In: *International Conference on Learning Representations*. 2018.

[62] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. "Training verified learners with learned verifiers". In: *arXiv preprint arXiv:1805.10265* (2018).

[63] Matthew Mirman, Timon Gehr, and Martin Vechev. "Differentiable abstract interpretation for provably robust neural networks". In: *International Conference on Machine Learning*. 2018.

[64] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. “On the effectiveness of interval bound propagation for training verifiably robust models”. In: *arXiv preprint arXiv:1810.12715* (2018).

[65] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. “Certified Robustness to Adversarial Examples with Differential Privacy”. In: *2019 IEEE Symposium on Security and Privacy (S&P)* (2018).

[66] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. “Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers”. In: *arXiv preprint arXiv:1906.04584* (2019).

[67] Richard S. Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. “Learning Fair Representations”. In: *Proceedings of the 30th International Conference on Machine Learning*. 2013.

[68] David Madras, Elliot Creager, Toniann Pitassi, and Richard S. Zemel. “Learning Adversarially Fair and Transferable Representations”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

[69] Daniel Moyer, Shuyang Gao, Rob Brekelmans, Aram Galstyan, and Greg Ver Steeg. “Invariant Representations without Adversarial Training”. In: *Advances in Neural Information Processing Systems 31*. 2018.

[70] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. “The Variational Fair Autoencoder”. In: *4th International Conference on Learning Representations*. 2016.

[71] Harrison Edwards and Amos J. Storkey. “Censoring Representations with an Adversary”. In: *4th International Conference on Learning Representations*. 2016.

[72] Qizhe Xie, Zihang Dai, Yulun Du, Eduard H. Hovy, and Graham Neubig. “Controllable Invariance through Adversarial Feature Learning”. In: *Advances in Neural Information Processing Systems 30*. 2017.

[73] Jiaming Song, Pratyusha Kalluri, Aditya Grover, Shengjia Zhao, and Stefano Ermon. “Learning Controllable Fair Representations”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019.

[74] Proteek Chandan Roy and Vishnu Naresh Boddeti. “Mitigating Information Leakage in Image Representations: A Maximum Entropy Approach”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[75] Jiachun Liao, Chong Huang, Peter Kairouz, and Lalitha Sankar. “Learning Generative Adversarial RePresentations (GAP) under Fairness and Censoring Constraints”. In: *CoRR* abs/1910.00411 (2019).

[76] Ayush Jaiswal, Daniel Moyer, Greg Ver Steeg, Wael AbdAlmageed, and Premkumar Natarajan. “Invariant Representations through Adversarial Forgetting”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020.

[77] Rui Feng, Yang Yang, Yuehan Lyu, Chenhao Tan, Yizhou Sun, and Chunping Wang. “Learning Fair Representations via an Adversarial Framework”. In: *CoRR* abs/1904.13341 (2019).

[78] Elliot Creager, David Madras, Jörn-Henrik Jacobsen, Marissa A. Weis, Kevin Swersky, Toniann Pitassi, and Richard S. Zemel. “Flexibly Fair Representation Learning by Disentanglement”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[79] Francesco Locatello, Gabriele Abbati, Thomas Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. “On the Fairness of Disentangled Representations”. In: *Advances in Neural Information Processing Systems 32*. 2019.

[80] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. “An abstract domain for certifying neural networks”. In: *Proceedings of the ACM on Programming Languages* (2019).

[81] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. “A convex relaxation barrier to tight robust verification of neural networks”. In: *arXiv preprint arXiv:1902.08722* (2019).

[82] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. “Exploring the Landscape of Spatial Robustness”. In: *International Conference on Machine Learning, (ICML)*. 2019.

[83] Patrick Cousot and Nicolas Halbwachs. “Automatic Discovery of Linear Restraints Among Variables of a Program”. In: *Symposium on Principles of Programming Languages, (POPL)*. 1978.

[84] Can Kanbak, Seyed Mohsen Moosavi Dezfooli, and Pascal Frossard. "Geometric robustness of deep networks: analysis and improvement". In: *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)* (2018).

[85] Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng. "Measuring invariances in deep networks". In: *Advances in Neural Information Processing Systems, (NeurIPS)*. 2009.

[86] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "The robustness of deep networks: A geometrical perspective". In: *IEEE Signal Processing Magazine* (2017).

[87] Rima Alaifari, Giovanni S. Alberti, and Tandri Gauksson. "ADef: an Iterative Algorithm to Construct Adversarial Deformations". In: *International Conference on Learning Representations, (ICLR)*. 2019.

[88] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. "Spatially Transformed Adversarial Examples". In: *International Conference on Learning Representations, (ICLR)*. 2018.

[89] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. "Transforming Auto-Encoders". In: *Artificial Neural Networks and Machine Learning – ICANN 2011*. 2011.

[90] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems, NIPS*. 2015.

[91] Gagandeep Singh, Markus Püschel, and Martin T. Vechev. "Fast polyhedra abstract domain". In: *Symposium on Principles of Programming Languages, (POPL)*. 2017.

[92] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming". In: *International Conference on Learning Representations*. 2019.

[93] Gagandeep Singh, Timon Gehr, Markus Puschel, and Martin Vechev. "Boosting Robustness Certification of Neural Networks". In: *International Conference on Learning Representations*. 2019.

[94] Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin T. Vechev. "Beyond the Single Neuron Convex Barrier for Neural Network Certification". In: *Advances in Neural Information Processing Systems 32*. 2019.

[95] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* (2010).

[96] Pierre Hansen and Brigitte Jaumard. “Lipschitz optimization”. In: *Handbook of global optimization*. 1995.

[97] Elwin de Weerdt, Qiping Chu, and J. A. Mulder. “Neural Network Output Optimization Using Interval Analysis”. In: *IEEE Transactions on Neural Networks* (2009).

[98] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017.

[99] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. 2009.

[100] Matthew Mirman, Gagandeep Singh, and Martin Vechev. “A Provable Defense for Deep Residual Networks”. In: *arXiv preprint arXiv:1903.12519* (2019).

[101] Jeet Mohapatra, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. “Towards Verifying Robustness of Neural Networks Against A Family of Semantic Perturbations”. In: *CVPR*. Computer Vision Foundation / IEEE, 2020, 241.

[102] Marc Fischer, Maximilian Baader, and Martin T. Vechev. “Certified Defense to Image Transformations via Randomized Smoothing”. In: *NeurIPS*. 2020.

[103] Motasem Alfarra, Adel Bibi, Naeemullah Khan, Philip H. S. Torr, and Bernard Ghanem. “DeformRS: Certifying Input Deformations with Randomized Smoothing”. In: *AAAI*. AAAI Press, 2022, 6001.

[104] Eric Wong, Leslie Rice, and J. Zico Kolter. “Fast is better than free: Revisiting adversarial training”. In: *ICLR*. OpenReview.net, 2020.

[105] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. “Feature denoising for improving adversarial robustness”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[106] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. “Towards Stable and Efficient Training of Verifiably Robust Neural Networks”. In: *International Conference on Learning Representations*. 2020.

[107] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. "Scaling provable adversarial defenses". In: *Advances in Neural Information Processing Systems 31*. 2018.

[108] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. "A Dual Approach to Scalable Verification of Deep Networks." In: *UAI*. 2018.

[109] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. "Fast and Effective Robustness Certification". In: *Advances in Neural Information Processing Systems 31*. 2018.

[110] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. "The Zonotope Abstract Domain Taylor1+". In: *Computer Aided Verification, (CAV)*. 2009.

[111] Ping Li, Trevor J Hastie, and Kenneth W Church. "Nonlinear estimators and tail bounds for dimension reduction in l1 using cauchy random projections". In: *Journal of Machine Learning Research* (2007).

[112] Kai Y. Xiao, Vincent Tjeng, Nur Muhammad (Mahi) Shafiullah, and Aleksander Madry. "Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability". In: *International Conference on Learning Representations*. 2019.

[113] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. 2019.

[114] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2018.

[115] Ruben Martinez-Cantin, Kevin Tee, and Michael McCourt. "Practical Bayesian optimization in the presence of outliers". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. 2018.

[116] Rüdiger Ehlers. "Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks". In: *Automated Technology for Verification and Analysis - 15th International Symposium*. 2017.

[117] Weilin Xu. “Reproducibility issue”. In: (2019). URL: `https://github.com/deepmind/interval-bound-propagation/issues/1#issuecomment-492552237`.

[118] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. “Towards Stable and Efficient Training of Verifiably Robust Neural Networks”. In: *arXiv preprint arXiv:1906.06316* (2019).

[119] Tim Brennan, William Dieterich, and Beate Ehret. “Evaluating the predictive validity of the COMPAS risk and needs assessment system”. In: *Criminal Justice and Behavior* (2009).

[120] Amit Datta, Michael Carl Tschantz, and Anupam Datta. “Automated Experiments on Ad Privacy Settings”. In: *Proc. Priv. Enhancing Technol.* (2015).

[121] Amir E Khandani, Adlar J Kim, and Andrew W Lo. “Consumer credit-risk models via machine-learning algorithms”. In: *Journal of Banking & Finance* (2010).

[122] Latanya Sweeney. “Discrimination in online ad delivery”. In: *Commun. ACM* (2013).

[123] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. “Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings”. In: *Advances in Neural Information Processing Systems 29*. 2016.

[124] Solon Barocas and Andrew D. Selbst. “Big Data’s Disparate Impact”. In: *California Law Review* (2016).

[125] Marc Fischer, Mislav Balunović, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin T. Vechev. “DL2: Training and Querying Neural Networks with Logic”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[126] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. “iFair: Learning Individually Fair Data Representations for Algorithmic Decision Making”. In: *35th IEEE International Conference on Data Engineering*. 2019.

[127] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. “Operationalizing Individual Fairness with Pairwise Fair Representations”. In: *Proc. VLDB Endow.* (2019).

[128] Mikhail Yurochkin, Amanda Bower, and Yuekai Sun. "Training individually fair ML models with sensitive subspace robustness". In: *8th International Conference on Learning Representations*. 2020.

[129] Hanchen Wang, Nina Grgic-Hlaca, Preethi Lahoti, Krishna P. Gummadi, and Adrian Weller. "An Empirical Study on Learning Fairness Metrics for COMPAS Data with Human Supervision". In: *CoRR* abs/1910.10255 (2019).

[130] Christina Ilvento. "Metric Learning for Individual Fairness". In: *1st Symposium on Foundations of Responsible Computing*. 2020.

[131] Debarghya Mukherjee, Mikhail Yurochkin, Moulinath Banerjee, and Yuekai Sun. "Two Simple Ways to Learn Individual Fairness Metrics from Data". In: *arXiv preprint arXiv:2006.11439* (2020).

[132] Shivam Garg, Vatsal Sharan, Brian Hu Zhang, and Gregory Valiant. "A Spectral View of Adversarially Robust Features". In: *Advances in Neural Information Processing Systems 31*. 2018.

[133] A. Pensia, V. Jog, and P. Loh. "Extracting Robust and Accurate Features via a Robust Information Bottleneck". In: *IEEE Journal on Selected Areas in Information Theory* (2020).

[134] Sicheng Zhu, Xiao Zhang, and David Evans. "Learning Adversarially Robust Representations via Worst-Case Mutual Information Maximization". In: *arXiv preprint arXiv:2002.11798* (2020).

[135] Christopher Jung, Sampath Kannan, and Neil Lutz. "A Center in Your Neighborhood: Fairness in Facility Location". In: *arXiv preprint arXiv:1908.09041* (2019).

[136] Sepideh Mahabadi and Ali Vakilian. "Individual Fairness for $k$-Clustering". In: *arXiv preprint arXiv:2002.06742* (2020).

[137] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. "Counterfactual Fairness". In: *Advances in Neural Information Processing Systems 30*. 2017.

[138] Lu Zhang, Yongkai Wu, and Xintao Wu. "A Causal Framework for Discovering and Removing Direct and Indirect Discrimination". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 2017.

[139] David Madras, Elliot Creager, Toniann Pitassi, and Richard S. Zemel. "Fairness through Causal Awareness: Learning Causal Latent-Variable Models for Biased Data". In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 2019.

[140] Yoichi Chikahara, Shinsaku Sakaue, Akinori Fujino, and Hisashi Kashima. "Learning Individually Fair Classifier with Path-Specific Causal-Effect Constraint". In: *arXiv preprint arXiv:2002.06746* (2020).

[141] Cynthia Dwork and Christina Ilvento. "Fairness Under Composition". In: *10th Innovations in Theoretical Computer Science Conference*. 2018.

[142] Cynthia Dwork, Christina Ilvento, and Meena Jagadeesan. "Individual Fairness in Pipelines". In: *1st Symposium on Foundations of Responsible Computing*. 2020.

[143] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. "Fairness through awareness". In: *Innovations in Theoretical Computer Science 2012*. 2012.

[144] Matthew Jagielski, Michael J. Kearns, Jieming Mao, Alina Oprea, Aaron Roth, Saeed Sharifi-Malvajerdi, and Jonathan R. Ullman. "Differentially Private Fair Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[145] Depeng Xu, Shuhan Yuan, and Xintao Wu. "Achieving Differential Privacy and Fairness in Logistic Regression". In: *Companion of The 2019 World Wide Web Conference*. 2019.

[146] Daniel McNamara, Cheng Soon Ong, and Robert C. Williamson. "Costs and Benefits of Fair Representation Learning". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 2019.

[147] Finale Doshi-Velez and Been Kim. "Considerations for Evaluation and Generalization in Interpretable Machine Learning". In: *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Springer International Publishing, 2018.

[148] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations*. 2015.

[149] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[150] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. "Boosting Robustness Certification of Neural Networks". In: *7th International Conference on Learning Representations*. 2019.

[151] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017.

[152] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. *Machine Bias*. 2016.

[153] F. Linda Wightman. *LSAC National Longitudinal Bar Passage Study*. 2017.

[154] Matthew Robert Wicker, Vihari Piratla, and Adrian Weller. "Certification of Distributional Individual Fairness". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[155] Florian E. Dorner, Momchil Peychev, Nikola Konstantinov, Naman Goel, Elliott Ash, and Martin T. Vechev. "Human-Guided Fair Classification for Natural Language Processing". In: *ICLR*. OpenReview.net, 2023.

[156] Yanai Elazar and Yoav Goldberg. "Adversarial Removal of Demographic Attributes from Text Data". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.

[157] Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. "A Theory of Usable Information under Computational Constraints". In: *8th International Conference on Learning Representations*. 2020.

[158] Umang Gupta, Aaron Ferber, Bistra Dilkina, and Greg Ver Steeg. "Controllable Guarantees for Fair Outcomes via Contrastive Information Estimation". In: *CoRR* abs/2101.04108 (2021).

[159] Congzheng Song and Vitaly Shmatikov. "Overlearning Reveals Sensitive Attributes". In: *8th International Conference on Learning Representations*. 2020.

[160] Moritz Hardt, Eric Price, and Nati Srebro. "Equality of Opportunity in Supervised Learning". In: *Advances in Neural Information Processing Systems 29*. 2016.

[161] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. "Fairness-aware Learning through Regularization Approach". In: *Data Mining Workshops (ICDMW)*. 2011.

[162] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. "Fairness Constraints: Mechanisms for Fair Classification". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. 2017.

[163] Mattia Cerrato, Marius Köppel, Alexander Segner, and Stefan Kramer. "Fair Group-Shared Representations with Normalizing Flows". In: *arXiv preprint arXiv:2201.06336* (2022).

[164] Meredith Whittaker, Kate Crawford, Roel Dobbe, Genevieve Fried, Elizabeth Kaziunas, Varoon Mathur, Sarah Mysers West, Rashida Richardson, Jason Schultz, and Oscar Schwartz. *AI now report 2018*. AI Now Institute at New York University New York, 2018.

[165] EU. *Proposal for a Regulation laying down harmonised rules on artificial intelligence*. 2021.

[166] FTC. *Aiming for truth, fairness, and equity in your company's use of AI*. 2021. URL: `https://www.ftc.gov/news-events/blogs/business-blog/2021/04/aiming-truth-fairness-equity-your-companys-use-ai`.

[167] Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. "Verifying Individual Fairness in Machine Learning Models". In: *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence*. 2020.

[168] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. "Perfectly parallel fairness certification of neural networks". In: *Proc. ACM Program. Lang.* (2020).

[169] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Third Theory of Cryptography Conference*. 2006.

[170] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V. Nori. "FairSquare: probabilistic verification of program fairness". In: *Proc. ACM Program. Lang.* (2017).

[171] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. "Probabilistic verification of fairness properties via concentration". In: *Proc. ACM Program. Lang.* (2019).

[172] Shahar Segal, Yossi Adi, Benny Pinkas, Carsten Baum, Chaya Ganesh, and Joseph Keshet. "Fairness in the Eyes of the Data: Certifying Machine-Learning Models". In: *CoRR* abs/2009.01534 (2020).

[173] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Nonlinear Independent Components Estimation". In: *3rd International Conference on Learning Representations*. 2015.

[174] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *CoRR* (2016).

[175] Diederik P. Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems 31*. 2018.

[176] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations*. 2014.

[177] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. 2014.

[178] Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. "Discrete Flows: Invertible Generative Models of Discrete Data". In: *NeurIPS*. 2019, 14692.

[179] Emiel Hoogeboom, Jorn W. T. Peters, Rianne van den Berg, and Max Welling. "Integer Discrete Flows and Lossless Compression". In: *NeurIPS*. 2019, 12134.

[180] Aditya Grover, Christopher Chute, Rui Shu, Zhangjie Cao, and Stefano Ermon. "AlignFlow: Cycle Consistent Learning from Multiple Domains via Normalizing Flows". In: *AAAI*. AAAI Press, 2020, 4028.

[181] Ben Usman, Avneesh Sud, Nick Dufour, and Kate Saenko. "Log-Likelihood Ratio Minimizing Flows: Towards Robust and Quantifiable Neural Distribution Alignment". In: *NeurIPS*. 2020.

[182] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. "A Kernel Method for the Two-Sample-Problem". In: *Advances in Neural Information Processing Systems 19*. 2006.

[183] Moritz Hardt and Eric Price. "Tight Bounds for Learning a Mixture of Two Gaussians". In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. 2015.

[184] Heinrich Jiang. "Uniform Convergence Rates for Kernel Density Estimation". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. 2017.

[185] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. "WaveNet: A Generative Model for Raw Audio". In: *The 9th ISCA Speech Synthesis Workshop*. 2016.

[186] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems 29*. 2016.

[187] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *Proceedings of the 33nd International Conference on Machine Learning*. 2016.

[188] Eric Wong and J. Zico Kolter. "Learning perturbation sets for robust machine learning". In: *CoRR* abs/2007.08450 (2020).

[189] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. "Generative Probabilistic Novelty Detection with Adversarial Autoencoders". In: *Advances in Neural Information Processing Systems 31*. 2018.

[190] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. "MADE: Masked Autoencoder for Distribution Estimation". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[191] Aditya Krishna Menon and Robert C. Williamson. "The cost of fairness in binary classification". In: *Conference on Fairness, Accountability and Transparency*. 2018.

[192] Han Zhao and Geoffrey J. Gordon. "Inherent Tradeoffs in Learning Fair Representations". In: *Advances in Neural Information Processing Systems 32*. 2019.

[193] Suresh Venkatasubramanian and Mark Alfano. "The philosophical basis of algorithmic recourse". In: *Conference on Fairness, Accountability, and Transparency*. 2020.

[194] Xin Zhang, Armando Solar-Lezama, and Rishabh Singh. "Interpreting Neural Network Judgments via Minimal, Stable, and Symbolic Corrections". In: *Advances in Neural Information Processing Systems 31*. 2018.

[195] Berk Ustun, Alexander Spangher, and Yang Liu. "Actionable Recourse in Linear Classification". In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 2019.

[196] Rafael Poyiadzi, Kacper Sokol, Raúl Santos-Rodríguez, Tijl De Bie, and Peter A. Flach. "FACE: Feasible and Actionable Counterfactual Explanations". In: *AAAI/ACM Conference on AI, Ethics, and Society*. 2020.

[197] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. “Neural spline flows”. In: *Advances in Neural Information Processing Systems* (2019).

[198] Yuhong Luo, Austin Hoag, and Philip S. Thomas. “Learning Fair Representations with High-Confidence Guarantees”. In: *CoRR* abs/2310.15358 (2023).