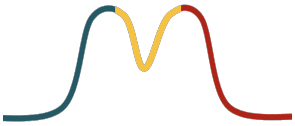
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Certified defences based on convex relaxations hurt generalisation

Master Thesis

Piersilvio De Bartolomeis

Thursday 10th November, 2022

Advisors: Jacob Clarysse, Dr. Amartya Sanyal, Prof. Dr. Fanny Yang
Department of Computer Science, ETH Zürich

A mio padre, per avermi trasmesso l'amore per la conoscenza.

Abstract

In recent years, much work has been devoted to designing certified defences for neural networks, i.e., methods for learning neural networks that are provably robust to certain adversarial perturbations. Due to the non-convexity of the problem, dominant approaches in this area rely on convex approximations, which are inherently loose. In this thesis, we question the effectiveness of such approaches for realistic computer vision tasks. First, we provide extensive empirical evidence to show that certified defences suffer not only worse accuracy but also worse robustness and fairness than empirical defences. We hypothesise that the reason why certified defences suffer in generalisation is (i) the large number of relaxed non-convex constraints and (ii) the strong alignment between the adversarial perturbations and the "signal" direction. We provide a combination of theoretical and experimental evidence to support these hypotheses.

Acknowledgements

First and foremost, I would like to thank my advisor Fanny Yang for welcoming me into her research group, organising a wonderful summer retreat in the Swiss Alps and, most importantly, teaching me not to fear vegetables. Likewise, I am very grateful to my advisors Jacob and Amartya for their invaluable guidance throughout this project, for their LaTeX-wizardry and, of course, for introducing me to the cult of Belgian beer. A big thanks to everyone else in the SML group, and in particular to Konstantin for teaching me a valuable lesson: beans do not grow on trees.

I am grateful to my fellow master students for many interesting chats during these years. To Luca and Antonio, for introducing me to Weight and Biases and for our movie nights. To Marco, for putting up with my blackboard presentations. To Mirko and Andrea, for the Uber Eats heists we pulled together. To Riccardo D., for our daily chats, for our collaborations and for introducing me to the world of RL. To Riccardo C., for being my continuous support and source of encouragement over the years.

I am grateful to all my friends from *Napulé*. To Luigi, for our daily chats and for sailing with me in the stormy waters of the current stock market. To Adriana, for always checking in on me and for enduring my rants about the terrible state of our country. To Gianluca and Mattia, for their vain attempts to take me to clubs. To Antonio and Michele, for keeping me entertained with movies and comic books. To Gaetano, for our unexpected encounters around the world and for our unhealthy addiction to Funko Pops. To everyone else I am forgetting to mention.

I am grateful to my family – especially my mother, brother and sister – for their unconditional love and support. A special thanks to *nonna* Adriana, for funding my education and treating me with the best homemade *limoncello*.

Last but not the least, I want to thank Algorithm 1, for getting me through the difficult times.

Contents

Acknowledgements iii

Contents v

1 Introduction 1
- 1.1 Problem formulation . . . 1
- 1.2 Summary of contributions and organisation . . . 2
- 1.3 Related work . . . 3

2 Adversarial robustness of neural networks 5
- 2.1 Robust optimisation . . . 5
  - 2.1.1 Solving the inner-maximisation . . . 6
- 2.2 Empirical defences . . . 7
  - 2.2.1 Fast gradient sign method . . . 7
  - 2.2.2 Projected gradient descent . . . 8
- 2.3 Certified defences . . . 8
  - 2.3.1 Convex outer adversarial polytope . . . 9
  - 2.3.2 Interval bound propagation . . . 12
- 2.4 Evaluation of adversarial defences . . . 15

3 Experiments on real-world and synthetic datasets 17
- 3.1 Experiments on real-world vision datasets . . . 17
  - 3.1.1 Certified defences hurt standard and robust error . . . 18
  - 3.1.2 Certified defences hurt fairness . . . 20
- 3.2 Experiments on synthetic datasets . . . 22
  - 3.2.1 Intuition: relaxed constraints and signal perturbations 23

4 Theoretical results for signal-directed perturbations 25
- 4.1 Certified defences for signal-directed perturbations . . . 25
- 4.2 Optimisation of a one-neuron neural network . . . 28
  - 4.2.1 Adversarial training . . . 28

4.2.2 Convex outer adversarial polytope 29
4.3 Signal-directed approximations hurt generalisation 30

5 Concluding thoughts **33**

A Deferred proofs **35**
A.1 Proof of Proposition 4.1 35
A.2 Proof of Theorem 4.2 36
A.3 Proof of Proposition 4.3 39
A.4 Proof of Proposition 4.4 41
A.5 Proof of Lemma 4.5 44
A.6 Proof of Theorem 4.6 45

B Auxiliary lemmas **49**
B.1 Upper bound on the exponential function 49
B.2 Lower bound on the difference of Gaussian CDFs 50
B.3 Upper bound on the ratio of Gaussian PDFs and CDFs 51

C Experimental details **53**
C.1 Real-world experiments with ℓ_2-ball perturbations 53
C.1.1 Datasets 53
C.1.2 Network architectures 55
C.1.3 Training details 56
C.2 Synthetic experiments with ℓ_2-ball perturbations 57
C.3 Synthetic experiments with signal-directed perturbations . . . 58

D Bonus: Tiramisù recipe **61**

Bibliography **63**

Chapter 1

Introduction

Deep neural networks have shown impressive performance in several tasks, ranging from computer vision and reinforcement learning to natural language processing [He et al., 2015, Krizhevsky et al., 2012, Collobert and Weston, 2008]. However, especially in computer vision, they have been shown to be vulnerable to adversarial attacks: imperceptible perturbations to the input can fool state-of-the-art classifiers [Biggio et al., 2013, Szegedy et al., 2014]. The existence of adversarial examples raises serious security concerns in many safety-critical applications such as vision for autonomous cars and healthcare [Tuncali et al., 2018, Finlayson et al., 2019]. Therefore, robustness to adversarial attacks has become a crucial design goal and various defence strategies have been proposed to that effect for modern deep learning architectures.

1.1 Problem formulation

In real-world scenarios, robustness against many different types of perturbations may be desired depending on the domain of application. Hence, in order to build robust models, we need to first define a threat model for the adversary. The most commonly used threat models in the literature are ℓ_p-ball perturbations, where $\mathcal{B}_\epsilon := \{\delta : \|\delta\|_p \leq \epsilon\}$ represents the set of allowed perturbations for some ℓ_p-ball with radius ϵ centred around the origin. For example, in Figure 1.1, we show that ℓ_∞-ball perturbations can fool neural networks into misclassifying an image of a panda as a gibbon.

Once a threat model is defined, we can formalise the problem of obtaining models robust to adversarial examples. For any distribution $\mathcal{D}$, neural network model $f_\theta : \mathbb{R}^d \to \mathcal{Y}$ parameterised by the weights $\theta \in \mathbb{R}^p$, and loss function L, we are interested in solving the following robust optimisation problem:

$$\min_\theta \mathbf{R}_\epsilon(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{B}_\epsilon} L(f_\theta(x+\delta), y) \right] \tag{1.1}$$

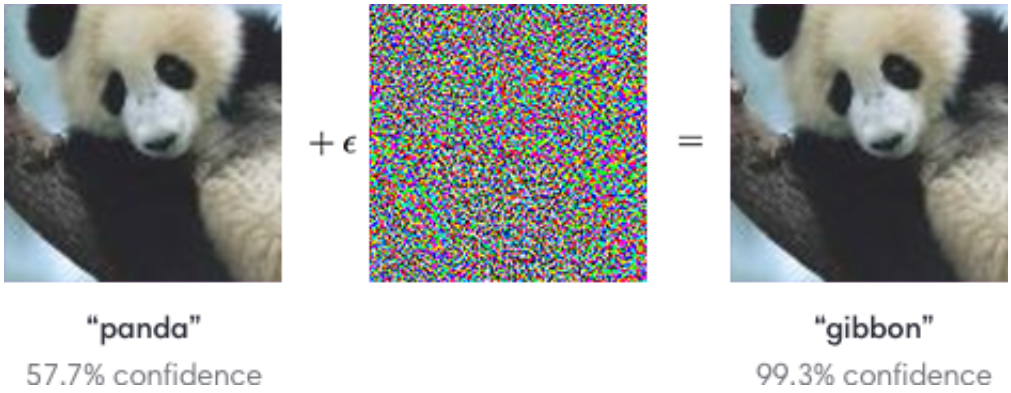
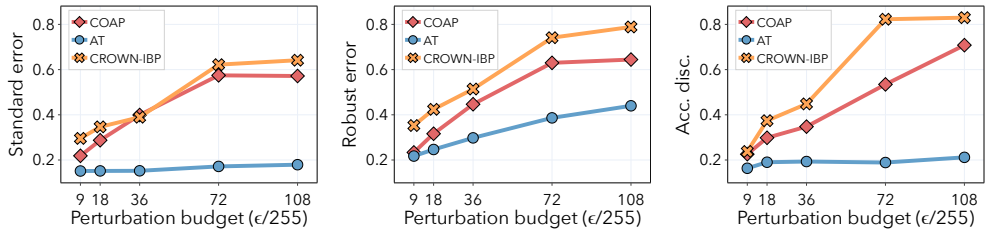


Figure 1.1: Imperceptible perturbations can fool deep neural networks classifiers in computer vision tasks. Source: Goodfellow et al. [2015].

We call $\mathbf{R}_\epsilon(\theta)$ the robust error when L is the 0-1 loss function. In practice, as the distribution $\mathcal{D}$ is unknown, we instead minimise the empirical robust error on a finite dataset D sampled from $\mathcal{D}$. Unfortunately, in the case of neural networks, the inner maximisation in Equation (1.1) is a non-convex optimisation problem and prohibitively hard to solve from a computational perspective [Katz et al., 2017, Weng et al., 2018]. Instead, two kinds of approaches are widely used to efficiently solve the problem: *empirical* defences that provide a lower bound on the solution and *certified* defences that provide an upper bound.

Among empirical defences, adversarial training (AT) [Goodfellow et al., 2015, Madry et al., 2018] is one of the few that has stood the test of time. AT minimises the worst-case empirical loss in Equation (1.1) by approximately solving the inner-maximisation problem with first-order gradient-based optimisation methods. However, despite its simplicity and computational efficiency, owing to its heuristic nature, AT does not provide any robustness guarantee. In many safety critical domains, such guarantees are of immense importance.

To address this limitation, much work has been devoted to designing certified defences, i.e., methods for learning neural networks that are provably robust to ℓ_p-ball perturbations on the training data. Many recent works [Wong and Kolter, 2018, Raghunathan et al., 2018, Dvijotham et al., 2018, Zhang et al., 2020] have proposed to solve a convex relaxation of the inner-maximisation problem by relaxing the non-convex ReLU constraint sets with convex ones. However, certified defences based on convex relaxations suffer from an inherent flaw: the upper bound they provide on the robust error is far from tight due to the looseness of the relaxation.



(a) Standard error **(b)** Robust error **(c)** Unfairness

Figure 1.2: Results for ℓ_2-ball perturbations on the CIFAR-10 test set. We compare ResNet architectures trained using state-of-the-art certified defenses CROWN-IBP [Zhang et al., 2020, Xu et al., 2020] and COAP [Wong et al., 2018, Wong and Kolter, 2018] against the most popular empirical defense to date AT [Madry et al., 2018, Goodfellow et al., 2015]. In Figures 1.2a, 1.2b and 1.2c we plot respectively standard error, robust error and accuracy discrepancy as the perturbation budget increases. We refer the reader to Chapter 3 for more details.

1.2 Summary of contributions and organisation

In this thesis, we argue that the inherent looseness of convex relaxations hinders the practical effectiveness of current certified defences. In particular, as shown in Figure 1.2, certified defences suffer significantly worse accuracy, robustness, and fairness on the test data compared to empirical defences.

The rest of the thesis is organised as follows:

- In Chapter 2, we provide the necessary background required to understand the rest of the thesis. First, we formalise the problem of learning neural networks that are robust to adversarial perturbations. Then, we present the key ideas behind most empirical and certified defences. Finally, we discuss the challenging problem of evaluating adversarial defences.
- In Chapter 3, we show that current certified defences hurt accuracy, robustness, and fairness across a range of ℓ_2-ball perturbations on real-world vision datasets. Further, we hypothesise that certified defences hurt generalisation because of (i) the large number of relaxed non-convex constraints and (ii) the strong alignment between the adversarial perturbations and the signal direction. We provide experimental evidence on synthetic datasets to support these hypotheses.
- In Chapter 4, we investigate hypothesis (ii) from a theoretical perspective. First, we introduce a new threat model consisting of perturbations that are perfectly aligned with the signal direction. Then, we extend the formulation of empirical and certified defences to this setting. Finally, we prove, for a simplified network in high-dimensions, that certified defences yield higher robust risk than empirical defences when the adversarial perturbation perfectly aligns with the signal direction.

1.3 Related work

We discuss here how our work relates to phenomena that have been observed in the literature before.

Looseness of convex relaxations The most relevant prior work investigating the looseness of convex relaxations is Salman et al. [2019]. They show that convex relaxations that approximate each ReLU output separately suffer an inherent tightness barrier. In particular, even the optimal convex relaxation cannot obtain tight bounds on the robust error. Subsequently, Singh et al. [2019] show that this barrier can be overcome by approximating multiple ReLUs outputs jointly. However, such approaches cannot be integrated into a training procedure due to their computational inefficiency. For this reason, this class of convex relaxations is not studied here. Instead, the thesis investigates how the tightness barrier affects generalisation and fairness for convex relaxations that approximate each ReLU output separately.

Randomised smoothing A popular alternative to certified defences based on convex relaxation is randomised smoothing [Cohen et al., 2019, Lécuyer et al., 2019, Li et al., 2019]. The key idea behind this technique is to transform an arbitrary classifier $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ into a "smoothed" classifier $g_\theta : \mathbb{R}^d \to \mathbb{R}^k$. In particular, for a given data point $x \in \mathbb{R}^d$ and variance σ^2, the smoothed classifier prediction $g_\theta(x)$ is defined as the most probable prediction by f_θ of the random variable $\mathcal{N}(x, \sigma^2 I_d)$. Despite its popularity, randomised smoothing still suffers from several limitations. For example, Mohapatra et al. [2021] investigate the side-effects of randomised smoothing and show that it significantly hurts the disparity in class-wise accuracy. Further, several works have exposed an accuracy-robustness tradeoff related to the smoothness of the classifier [Yang et al., 2020, Blum et al., 2020, Kumar et al., 2020].

Chapter 2

Adversarial robustness of neural networks

In this chapter, we provide the necessary background required to understand the rest of the thesis. First, we formalise the problem of learning neural networks that are robust to adversarial perturbations. Then, we present the key ideas behind most empirical and certified defences. Finally, we discuss the challenging problem of evaluating adversarial defences.

2.1 Robust optimisation

As we have seen in Chapter 1, the problem of finding a model that is robust to adversarial attacks can be formulated as a robust optimisation problem.

Recall, for any distribution $\mathcal{D}$, neural network model f_θ parameterised by the weights $\theta \in \mathbb{R}^p$, threat model $\mathcal{B}_\epsilon$, and loss function L, we are interested in solving the following optimisation problem:

$$\min_\theta \mathbf{R}_\epsilon(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta \in \mathcal{B}_\epsilon} L(f_\theta(x+\delta), y) \right] \tag{2.1}$$

We call $\mathbf{R}_\epsilon(\theta)$ the robust error when L is the 0-1 loss function. In practice, as the distribution $\mathcal{D}$ is unknown, we instead minimise the empirical robust error on a finite dataset D sampled from $\mathcal{D}$. For deep neural networks, the optimisation problem is solved using stochastic gradient descent by iteratively updating the network parameters θ according to:

$$\theta \leftarrow \theta - \frac{\eta}{|D|} \sum_{(x,y)\in D} \nabla_\theta \max_{\delta \in \mathcal{B}_\epsilon} L(f_\theta(x+\delta), y) \tag{2.2}$$

where η is a user-specified learning rate. However, computing the gradient of the inner-maximisation problem is a non-trivial problem. In practice, the

most widespread approach is computing the gradient of the loss function L at a maximiser of the inner problem. Specifically, we define $\delta^\star$ as the solution to the inner-maximisation problem:

$$\delta^\star = \underset{\delta \in \mathcal{B}_\epsilon}{\operatorname{argmax}}\, L(f_\theta(x+\delta), y) \tag{2.3}$$

and the gradient is given by:

$$\nabla_\theta \max_{\delta \in \mathcal{B}_\epsilon} L(f_\theta(x+\delta), y) = \nabla_\theta L(f_\theta(x+\delta^\star), y) \tag{2.4}$$

A priori, it is not clear whether this equality holds. Madry et al. [2018] showed that, if the loss L is continuously differentiable w.r.t. θ, then according to Danskin's theorem Equation (2.4) holds. However, there are two technical issues in applying Danskin's theorem when dealing with neural networks.

First of all, since most networks architectures use ReLU as activation function, the loss function L is not continuously differentiable w.r.t. θ. Nevertheless, since the set of discontinuities has measure zero, we can reasonably assume that this will not be an issue in practice. Secondly, Danskin's theorem requires that the inner-maximisation problem is solved exactly, i.e., that we can effectively find $\delta^\star$. Recall that solving the maximisation in Equation (2.3) is NP-hard in general, hence we are not able to compute the exact maximiser.

2.1.1 Solving the inner-maximisation

Despite the fact that the exact assumptions of Danskin's theorem do not hold for neural networks, this approach has turned out to be extremely successful in practice. Therefore, various strategies have been developed to efficiently solve the inner-maximisation problem. Below, we report the three different options.

Lower bound A widespread solution is to provide a lower bound on the maximisation problem. Note that by definition any feasible δ will give us a lower bound since:

$$L(f_\theta(x+\delta), y) \leq L(f_\theta(x+\delta^\star), y)\ ,\ \forall \delta \in \mathcal{B}_\epsilon \tag{2.5}$$

Hence, this approach consists in approximately solving the optimisation problem via heuristics, e.g., first-order gradient-based optimisation methods. We dub *empirical* defences all the adversarial defences based on a lower bound solution of the inner-maximisation.

Exact solution For ReLU networks, in principle, it is possible to solve the maximisation exactly via mixed-integer linear programming (MILP) solvers [Cheng et al., 2017, Dutta et al., 2018, Tjeng et al., 2019, Lomuscio

and Maganti, 2017] or satisfiability modulo theories (SMT) solvers [Scheibler et al., 2015, Katz et al., 2017, Carlini et al., 2018, Ehlers, 2017]. However, it has been shown that it is an NP-complete problem [Katz et al., 2017, Weng et al., 2018], and thus its computational complexity is exponential in the size of the network. For this reason, it is practically infeasible to design defences based on exact verifiers for modern deep neural networks.

Upper bound In domains where it is critical to have robustness guarantees, we might want to provide an upper bound on the solution. A dominant approach is to solve a convex relaxation of the inner-maximisation problem by relaxing the non-convex ReLU constraint sets with convex ones. This class of defences are provably robust to ℓ_p-ball perturbations on the training data. Hence, we dub *certified* defences all the adversarial defences based on an upper bound solution of the inner-maximisation.

In the following sections, we delve into *empirical* and *certified* defences which arises respectively from the lower and upper bounds on the inner-maximisation problem. We omit the exact combinatorial methods as they do not scale to neural networks of practical interest and cannot be integrated into any training procedure.

2.2 Empirical defences

We discuss here two popular methods for training neural networks that are empirically robust to ℓ_p-ball perturbations. Specifically, we present one of the first methods proposed in the deep learning literature, i.e., the Fast Gradient Sign Method (FSGM) [Goodfellow et al., 2015]. Further, we present its natural evolution, i.e., Projected Gradient Descent (PGD) [Madry et al., 2018].

2.2.1 Fast gradient sign method

The fast gradient sign method (FSGM) was first introduced by Goodfellow et al. [2015] and it is the one of the most computationally efficient method to solve the inner-maximisation problem. The key idea is to move δ in the direction of its gradient. In particular, let g be the gradient of the loss w.r.t. the perturbation δ:

$$g := \nabla_\delta L(f_\theta(x+\delta), y) \tag{2.6}$$

Then, to maximise the loss we can simply do gradient ascent, i.e., take a step for some user-defined learning rate η:

$$\delta \leftarrow \delta + \eta g \tag{2.7}$$

and then project back δ onto the ℓ_p-ball defined by $\mathcal{B}_\epsilon$. Goodfellow et al. [2015] focus on ℓ_∞-ball perturbations, hence projecting into the ℓ_∞-ball is

equivalent to clipping the values of δ within the range $[-\epsilon, \epsilon]$. In this setting, the update becomes:

$$\delta \leftarrow \begin{cases} +\epsilon & \text{if } \eta g > \epsilon \\ \eta g & \text{if } -\epsilon \leq \eta g \leq \epsilon \\ -\epsilon & \text{if } \eta g < -\epsilon \end{cases} \tag{2.8}$$

Next, observe that when the initial perturbation is $\delta = 0$, we can efficiently compute the gradient g via back-propagation:

$$g = \nabla_\delta L(f_\theta(x + \delta), y) \tag{2.9}$$

Finally, since we are only taking one step, we should make it as big as we can and thus δ will either be $+\epsilon$ or $-\epsilon$, depending on the sign of the gradient. This results in the FSGM update as described in Goodfellow et al. [2015]:

$$\delta \leftarrow \epsilon \operatorname{sgn}\left(\nabla_x L(f_\theta(x), y)\right) \tag{2.10}$$

2.2.2 Projected gradient descent

The projected gradient descent (PGD) method was first proposed by Madry et al. [2018]. It is a straightforward extension of FSGM which consists in iterating FSGM with a smaller step size η.

More concretely, let Π_ϵ be the projection onto the ℓ_p-ball defined by $\mathcal{B}_\epsilon$. Then, PGD consists in iterating the following update rule multiple times:

$$\delta \leftarrow \Pi_{\mathcal{B}_\epsilon}\left(\delta + \eta \nabla_\delta L(f_\theta(x + \delta), y)\right) \tag{2.11}$$

In practice, many tricks are used to improve the performance of this method. For example, Madry et al. [2018] initialise the perturbation to a random point in the ℓ_p-ball of interest and perform several random restarts. However, this flexibility comes at a cost. PGD has now three user-specified hyper-parameters: the step-size η, the number of iterations and the number of random restarts.

2.3 Certified defences

In this section, we describe two popular methods for training neural networks that are provably robust to ℓ_p-ball perturbations on the training data. In particular, we present the convex outer adversarial polytope (COAP) [Wong and Kolter, 2018, Wong et al., 2018], a method which relaxes the non-convex ReLU constraint sets with convex ones. Further, we present a more efficient class of methods, albeit at the cost of a looser robust error bound, which are based on interval bound propagation [Gowal et al., 2019, Mirman et al.,

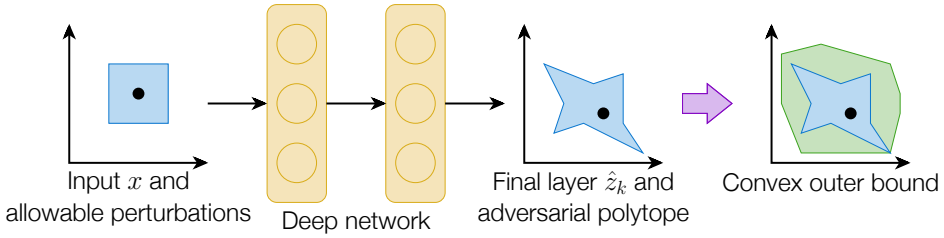


Figure 2.1: Illustration of the adversarial polytope and convex outer bound for ℓ_∞-ball perturbations. Source: Wong and Kolter [2018].

2018]. Finally, we introduce CROWN-IBP [Zhang et al., 2020], a method that combines convex relaxations with interval bound propagation and achieves state-of-the art certified robustness for ℓ_∞-ball perturbations.

For the rest of this section we consider a k-layers feed-forward ReLU network. In particular, we define $f_\theta(x) : \mathbb{R}^d \to \mathbb{R}^h$ as follows:

$$\begin{aligned} z_1 &= x \\ \hat{z}_{i+1} &= W_i z_i + b_i \ , \ i = 1, \dots, k-1 \\ z_i &= \text{ReLU}\left(\hat{z}_i\right) \ , \ i = 2, \dots, k-1 \\ f_\theta(x) &= \hat{z}_k \end{aligned} \tag{2.12}$$

where W_i represents a linear operator and $\theta = \{W_i, b_i\}_{i=1,\dots,k}$ is the set of network parameters. Further, we will refer to $\hat{z}$ and z as pre-activations and activations, respectively.

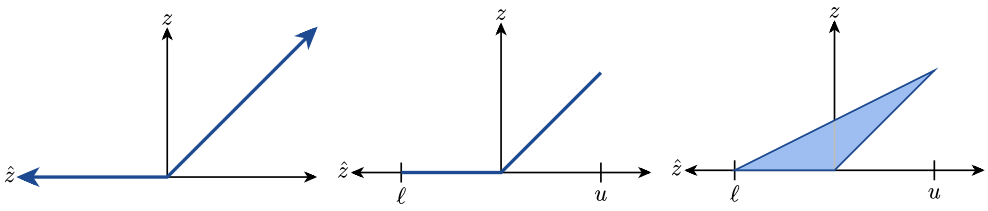
2.3.1 Convex outer adversarial polytope

We now describe the convex outer adversarial polytope (COAP) proposed by Wong and Kolter [2018]. As the name suggests, the key idea behind this method is to construct a convex outer bound to the adversarial polytope, i.e., the set of all final-layer activations $\hat{z}_k$ attainable by perturbing x with some $\delta \in \mathcal{B}_\epsilon$. Specifically, for any input x to the network, we define the adversarial polytope $\mathcal{Z}_\epsilon(x)$ as follows:

$$\mathcal{Z}_\epsilon(x) = \{f_\theta(x + \delta) : \delta \in \mathcal{B}_\epsilon\} \tag{2.13}$$

where $\mathcal{B}_\epsilon$ is any ℓ_p-ball with radius ϵ centred around the origin. We will focus on the ℓ_∞-ball perturbations for the rest of this section, but the method extends to other ℓ_p-balls and beyond as we will see in Section 4.1.

In Figure 2.1 we illustrate the adversarial polytope and the convex outer bound that we aim to construct. Armed with this definition, we now formulate the problem of verifying if an adversarial example exists as a constrained



(a) ReLU set **(b)** Bounded ReLU set **(c)** Convex ReLU set

Figure 2.2: In Figures 2.2a to 2.2c we illustrate the ReLU set, bounded ReLU set and convex envelope of the bounded ReLU set. Source: Wong and Kolter [2018].

optimisation problem. Given a data point x with known label y, we have:

$$\min_{\hat{z}_k} [\hat{z}_k]_y - [\hat{z}_k]_{\tilde{y}} \text{ s.t. } \hat{z}_k \in \mathcal{Z}_\epsilon(x) \tag{2.14}$$

where $\tilde{y} \neq y$ is any class label different from the true label. Now, if the solution to this problem is positive for all the alternative class labels we have a guarantee that no adversarial example exists within our threat model. Unfortunately, the problem in Equation (2.14) cannot be solved efficiently in its current form since $\mathcal{Z}_\epsilon(x)$ is in general a non-convex set.

Convex relaxation of the ReLU activations The first step towards efficiently solving the optimisation problem in Equation (2.14) is to relax the non-convex ReLU constraints. In particular, the ReLU constraints require that pre-activations and activations must lie in the non-convex set $\{(\hat{z}, z) : z = \text{ReLU}(\hat{z})\}$ as shown in Figure 2.2a.

The key idea is to replace the ReLU set with its convex envelope. One caveat, however, is that we need the set to be bounded. Specifically, we assume that the pre-activations $\hat{z}$ are bounded within the range $[\ell, u]$ for some lower bound ℓ and upper bound u. We illustrate the bounded ReLU set in Figure 2.2b and refer the reader to Wong and Kolter [2018] for an algorithm that computes the pre-activations bounds.

We can now relax the ReLU bounded set with its convex envelope:

$$\mathcal{C}(\ell, u) := \{(\hat{z}, z) : z \geq 0,\ z \geq \hat{z},\ (u - \ell)z \leq u\hat{z} - u\ell\} \tag{2.15}$$

as shown in Figure 2.2c. Further, for any data point x, let $\tilde{\mathcal{Z}}_\epsilon(x)$ be the outer bound on the adversarial polytope obtained from relaxing ReLU constraints. We can easily optimise over this set since it is convex, and it is a strictly larger set than the original adversarial polytope. This immediately implies that if no adversarial example exists in the outer bound $\tilde{\mathcal{Z}}_\epsilon(x)$, then none exists in the adversarial polytope $\mathcal{Z}_\epsilon(x)$ either.

Linear program formulation Armed with the convex outer bound we now turn the non-convex optimisation problem in Equation (2.14) into a linear program. Given a data point x with known label y, we have:

$$\min_{\hat{z}_k} [\hat{z}_k]_y - [\hat{z}_k]_{\tilde{y}} = c^\top \hat{z}_k \;\text{ s.t. }\; \hat{z}_k \in \tilde{\mathcal{Z}}_\epsilon(x) \tag{2.16}$$

Note that if we solve this linear program for all possible alternative classes $\tilde{y}$ and find that the objective is positive, then we know that no input perturbation can misclassify the example. Nevertheless, solving a linear program for every example in the dataset and every target class is still intractable for realistic networks.

Efficient optimisation via the dual problem In order to scale this approach to realistic neural networks, we consider the dual formulation of the linear program in Equation (2.16) and take a feasible solution. Since any feasible solution to the dual problem provides a lower bound on the solution of the primal problem, we still have a formal guarantee of robustness.

Specifically, in Theorem 2.1, we outline the dual problem formulation as described in Wong and Kolter [2018]. Most importantly, this theorem shows that the feasible set of the dual problem can be represented as a neural network, hence making the optimisation much more efficient.

Theorem 2.1 (Wong and Kolter [2018]) *Consider the neural network f_θ defined in Equation (2.12) and the ℓ_∞-ball perturbations threat model. Let $\ell_{i,j}$ and $u_{i,j}$ be the lower and upper bounds on the j-th component of the pre-activation vector $\hat{z}_i$. Further, we define $\mathcal{I}_i^-, \mathcal{I}_i^+$ and $\mathcal{I}_i$ as the sets of indexes j for which the lower and upper bounds of the pre-activation vector $\hat{z}_i$ are both negative, both positive or span zero, respectively.*

The dual of the linear program (2.16) can be written as

$$\begin{array}{ll} \max\limits_{\alpha} & \widetilde{J}_\epsilon\left(x, g_\theta(c, \alpha)\right) \\ s.t. & \alpha_{i,j} \in [0,1],\ \forall i, j \end{array} \tag{2.17}$$

where $\widetilde{J}_\epsilon(x, \nu)$ is equal to

$$\sum_{i=2}^{k-1} \sum_{j \in \mathcal{I}_i} \ell_{i,j} [\nu_{i,j}]_+ - \sum_{i=1}^{k-1} \nu_{i+1}^\top b_i - x^\top \hat{\nu}_1 - \epsilon \left\| \hat{\nu}_1 \right\|_1 \tag{2.18}$$

and $g_\theta(c, \alpha)$ is a k-layer neural network given by the equations

$$\begin{array}{ll} \nu_k = -c & \\ \hat{\nu}_i = W_i^\top \nu_{i+1} & \text{for } i = k-1, \ldots, 1 \\ \nu_{i,j} = 0 & \text{for } j \in \mathcal{I}_i^- \text{ and } i = k-1, \ldots, 2 \\ \nu_{i,j} = \hat{\nu}_{i,j} & \text{for } j \in \mathcal{I}_i^+ \text{ and } i = k-1, \ldots, 2 \\ \nu_{i,j} = \frac{u_{i,j}}{u_{i,j} - \ell_{i,j}} [\hat{\nu}_{i,j}]_+ - \alpha_{i,j} [\hat{\nu}_{i,j}]_- & \text{for } j \in \mathcal{I}_i \text{ and } i = k-1, \ldots, 2 \end{array} \tag{2.19}$$

We refer the reader to the Appendix of Wong and Kolter [2018] for a complete proof of the theorem.

Note that, only α is a free variable. Thus, we can compute a lower bound on the original linear program very efficiently setting α to be some fixed value and evaluating the objective term. Along the lines of Wong and Kolter [2018], we choose the fixed dual feasible solution:

$$\alpha_{i,j} = \frac{u_{i,j}}{u_{i,j} - \ell_{i,j}}, \quad \forall i, j \tag{2.20}$$

Upper bound on the inner maximisation Using the lower bound $\widetilde{J}_\epsilon$ we developed in Theorem 2.1, we can now obtain an upper bound on the worst case adversarial loss. Note that, in the theorem below, $\widetilde{J}_\epsilon$ is now vector valued as we evaluate the lower bound for all the labels $\tilde{y} \neq y$.

Theorem 2.2 (Wong and Kolter [2018]) *Let L be the cross-entropy loss function. For any data point (x, y), and $\epsilon > 0$, the worst case adversarial loss in Equation (2.1) can be upper bounded as follows:*

$$\max_{\|\delta\|_\infty \leq \epsilon} L(f_\theta(x+\delta), y) \leq L\left(-\widetilde{J}_\epsilon(x, g_\theta(e_y 1^\top - I)), y\right) \tag{2.21}$$

where e_y is the standard basis vector, $\widetilde{J}_\epsilon$ is as defined in Equation (2.18) for a given ϵ, and g_θ is as defined in Equation (2.19) for the given network parameters θ.

We refer the reader to the Appendix of Wong and Kolter [2018] for a complete proof of the theorem.

Scalability beyond MNIST Despite significantly improving on the computational efficiency of solving a linear program for every example in the dataset, the relaxation presented so far only scales to small-sized networks and datasets, e.g. MNIST. The main limitation is that computing the upper bound in Equation (2.21) scales quadratically with the number of hidden units of the network. Wong et al. [2018] make substantial progress towards scaling the method to bigger networks and datasets, e.g. CIFAR-10. In particular, they extend the techniques of Wong and Kolter [2018] to deal with skip connections and arbitrary activation functions. Further, they use random projections to empirically estimate the bound in Equation (2.21) in such a way that the worst case scales linearly with the number of hidden units of the network. We refer the reader to Wong et al. [2018] for an in-depth exposition of the scalability techniques.

2.3.2 Interval bound propagation

We now outline the key idea behind interval bound propagation (IBP) and provide a conceptual illustration of this method in Figure 2.3.

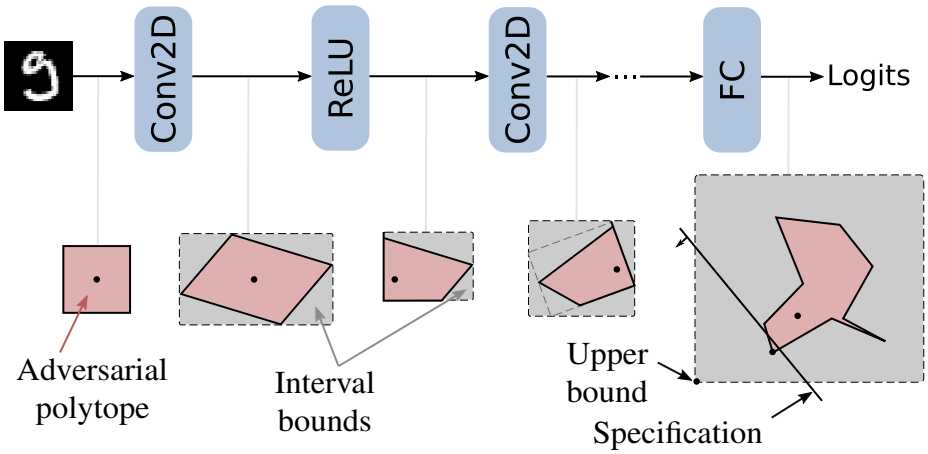


Figure 2.3: Illustration of interval bound propagation for ℓ_∞-ball perturbations. From the left, the adversarial polytope (in red) is propagated through a convolutional network. Interval bounds (in gray) can be propagated similarly: after each layer the bounds are reshaped to be axis-aligned bounding boxes that always encompass the adversarial polytope. Source: Gowal et al. [2019].

As in the previous section, we want to derive a lower bound on the solution of the optimisation problem in Equation (2.14). More concretely, we consider a k-layer feed-forward ReLU network as described in Equation (2.12). Assume, for the time being, that we have an interval bound on the second-to-last layer activations. Formally, we know some vectors ℓ and u such that:

$$\ell \leq z_{k-1} \leq u \tag{2.22}$$

where the inequality holds component-wise. Then we can lower bound the solution of the optimisation problem as follows:

$$\min_{z_{k-1}, \hat{z}_k} [\hat{z}_k]_y - [\hat{z}_k]_{\tilde{y}} = c^\top \hat{z}_k \tag{2.23}$$

$$\text{s.t. } \hat{z}_k = W_{k-1} z_{k-1} + b_{k-1} \text{ and } \ell \leq z_{k-1} \leq u \tag{2.24}$$

where $c = e_y - e_{\tilde{y}}$ and e is the standard basis vector. Further, we can eliminate the variable $\hat{z}_k$ and arrive at the following:

$$\min_{z_{k-1}} c^\top (W_{k-1} z_{k-1} + b_{k-1}) \tag{2.25}$$

$$\text{s.t. } \ell \leq z_{k-1} \leq u \tag{2.26}$$

Now, note that this optimisation problem has a simple analytical solution. In particular, z_{k-1} is given by:

$$[z_{k-1}]_j = \begin{cases} \ell & \text{if } [W_{k-1}^\top c]_j > 0 \\ u & \text{else} \end{cases} \tag{2.27}$$

Hence, we have successfully lower bounded the original non-convex optimisation problem in Equation (2.14) with the solution to a much simpler optimisation problem.

Computing lower and upper bounds So far, we have ignored the problem of computing lower and upper bounds on the ReLU activations in the second-to-last layer. We explain now how to efficiently compute ℓ and u for ℓ_∞-ball perturbations. In particular, our strategy will be to bound the pre-activations $\hat{z}_1$ in the first layer and propagate these bounds all the way to the second-to-last layer activations z_{k-1}.

First, we note that the ℓ_∞-ball threat model imposes component-wise interval bounds on z_1, which are given by:

$$[\ell_1]_j := [x]_j - \epsilon \leq [z_1]_j \leq [x]_j + \epsilon := [u_1]_j \tag{2.28}$$

We can now propagate these to obtain lower bounds on the pre-activations $\hat{z}_1$. Specifically, we solve the following optimisation problem:

$$\min_{z_1} W_1 z_1 + b_1 \tag{2.29}$$

$$\text{s.t. } \ell_1 \leq z_1 \leq u_1 \tag{2.30}$$

which has a simple analytical solution:

$$[z_1]_j = \begin{cases} \ell_1 & \text{if } [W_1]_j > 0 \\ u_1 & \text{else} \end{cases} \tag{2.31}$$

Along the same line, we can solve a maximisation problem instead of a minimisation problem to obtain the upper bounds. These are given by:

$$[z_1]_j = \begin{cases} \ell_1 & \text{if } [W_1]_j < 0 \\ u_1 & \text{else} \end{cases} \tag{2.32}$$

Finally, we repeat this bound propagation technique and clip the lower and upper bounds to zero for the ReLU operations until we reach the second-to-last layer activations z_{k-1}.

Related works This approach was first studied to some extent in Mirman et al. [2018], Dvijotham et al. [2018] but without achieving state-of-the-art performance. Subsequently, Gowal et al. [2019] significantly improved the performance of IBP by using a more sophisticated loss function and propagating the bounds only up to the second-to-last layer, hence improving the estimate of the worst-case logits. Note that the bounds we obtain from IBP are looser than those based on convex relaxations, but they have the advantage of being computationally efficient. In fact, the computational cost of IBP is comparable to two forward passes through the network.

Convex relaxations meet interval bound propagation Despite providing looser bounds on the robust error, interval bound propagation outperforms many convex relaxation-based methods in practice. However, it suffers from poor training stability due to the extremely loose bounds in the initial phase of training. To mitigate this issue, Zhang et al. [2020] propose to use IBP in a forward pass to compute all intermediate layers pre-activation bounds and use the tight convex relaxation CROWN [Zhang et al., 2018] to obtain the last layer bounds in a backward pass. Notably, the combination of IBP and CROWN into CROWN-IBP [Zhang et al., 2020] achieves state-of-the-art certified robustness under ℓ_∞-ball perturbations.

2.4 Evaluation of adversarial defences

Evaluating the robustness of adversarial defences is a complex challenge. In fact, finding adversarial examples is equivalent to solving the inner-maximisation problem in Equation (2.3), which is known to be an NP-hard problem. Hence, gradient-based first-order methods are widely used to overcome this computational barrier. Among these, projected gradient descent (PGD) is one of the most popular attacks to evaluate adversarial defences. However, robust evaluation based exclusively on PGD should be avoided because some defences obfuscate the gradients, making first-order methods ineffective. Athalye et al. [2018] show that defences relying on obfuscated gradients can be easily circumvented. Further, based on this observation, they break most of the adversarial defence papers at the ICLR 2018 conference prior to the review period completing.

In settings where gradients are obfuscated, we can evaluate robustness either using zeroth-order methods or designing adaptive attacks as described in Tramèr et al. [2020]. Adaptive attacks, however, cannot be automated and require careful tuning to the specific defence under consideration. For this reason, designing ad-hoc adaptive attacks is outside the scope of this thesis.

Nevertheless, we still want to ensure a reliable evaluation of the adversarial defences presented in this thesis. This is particularly important for the robust evaluation of empirical defences, as they come with no formal guarantee of robustness. Therefore, we rely on AutoAttack [Croce and Hein, 2020b] – a well established robust evaluation framework – for all our experiments. AutoAttack evaluation has two key strengths: (i) the diversity among the attacks and (ii) the parameter-free nature of the attacks. In particular, AutoAttack is an ensemble of four different parameter-free white-box and black-box attacks. Below, we provide a brief overview of the four attacks and refer the reader to the respective references for a more in-depth exposition.

APGD-CE and APGD-DLR [Croce and Hein, 2020b] Robust evaluation based on PGD has two main shortcomings. First, it relies on the user choice of a fixed step size. This is problematic as the performance of the algorithm is highly influenced by the choice of the step size [Mosbach et al., 2018]. Second, PGD is in general agnostic of the budget given to the attack. Croce and Hein [2020b] propose a budget-aware step size free variant of PGD, dubbed Auto-PGD (APGD), to address these shortcomings. Further, they include two variants of APGD in the AutoAttack ensemble to prevent gradient obfuscation: (i) APGD-CE which relies on the cross-entropy loss function and (ii) APGD-DLR which relies on the difference of logits loss function.

Square [Andriushchenko et al., 2020] Square Attack is a score-based black-box attack for ℓ_p-ball perturbations with $p \in \{2, \infty\}$. This method relies on random search instead of gradient information to find adversarial examples. By doing so, it is very effective on defences based on gradient obfuscation. The main idea behind the attack is to iteratively sample a random perturbation δ and to add this to the current adversarial example $\tilde{x}$ if it improves the objective function. It differs from classical random search techniques as the changes are localised in the image. In particular, as the name suggests, at each iteration the attack modifies only a small fraction of contiguous pixels shaped into squares.

FAB [Croce and Hein, 2020a] FAB is a white-box attack for ℓ_p-ball perturbations with $p \in \{1, 2, \infty\}$. This method aims to find minimal adversarial examples, i.e., the smallest perturbation resulting in a misclassification. More concretely, given a data point $x \in \mathbb{R}^d$ with label y, it solves the following optimisation problem:

$$\operatorname*{argmin}_{\delta \in \mathbb{R}^d} \|\delta\|_p \tag{2.33}$$

$$\text{s.t.} \max_{\tilde{y} \neq y} f_{\tilde{y}}(x + \delta) \geq f_y(x + \delta) \tag{2.34}$$

where $f : \mathbb{R}^d \to \mathbb{R}^k$ is a neural network, k is the number of classes and $f_y(x)$ is the predicted score that x belongs to class y. As opposed to PGD, this attack does not require threshold ϵ to be specified and, most importantly, it does not require a step size.

Chapter 3

Experiments on real-world and synthetic datasets

In this chapter, we show that current certified defences hurt standard error, robust error, and fairness on three widely used computer vision datasets: MNIST [LeCun et al., 1998], CIFAR-10 [Krizhevsky, 2009] and Tiny ImageNet [Le and Yang, 2015]. Further, we hypothesise that certified defences hurt generalisation because of (i) the large number of relaxed non-convex constraints and (ii) the strong alignment between the adversarial perturbations and the signal direction. We investigate these hypotheses on two synthetic data distributions: a linearly separable distribution and a distribution where the two classes are concentric spheres.

3.1 Experiments on real-world vision datasets

In this section, we compare certified and empirical defences on real-world vision datasets. Among certified defences, we consider the convex outer adversarial polytope (COAP) [Wong et al., 2018, Wong and Kolter, 2018], which achieves state-of-the-art certified robustness under ℓ_2-ball perturbations. Additionally, we consider CROWN-IBP [Zhang et al., 2018, Xu et al., 2020], which combines the tight convex relaxation CROWN [Zhang et al., 2018] with IBP [Gowal et al., 2019] and achieves state-of-the-art certified robustness under ℓ_∞-ball perturbations. As for empirical defences, we consider adversarial training (AT) [Madry et al., 2018, Goodfellow et al., 2015], which is one of the most popular and effective defences to date.

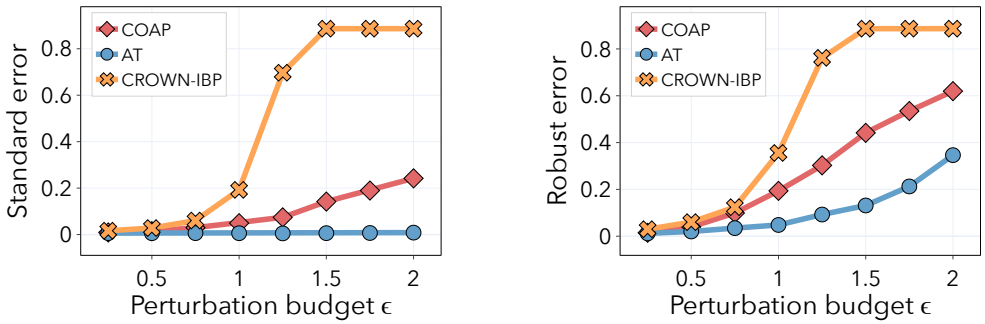
We refer the reader to Appendices C.1.1 and C.1.2 for a self-contained overview of the computer vision datasets and architectures used throughout this section.

Models and robust evaluation We consider the ℓ_2-ball perturbations threat model. To reliably evaluate the robust error, we use the strongest version of AutoAttack (AA+) [Croce and Hein, 2020b]. For CIFAR-10, we train a residual network (ResNet) and for MNIST we train a large convolutional neural network (CNN). Both architectures were introduced in Wong et al. [2018] as standard benchmarks for certified defences. For Tiny Imagenet, we train a WideResNet along the same lines of Xu et al. [2020]. We refer the reader to Appendix C.1.3 for complete experimental details.

3.1.1 Certified defences hurt standard and robust error

Several studies have shown that adversarial training may lead to an increase in standard error when compared with standard training [Raghunathan et al., 2020, Tsipras et al., 2019, Zhang et al., 2019]. Here, we observe the same phenomenon to a much higher degree in certified defences. Specifically, our experimental results show that certified defences not only suffer worse standard error but also worse robust error than adversarial training.

MNIST We discuss here the results on the MNIST dataset for ℓ_2-ball perturbations. First, we observe in Figure 3.1a that, for increasing perturbation budget, the standard error gap between certified (CROWN-IBP, COAP) and empirical defences (AT) increases. In particular, the gap reaches almost 90% for CROWN-IBP when $\epsilon = 1.5$. Secondly, we observe that the robust error gap increases with increasing perturbation budget in Figure 3.1b. In particular, the gap reaches almost 40% for the largest perturbation budgets.

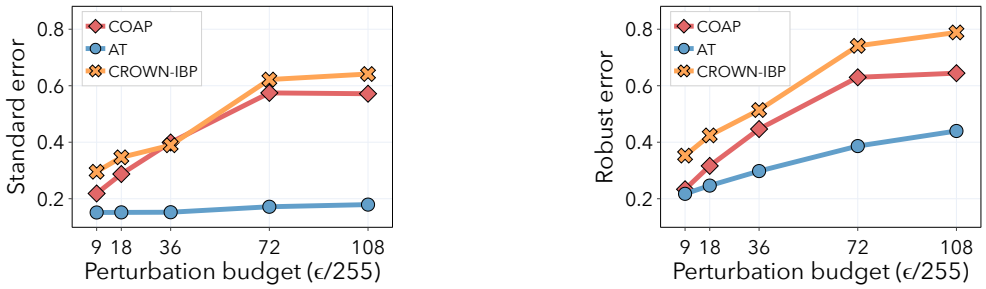


(a) Standard error **(b)** Robust error

Figure 3.1: Results for ℓ_2-ball perturbations on MNIST test set. In Figures 3.1a and 3.1b we plot, respectively, standard error and robust error for a CNN architecture trained on MNIST, as the perturbation budget ϵ increases.

CIFAR-10 We discuss here the results on the CIFAR-10 dataset for ℓ_2-ball perturbations. First, we observe in Figure 3.2a that, for increasing

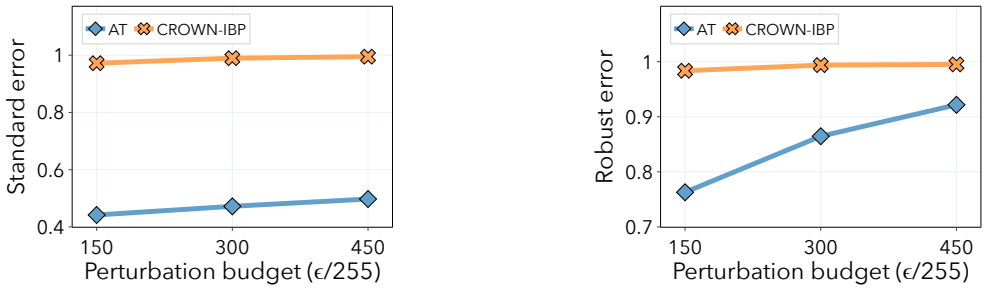
perturbation budget, the standard error gap between certified (CROWN-IBP, COAP) and empirical defences (AT) increases. In particular, the gap reaches almost 40% for the largest perturbation budgets. Secondly, we observe that the robust error gap increases with increasing perturbation budget in Figure 3.2b. In particular, the gap reaches almost 20% for the largest perturbation budgets.



(a) Standard error **(b)** Robust error

Figure 3.2: Results for ℓ_2-ball perturbations on CIFAR-10 test set. In Figures 3.2a and 3.2b we plot, respectively, standard error and robust error for a ResNet architecture trained on CIFAR-10, as the perturbation budget ϵ increases.

Tiny ImageNet We discuss here the results on the Tiny ImageNet dataset for ℓ_2-ball perturbations. We omit results for COAP in this setting as the method does not scale to such dataset and model sizes. First, we observe in Figure 3.3a a significant standard error gap between certified (CROWN-IBP) and empirical defences (AT). In particular, the gap reaches almost 60% for small perturbation budgets. Similarly, we observe a significant robust error gap increases in Figure 3.2b. In particular, the gap reaches almost 25% for small perturbation budgets.



(a) Standard error **(b)** Robust error

Figure 3.3: Results for ℓ_2-ball perturbations on Tiny ImageNet test set. In Figures 3.3a and 3.3b we plot, respectively, standard error and robust error for a WideResNet architecture trained on Tiny ImageNet, as the perturbation budget ϵ increases.

3.1.2 Certified defences hurt fairness

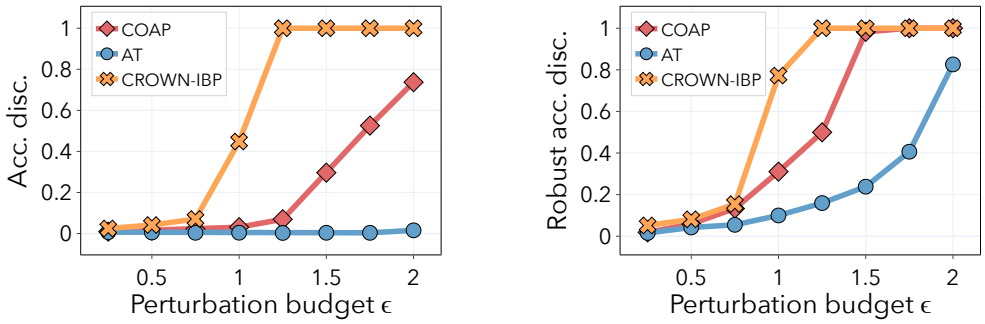
We now take one step further and show that certified defences (CROWN-IBP, COAP) suffer significantly worse fairness than empirical defences (AT).

Let $\mathbf{R}(\theta)$ be the standard error of the classifier f_θ and $\mathbf{R}^k(\theta)$ the standard error conditioned on the class label k. We measure the degree of unfairness as:

$$\frac{\max_k \mathbf{R}^k(\theta) - \mathbf{R}(\theta)}{1 - \mathbf{R}(\theta)} \tag{3.1}$$

Using the terminology in Sanyal et al. [2022], we refer to this metric as *accuracy discrepancy*. However, we expect our results to translate to other related fairness metrics as well [Duchi and Namkoong, 2018, Dwork et al., 2012, Hardt et al., 2016, Hébert-Johnson et al., 2018]. Additionally, we consider the discrepancy in robust accuracy, as it was observed in Xu et al. [2021] that adversarial defences may induce a large discrepancy of robustness among different classes. We refer to this metric as *robust accuracy discrepancy* and it consists in replacing the standard error with the robust error in Equation (3.1).

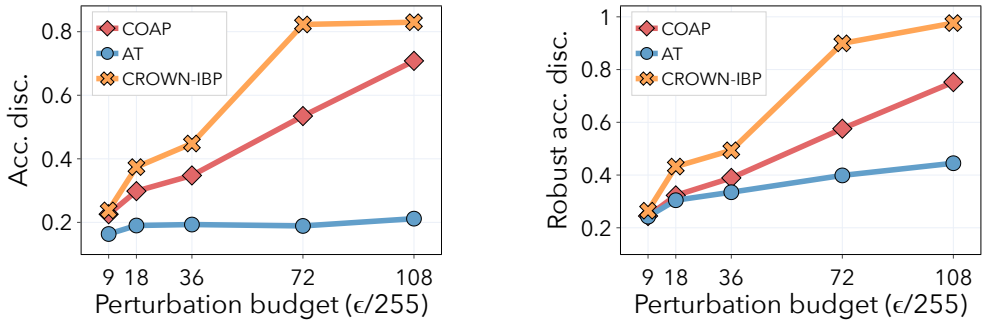
MNIST We discuss here the results on the MNIST dataset for ℓ_2-ball perturbations. We observe in Figures 3.4a and 3.4b that COAP and CROWN-IBP have a significant discrepancy for both standard and robust accuracy. For large perturbations, these methods obtain 100% discrepancy, meaning that their accuracy on the worst class can be as low as 0%. By contrast, AT preserves fairness for both standard and robust accuracy much better. In particular, the discrepancy for standard accuracy is always less than 2% for all perturbation budgets considered.



(a) Unfairness **(b)** Robust unfairness

Figure 3.4: Results for ℓ_2-ball perturbations on MNIST test set. In Figures 3.4a and 3.4b we plot, respectively, accuracy discrepancy and robust accuracy discrepancy for a CNN architecture trained on CIFAR-10, as the perturbation budget ϵ increases.

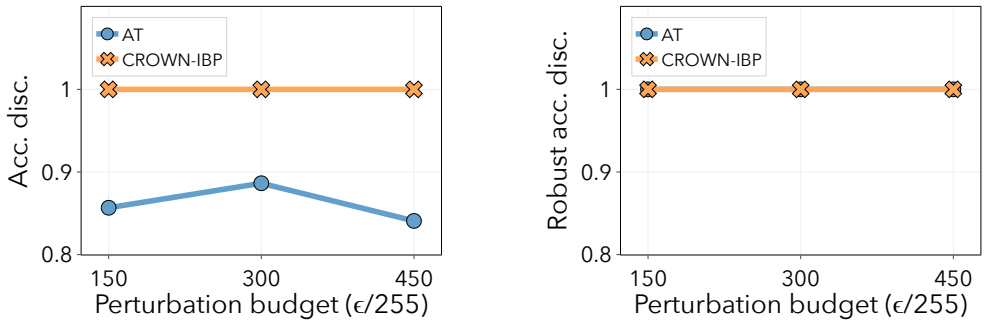
CIFAR-10 We discuss here the results on the CIFAR-10 dataset for ℓ_2-ball perturbations. AT maintains a constant accuracy discrepancy around 20% for all perturbation budgets considered, whereas for certified defences it steadily increases with the perturbation budget. Additionally, for robust accuracy, we observe a discrepancy gap of 35% between the best certified and empirical defences for the largest perturbation budget considered.



(a) Unfairness **(b)** Robust unfairness

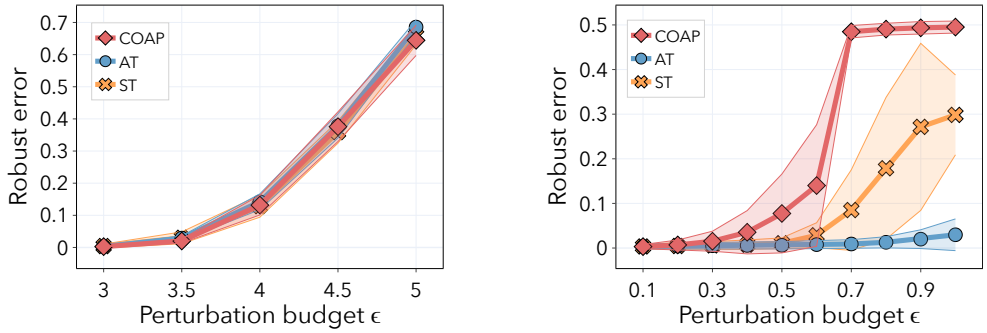
Figure 3.5: Results for ℓ_2-ball perturbations on CIFAR-10 test set. In Figures 3.5a and 3.5b we plot, respectively, accuracy discrepancy and robust accuracy discrepancy for a ResNet architecture trained on CIFAR-10, as the perturbation budget ϵ increases.

Tiny ImageNet We discuss here the results on the Tiny ImageNet dataset for ℓ_2-ball perturbations. We omit results for COAP as the method does not scale to such dataset and model sizes. AT maintains accuracy discrepancy between 80% and 90% for all perturbation budgets, whereas for certified defences it is constantly 100%. On the other hand, for robust accuracy, we observe a constant discrepancy of 100% for both empirical and certified defences.



(a) Unfairness **(b)** Robust unfairness

Figure 3.6: Results for ℓ_2-ball perturbations on Tiny ImageNet test set. In Figures 3.6a and 3.6b we plot, respectively, accuracy discrepancy and robust accuracy discrepancy for a WideResNet architecture trained on Tiny ImageNet, as the perturbation budget ϵ increases.



(a) Linearly separable **(b)** Concentric spheres

Figure 3.7: We report mean and standard deviation over 15 seeds. In Figures 3.7a and 3.7b we plot the robust error for standard training (ST), adversarial training (AT) and convex outer adversarial polytope (COAP), when training on the linearly separable and concentric spheres distributions respectively. See Appendix C.2 for complete experimental details.

3.2 Experiments on synthetic datasets

In this section, we hypothesise that certified defences hurt robust and standard generalisation because of (i) the large number of relaxed non-convex constraints and (ii) the strong alignment between the adversarial perturbations and the signal direction. We investigate these hypotheses on more controlled settings. In particular, we study two synthetic data distributions: a linearly separable distribution as in Clarysse et al. [2022], which is similar to distributions studied in Nagarajan and Kolter [2019], Tsipras et al. [2019], and the concentric spheres distribution studied in Gilmer et al. [2018], Nagarajan and Kolter [2019].

Data models We consider the linearly separable distribution where first, the label $y \in \{+1, -1\}$ is drawn with equal probability. Then, for some $\gamma > 0$, the covariate vector is $x = [\gamma \operatorname{sgn}(y); \tilde{x}]$, where $\tilde{x} \in \mathbb{R}^{d-1}$ is a random vector drawn from a standard normal distribution $\tilde{x} \sim \mathcal{N}\left(0, \sigma^2 I_{d-1}\right)$ and $[;]$ represents concatenation. We sample the concentric spheres dataset as follows; for $0 < R_1 < R_{-1}$, we first draw a binary label $y \in \{+1, -1\}$ with equal probability and then the covariate vector $x \in R^d$ is distributed uniformly on the sphere of radius R_y. Observe that achieving a low test error on the concentric spheres distribution requires a non-linear classifier.

Models and robust evaluation We compare certified defences (COAP) with empirical defences (AT). Additionally, we introduce standard training as a baseline, where the network is trained to minimise the standard cross-entropy loss. As for the threat model, we consider ℓ_2-ball perturbations. We evaluate robust error using Auto-PGD [Croce and Hein, 2020b] with 100 iterations and 5 random restarts.

Results In Figures 3.7a and 3.7b, we plot the robust error of standard training (ST), adversarial training (AT), and convex outer adversarial polytope (COAP) on the linear and concentric spheres distributions respectively. We observe that, in contrast to the linear setting, COAP has a much higher robust error on the concentric spheres distribution than AT and ST, where the gap increases with increasing perturbation budget ϵ. In the next section, we provide intuition as to why COAP has a much higher robust error than AT on the concentric spheres distribution.

3.2.1 Intuition: relaxed constraints and signal perturbations

The intuition is two-fold. First of all, COAP relaxes the non-convex ReLU constraints for all neurons that activate within the perturbation set, i.e., there exists $\delta \in \mathcal{B}_\epsilon$ for which the input to the neuron equals 0. Hence, the larger the percentage of relaxed neurons, the worse the approximation. This is formally captured by Theorem 2.1 in Section 2.3.1. Secondly, the ℓ_2-ball perturbations are significantly aligned with the signal direction, meaning that they effectively reduce the information about the label in the input data. We will prove that applying an approximation in this direction yields poor generalisation with Theorem 4.6 in Section 4.2 for the linearly separable distribution.

COAP relaxes many constraints on the concentric spheres In Figure 3.8 we empirically show that COAP convexly approximates a large number of constraints when trained on the concentric spheres distribution. In particular, we plot the percentage of active neurons on the concentric spheres and linear distributions as the perturbation budget increases. Note that the percentage is much higher for the concentric spheres than for the linearly separable distribution and increases with perturbation budget ϵ. Indeed, the complex spherical decision boundary requires much more active neurons compared to the linear decision boundary which only needs 1 active neuron.

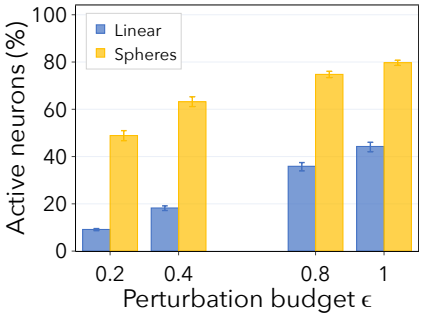
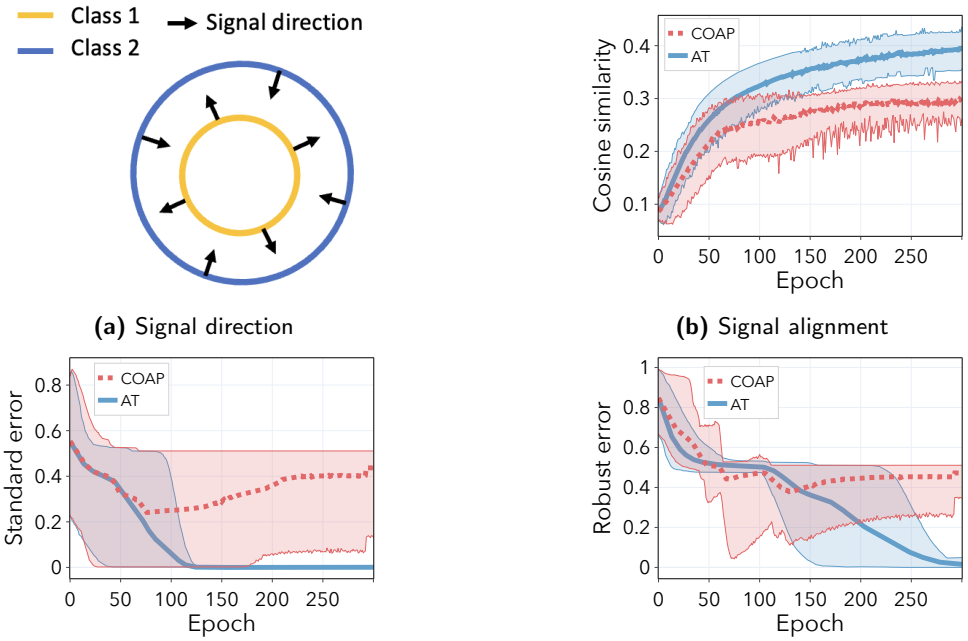


Figure 3.8: In Figure 3.8 we plot the percentage of neurons in the activation set for the linearly separable and concentric spheres distribution respectively. See Appendix C.2 for complete experimental details.



(a) Signal direction **(b)** Signal alignment

(c) Standard error **(d)** Robust error

Figure 3.9: We report mean and standard deviation over 15 seeds. In Figure 3.9a we plot a 2-dimensional illustration of the concentric spheres dataset, the black arrows indicate the signal direction. In Figure 3.9b we plot the cosine similarity between ℓ_2-ball perturbations on the training data (average) and the signal directed vector. In Figures 3.9c and 3.9d we plot, respectively, standard and robust error for adversarial training (AT) and convex outer adversarial polytope (COAP). We observe that when cosine similarity is high, the gap in standard and robust error between COAP and AT increases. Hence, especially approximations in the signal direction can hurt standard and robust generalisation.

ℓ_2-ball perturbations align with the signal direction We empirically show that ℓ_2-ball perturbations align with the signal direction on the concentric spheres distribution. Note that for a point x drawn from the concentric spheres distribution, the signal direction is given by $y\frac{x}{\|x\|_2}$ (see Figure 3.9a for a 2D visualization). In Figure 3.9b, we plot the cosine distance between the ℓ_2-ball perturbations computed on the training set, and the signal direction. Comparing Figures 3.9b to 3.9d, we see that during the early stages of training, the ℓ_2-ball perturbations are not aligned with the signal direction and the robust and standard errors for COAP are similar to AT. However, after some epochs, when the perturbations start to align with the signal direction, both the robust and standard error gaps between COAP and AT increase. This provides evidence that, as training progresses, ℓ_2-ball perturbations become significantly aligned with the signal direction and the generalisation gap worsens.

Chapter 4

Theoretical results for signal-directed perturbations

In this chapter, we further investigate our hypothesis that certified defences hurt generalisation when the adversarial perturbations are aligned with the signal direction. In particular, we study the linearly separable distribution from the previous section and assume that the adversarial attacks concentrate all of their perturbation budget along the signal direction. First, we extend the convex outer adversarial polytope (COAP) [Wong and Kolter, 2018] to signal-directed perturbations. Then, we prove in Theorem 4.6 for a simple neural network that, in high dimensions, certified defences (COAP) yield higher robust error than empirical defences (AT) for large perturbation budgets. Further, we corroborate our theoretical results with experimental evidence on synthetic data.

4.1 Certified defences for signal-directed perturbations

We now formulate the convex outer adversarial polytope (COAP) [Wong and Kolter, 2018] for adversaries that concentrate all their budget along the signal direction in the input. Our derivation can be seen as an extension of Wong and Kolter [2018], Erdemir et al. [2021].

Data and threat model We consider the linearly separable distribution described in Section 3.2. First, the label $y \in \{+1, -1\}$ is drawn with equal probability. Then, for some $\gamma > 0$, the covariate vector is $x = [\gamma \operatorname{sgn}(y); \tilde{x}]$, where $\tilde{x} \in \mathbb{R}^{d-1}$ is a random vector drawn from a standard normal distribution $\tilde{x} \sim \mathcal{N}\left(0, \sigma^2 I_{d-1}\right)$ and $[;]$ represents concatenation. As for the threat model, we consider signal-directed attacks that efficiently concentrate their attack budget on the signal in the input. Since the signal direction corresponds to the first component of the data, we define the set of allowed perturbations

as:

$$\mathcal{B}_\epsilon(x) = \{z_1 = x + e_1\beta \mid |\beta| \leq \epsilon\} \tag{4.1}$$

where e_1 is the standard basis vector of the first coordinate.

Network structure For the rest of this section we consider a 2 layer feed-forward ReLU network. In particular, we define $f_\theta(x) : \mathbb{R}^d \to \mathbb{R}^2$ as follows:

$$x \xrightarrow{x+\delta} z_1 \xrightarrow{W_1 z_1 + b_1} \hat{z}_2 \xrightarrow{\text{ReLU}(\cdot)} z_2 \xrightarrow{W_2 z_2 + b_2} \hat{z}_3 \tag{4.2}$$

where $x \in \mathbb{R}^d$, $z_1 \in \mathcal{B}_\epsilon(x)$, W_1 and W_2 are linear operators, and we define the set of network parameters as $\theta = \{W_i, b_i\}_{i=1,2}$.

Constructing the convex outer bound Recall from Section 2.3.1, the adversarial polytope $\mathcal{Z}_\epsilon(x)$ is the set of all final-layer activations attainable by perturbing x with some $\tilde{x} \in \mathcal{B}_\epsilon(x)$:

$$\mathcal{Z}_\epsilon(x) = \{f_\theta(\tilde{x}) : \tilde{x} \in \mathcal{B}_\epsilon(x)\} \tag{4.3}$$

Our approach will be to construct a convex outer bound on this adversarial polytope: if no adversarial example exists in this outer approximation, then we are guaranteed that no adversarial example exists in the original polytope. We relax the ReLU activations $z = \text{ReLU}(\hat{z})$ with their convex envelopes:

$$z \geq 0,\ z \geq \hat{z},\ (u - \ell)z \leq u\hat{z} - u\ell \tag{4.4}$$

where u and ℓ are respectively the pre-activations $\hat{z}$ upper and lower bounds.

First, we address the problem of obtaining the upper and lower bounds u and ℓ for the pre-activations $\hat{z}$. Specifically, the following proposition gives a closed form solution for ℓ and u.

Proposition 4.1 *Consider the neural network f_θ defined in Equation (4.2). Let w_1 be the first column of W_1. Then, for a data point x and perturbation budget ϵ, we have the following element-wise bounds on the pre-activation vector $\hat{z}_2$:*

$$\ell \leq \hat{z}_2 \leq u \tag{4.5}$$

where

$$\ell = W_1 x + b_1 - \epsilon|w_1| \ \text{ and } \ u = W_1 x + b_1 + \epsilon|w_1| \tag{4.6}$$

We defer the proof of Proposition 4.1 to Appendix A.1.

Next, we define the outer bound on the adversarial polytope we get from relaxing ReLU constraints as $\tilde{\mathcal{Z}}_\epsilon(x)$. As we have seen in Section 2.3.1, given a

data point x with known label y, we can formulate the problem of finding an adversarial example with a linear program as follows:

$$\min_{\hat{z}_3} [\hat{z}_3]_y - [\hat{z}_3]_{\bar{y}} = c^\top \hat{z}_3 \qquad \text{s.t. } \hat{z}_3 \in \tilde{\mathcal{Z}}_\epsilon(x) \tag{4.7}$$

where $\bar{y}$ is the binary negation of y. Note that if we solve this linear program and find that the objective is positive, then we know that no input perturbation within the threat model can misclassify the example.

However, solving the linear program in Equation (4.7) for every example in the dataset is intractable. Therefore, we consider the dual formulation and take a feasible solution. In the following theorem, we state the dual problem formulation of the linear program in Equation (4.7).

Theorem 4.2 *The dual of the linear program* (4.7) *can be written as*

$$\begin{array}{ll} \max\limits_{\alpha} & \widetilde{J}_\epsilon\left(x, g_\theta(c, \alpha)\right) \\ s.t. & \alpha_j \in [0,1],\ \forall j \end{array} \tag{4.8}$$

where $\widetilde{J}_\epsilon(x, \nu_1, \nu_2, \nu_3)$ *is equal to*

$$-\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j \in \mathcal{I}} \ell_j [\nu_2]_j^+ - \hat{\nu}_1^\top x - \epsilon \left\| [\hat{\nu}_1]_1 \right\|_1 \tag{4.9}$$

and g_θ *is a one-hidden layer neural network given by the equations*

$$\begin{array}{l} \nu_3 = -c \\ \hat{\nu_2} = W_2^\top \nu_3 \\ [\nu_2]_j = 0,\ j \in \mathcal{I}^- \\ [\nu_2]_j = [\hat{\nu_2}]_j,\ j \in \mathcal{I}^+ \\ [\nu_2]_j = \frac{u_j}{u_j - \ell_j} [\hat{\nu_2}]_j^+ - \alpha_j [\hat{\nu_2}]_j^-,\ j \in \mathcal{I} \\ \hat{\nu_1} = W_1^\top \nu_2 \end{array} \tag{4.10}$$

where $\mathcal{I}^-, \mathcal{I}^+$ *and* $\mathcal{I}$ *denote the sets of activations in the hidden layer where* ℓ *and* u *are both negative, both positive or span zero, respectively.*

We defer the proof of Theorem 4.2 to Appendix A.2.

Most importantly, the theorem states that we can represent the dual problem as a linear back propagation network, which provides a tractable solution for a lower bound on the primal objective. In practice, rather than solving the exact dual problem, we choose a fixed dual feasible solution:

$$\alpha_j = \frac{u_j}{u_j - \ell_j} \tag{4.11}$$

4.2 Optimisation of a one-neuron neural network

In this section, we study a simplified one-neuron neural network. We consider the linearly separable distribution from the previous section and assume that the adversarial attacks concentrate all of their perturbation budget along the signal direction. For a formal definition of the data and threat models we refer the reader to Section 4.1.

One-neuron neural network We consider the hypothesis class to be the set of one-neuron shallow neural networks $f_\theta : \mathbb{R}^d \to \mathbb{R}$, defined by:

$$f_\theta(x) = a\,\mathrm{ReLU}\left(\theta^\top x\right) + b \tag{4.12}$$

where $x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, a \in \mathbb{R}$, $b \in \mathbb{R}$ and the only trainable parameter is θ_1. Note that as our distribution is linearly separable, our hypothesis class includes the ground truth. Further, we focus on the classification setting with binary cross-entropy loss:

$$L(x, y) = y \log\left(\sigma\left(x\right)\right) + (1 - y) \log\left(1 - \sigma\left(x\right)\right) \tag{4.13}$$

where $\sigma\left(\cdot\right)$ is the sigmoid function.

4.2.1 Adversarial training

As we have seen in Section 2.1, the basic idea behind adversarial training is to update the network parameters according to the following rule:

$$\theta \leftarrow \theta - \frac{\eta}{|D|} \sum_{(x,y)\in D} \nabla_\theta \max_{x+\delta\in\mathcal{B}_\epsilon(x)} L(f_\theta(x+\delta), y) \tag{4.14}$$

This is usually done by applying some first-order approximation to the maximisation problem. However, for our simplified network we can analytically compute the gradient.

First of all, note that when L is the binary cross-entropy loss function we can rewrite the maximisation problem as follows:

$$\max_{x+\delta\in\mathcal{B}_\epsilon(x)} L(f_\theta(x+\delta), y) = L\left(\mathrm{sgn}(y) \overbrace{\min_{x+\delta\in\mathcal{B}_\epsilon(x)} \mathrm{sgn}(y) f_\theta(x+\delta)}^{:=J_\epsilon(x,y)},\ y\right) \tag{4.15}$$

In particular, if J_ϵ is strictly positive then no adversarial example exists that fools the network. Further, note that this formulation is closely related to the objective considered in Section 4.1 for the convex outer adversarial polytope. This will be useful when comparing COAP and AT gradients. Below we provide the gradient of the adversarial training objective w.r.t. the network parameters θ.

Proposition 4.3 *Consider the neural network f_θ defined in Equation (4.12) and the threat model $\mathcal{B}_\epsilon$ defined in Equation (4.1). Let L be the binary cross-entropy loss function, as defined in Equation (4.13). Then, we have:*

$$\nabla_{\theta_1} \max_{x+\delta \in \mathcal{B}_\epsilon(x)} L(f_\theta(x+\delta), y)$$
$$= -\operatorname{sgn}(y)\sigma\left(-J_\epsilon(x,y)\right) \begin{cases} a(x_1 - \epsilon \operatorname{sgn}(\theta_1))\mathbf{1}\{\ell > 0\} & \text{if } a\operatorname{sgn}(y) > 0 \\ a(x_1 + \epsilon \operatorname{sgn}(\theta_1))\mathbf{1}\{u > 0\} & \text{if } a\operatorname{sgn}(y) < 0 \end{cases}$$

where $\ell = \theta^\top x - \epsilon|\theta_1|$ and $u = \theta^\top x + \epsilon|\theta_1|$ are respectively lower and upper bounds on the ReLU *inputs.*

We defer the proof of Proposition 4.3 to Appendix A.3.

4.2.2 Convex outer adversarial polytope

We now consider the dual approximation $\widetilde{J}_\epsilon$ to the optimisation problem in Equation (4.15). Note that, for a binary classification problem, we have $c = \operatorname{sgn}(y)$ and the dual objective in Theorem 4.2 becomes:

$$\widetilde{J}_\epsilon\left(x, g_\theta(c, \alpha)\right) = \widetilde{J}_\epsilon\left(x, y\right) \tag{4.16}$$

where we set α to the dual feasible solution and for the sake of clarity we omit the dependence on the network parameters θ.

We are particularly interested in the data points for which $J_\epsilon(x,y) \neq \widetilde{J}_\epsilon(x,y)$, i.e., when the certified and adversarial training objectives differ. Below, we provide a necessary and sufficient condition to have a mismatch between the two objectives.

Proposition 4.4 *Consider the neural network f_θ defined in Equation (4.12) and the threat model $\mathcal{B}_\epsilon$ defined in Equation (4.1). Let L be the binary cross-entropy loss function, as defined in Equation (4.13). Further, we define $\ell = \theta^\top x - \epsilon|\theta_1|$ and $u = \theta^\top x + \epsilon|\theta_1|$ respectively as lower and upper bounds on the* ReLU *inputs.*

Let $\mathcal{I}^\star = \{(x,y) : 0 \in [\ell, u] \wedge a\operatorname{sgn}(y) > 0\}$. Then, for data points in $\mathcal{I}^\star$, we have that AT and COAP gradients differ:

$$\nabla_{\theta_1} J_\epsilon(x,y) \neq \nabla_{\theta_1} \widetilde{J}_\epsilon(x,y) \;\; \forall (x,y) \in \mathcal{I}^\star$$

and COAP gradient is given by:

$$\nabla_{\theta_1} L(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x,y), y)$$
$$= -\frac{a\operatorname{sgn}(y)\sigma\left(-\widetilde{J}_\epsilon(x,y)\right)}{2\epsilon}\left(\frac{\ell}{\|\theta_1\|_1}(x_1 + \epsilon\operatorname{sgn}(\theta_1)) + u\frac{x_1\|\theta_1\|_1 - \theta^\top x \operatorname{sgn}(\theta_1)}{\theta_1^2}\right)$$

Further, for data points that are not in $\mathcal{I}^\star$ we have that AT and COAP gradients are equivalent:

$$\nabla_{\theta_1} J_\epsilon(x,y) = \nabla_{\theta_1} \widetilde{J}_\epsilon(x,y) \;\; \forall (x,y) \notin \mathcal{I}^\star$$

We defer the proof of Proposition 4.4 to Appendix A.4.

4.3 Signal-directed approximations hurt generalisation

In this section, we prove that convex relaxations along the signal direction hurt robust generalisation. Further, we conduct extensive experiments on synthetic data to corroborate our theoretical result.

First, in Lemma 4.5 we relate the robust error of the classifier f_θ to the signal parameter θ_1. Specifically, we show that robust error monotonically decreases in θ_1.

Lemma 4.5 *Let f_θ be the neural network defined in Equation (4.12) and $\mathcal{B}_\epsilon$ the threat model defined in Equation (4.1). We define the robust risk $\mathbf{R}_\epsilon$ of f_θ as follows:*

$$\mathbf{R}_\epsilon(\theta) := \mathbb{P}_{(x,y)}\left[\exists z \in \mathcal{B}_\epsilon(x) : y \neq \operatorname{sgn}\left(f_\theta(z)\right)\right] \tag{4.17}$$

Then, $\mathbf{R}_\epsilon(\theta)$ is monotonically decreasing in θ_1.

We defer the proof of Lemma 4.5 to Appendix A.5.

One gradient step training We now investigate the early phase of neural network optimisation. Under structural assumptions on the data, it has been proved that one gradient step with sufficiently large learning rate can drastically decrease the training loss [Chatterji et al., 2021] and extract task-relevant features [Frei et al., 2022, Daniely and Malach, 2020]. A similar setting was also studied recently in Ba et al. [2022] for the MSE loss in the high-dimensional asymptotic limit.

Below we present our main result. Theorem 4.6 relies on two main assumptions. The first is an assumption on the data dimensionality and the initialisation of the network parameters θ. For instance, if the network parameters are initialised sampling from a standard multivariate gaussian $\theta \sim \mathcal{N}(0, I_d)$, then we know that $\|\theta\|_2$ concentrates around $\sqrt{d}$ with high probability. Hence, the assumption is satisfied when the data dimensionality d is sufficiently high. Further, the second assumption requires that the perturbation budget ϵ is sufficiently close to the separation margin γ. This is consistent with the experimental evidence we presented so far, as the generalisation of certified defences significantly worsen for large perturbation budgets.

Theorem 4.6 *Let $\bar{\theta}$ and $\tilde{\theta}$ be the network parameters after one step of gradient descent with respect to AT and COAP objectives. Let,*

$$\frac{\|\theta_{2:d}\|_2}{\|\theta_1\|_2} > \sqrt{\frac{24\gamma^3}{\sigma^2}} \quad \textit{and} \quad \frac{2}{3}\gamma < \epsilon < \gamma \tag{4.18}$$

where θ are the network parameters at initialization. Then, COAP yields higher robust risk than AT:

$$\mathbf{R}_\epsilon(\tilde{\theta}) > \mathbf{R}_\epsilon(\bar{\theta}) \tag{4.19}$$

Below we present the basic idea behind the proof and refer the read to Appendix A.6 for a complete proof of Theorem 4.6.

Proof Since we are only training the signal parameter θ_1, after one gradient descent step, we have:

$$\left\|\bar{\theta}_{2:d}\right\|_2 = \left\|\tilde{\theta}_{2:d}\right\|_2 \tag{4.20}$$

Further, from Lemma 4.5 we know that AT yields smaller robust risk than COAP if the following holds:

$$\left\|\bar{\theta}_1\right\|_2 > \left\|\tilde{\theta}_1\right\|_2 \implies \mathbf{R}_\epsilon(\tilde{\theta}) > \mathbf{R}_\epsilon(\bar{\theta}) \tag{4.21}$$

which, after one step of gradient descent, is equivalent to:

$$\mathbb{E}_{(x,y)}\left[\nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) J_\epsilon(x,y)\right), y\right)\right] < \mathbb{E}_{(x,y)}\left[\nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x,y)\right), y\right)\right]$$

Now recall from Propositions 4.3 and 4.4 that the gradients of AT and COAP differ only on the set $\mathcal{I}^\star$. In particular, we have that:

$$(x,y) \notin \mathcal{I}^\star \implies \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) J_\epsilon(x,y)\right), y\right) = \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x,y)\right), y\right) < 0$$

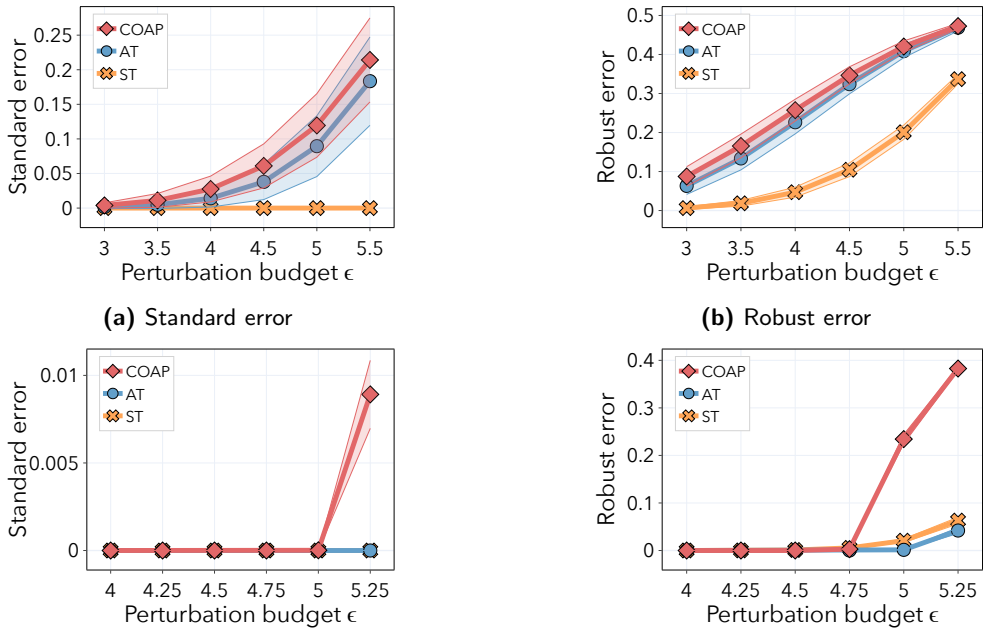
and

$$(x,y) \in \mathcal{I}^\star \implies 0 = \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) J_\epsilon(x,y)\right), y\right) \neq \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x,y)\right), y\right)$$

Hence, for our purpose we need to show that:

$$\mathbb{E}_{(x,y)}\left[\nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x,y)\right), y\right) \mid (x,y) \in \mathcal{I}^\star\right] > 0 \tag{4.22}$$

We defer the proof of this last step, which deals with some technical details, to Appendix A.6. □



(a) Standard error **(b)** Robust error

(c) Standard error **(d)** Robust error

Figure 4.1: We report mean and standard deviation over 15 seeds. In Figure 4.1a and 4.1b we plot respectively the standard and robust errors in the small sample size regime ($n = 50$) for standard training (ST), adversarial training (AT) and convex outer adversarial polytope (COAP) as the perturbation budget ϵ increases. In Figure 4.1c and 4.1d we plot respectively the standard and robust errors in the large sample size regime ($n = 10000$) for standard training (ST), adversarial training (AT) and convex outer adversarial polytope (COAP) as the perturbation budget ϵ increases. See Appendix C.3 for complete experimental details.

Synthetic experiments We corroborate our theory with experimental evidence using a one-hidden layer neural network with 100 neurons. In particular, we investigate the effect of perturbation budget ϵ on generalisation for three different models: standard training (ST), adversarial training (AT) [Madry et al., 2018, Goodfellow et al., 2015] and convex outer adversarial polytope (COAP) [Wong and Kolter, 2018, Wong et al., 2018]. In Figure 4.1, we plot robust and standard errors for both small and large sample size regimes as the perturbation budget ϵ increases. The generalisation gap in the small sample size regime between standard and adversarial training was already observed in Clarysse et al. [2022] for linear classifiers. Here, we observe a further generalisation gap between AT and COAP in both small and large sample size regimes, which surprisingly worsens in the large sample regime.

Chapter 5

Concluding thoughts

This thesis investigates the practical effectiveness of certified defences based on convex relaxations. In particular, we show that certified defences can hurt accuracy, robustness and fairness for realistic datasets and adversarial perturbations.

Our leading hypothesis is that certified defences hurt generalisation because of (i) the large number of relaxed non-convex constraints and (ii) the strong alignment between the adversarial perturbations and the signal direction. We corroborate these hypotheses with extensive experiments on synthetic datasets.

Further, we investigate hypothesis (ii) from a theoretical perspective. In particular, we extend the formulation of empirical and certified defences to a new threat model consisting of perturbations that are perfectly aligned with the signal direction. In this setting, we prove, for a simplified network in high-dimensions, that certified defences yield higher robust risk than empirical defences.

The insights presented so far motivate future research. First, note that this thesis focuses primarily on generalisation rather than fairness. Future research could investigate the fairness of certified defences more deeply, both theoretically and experimentally. Second, the empirical evidence supporting hypotheses (i) and (ii) is restricted to synthetic datasets. Future research could validate these hypotheses on real-world image datasets.

In conclusion, we believe that shedding light on the performance gap between empirical and certified defences will not only provide us with a clearer picture of the trade-offs observed in practice but also lead to better approaches for adversarial robustness.

Appendix A

Deferred proofs

A.1 Proof of Proposition 4.1

Proposition A.1 *Consider the neural network f_θ defined in Equation (4.2). Let w_1 be the first column of W_1. Then, for a data point x and perturbation budget ϵ, we have the following element-wise bounds on the pre-activation vector $\hat{z}_2$:*

$$\ell \leq \hat{z}_2 \leq u \tag{4.5}$$

where

$$\ell = W_1 x + b_1 - \epsilon |w_1| \;\; \textit{and} \;\; u = W_1 x + b_1 + \epsilon |w_1| \tag{4.6}$$

Proof Given a data point x and perturbation budget ϵ, let $\tilde{x} = x + \delta$ be the perturbed input to the network. First, we find an upper bound the pre-activations values $\hat{z}_2$:

$$\hat{z}_2 = W_1(x + \delta) + b_1 = W_1 x + b_1 + W_1 \delta \tag{A.1}$$

In particular, we want to solve the following optimisation problem for each component of the pre-activation vector:

$$u_i = \max_{\tilde{x} \in \mathcal{B}_\epsilon(x)} [\hat{z}_2]_i = [W_1 x]_i + [b_1]_i + \max_{\tilde{x} \in \mathcal{B}_\epsilon(x)} [W_1 \delta]_i \tag{A.2}$$

where u will be the vector containing element-wise upper bounds. Note that $\delta = \beta e_1$, thus the optimisation problem can be rewritten as:

$$\max_{\tilde{x} \in \mathcal{B}_\epsilon(x)} [W_1 \delta]_i = \max_{\|\beta\|_1 \leq \epsilon} \beta \cdot [w_1]_i = \epsilon \cdot \| [w_1]_i \|_1 \tag{A.3}$$

where w_1 is the first column of W_1. The vector of upper bounds will then be:

$$u = W_1 x + b_1 + \epsilon |w_1| \tag{A.4}$$

Along the same lines, we can derive the vector of lower bounds ℓ:

$$l = W_1 x + b_1 - \epsilon |w_1| \tag{A.5}$$

□

A.2 Proof of Theorem 4.2

Theorem 4.2 *The dual of the linear program (4.7) can be written as*

$$\begin{array}{ll} \max\limits_{\alpha} & \widetilde{J}_\epsilon\left(x, g_\theta(c,\alpha)\right) \\ s.t. & \alpha_j \in [0,1],\ \forall j \end{array} \tag{4.8}$$

where $\widetilde{J}_\epsilon(x, \nu_1, \nu_2, \nu_3)$ *is equal to*

$$-\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j\in\mathcal{I}} \ell_j [\nu_2]_j^+ - \hat{\nu}_1^\top x - \epsilon \left\| [\hat{\nu}_1]_1 \right\|_1 \tag{4.9}$$

and g_θ *is a one-hidden layer neural network given by the equations*

$$\begin{array}{l} \nu_3 = -c \\ \hat{\nu_2} = W_2^\top \nu_3 \\ [\nu_2]_j = 0,\ j \in \mathcal{I}^- \\ [\nu_2]_j = [\hat{\nu_2}]_j,\ j \in \mathcal{I}^+ \\ [\nu_2]_j = \frac{u_j}{u_j - \ell_j}[\hat{\nu_2}]_j^+ - \alpha_j [\hat{\nu_2}]_j^-,\ j \in \mathcal{I} \\ \hat{\nu_1} = W_1^\top \nu_2 \end{array} \tag{4.10}$$

where $\mathcal{I}^-, \mathcal{I}^+$ *and* $\mathcal{I}$ *denote the sets of activations in the hidden layer where* ℓ *and* u *are both negative, both positive or span zero, respectively.*

Proof Given a data point x, let $\tilde{x} = x + \delta$ be its perturbation. First, we explicit all the constraints for the linear program defined in (4.7):

$$\begin{array}{l} \min\limits_{\hat{z}_3} [\hat{z}_3]_y - [\hat{z}_3]_{\bar{y}} = c^\top \hat{z}_3\ , \quad \text{s.t.} \\ x + \delta \in \mathcal{B}_\epsilon(x) \\ z_1 = x + \delta \\ \hat{z}_2 = W_1 z_1 + b_1 \\ \hat{z}_3 = W_2 z_2 + b_2 \\ [z_2]_j = 0,\ \forall j \in \mathcal{I}^- \\ [z_2]_j = [\hat{z}_2]_j,\ \forall j \in \mathcal{I}^+ \\ [z_2]_j \geq 0,\ \forall j \in \mathcal{I} \\ [z_2]_j \geq [\hat{z}_2]_j,\ \forall j \in \mathcal{I} \\ \left((u_j - \ell_j)\,[z_2]_j\ - u_j [\hat{z}_2]_j\right) \leq -u_j \ell_j,\ \forall j \in \mathcal{I} \end{array} \tag{A.6}$$

where $\mathcal{I}^-, \mathcal{I}^+$ and $\mathcal{I}$ denote the sets of activations in the hidden layer where ℓ and u are both negative, both positive, or span zero respectively. In order to compute the dual of this problem, we associate the following Lagrangian

variables with each of the constraints:

$$\begin{aligned} \hat{z}_2 = W_1 z_1 + b_1 &\Rightarrow \nu_2 \\ \hat{z}_3 = W_2 z_2 + b_2 &\Rightarrow \nu_3 \\ z_1 = x + \delta &\Rightarrow \psi \\ -[z_2]_j \leq 0 &\Rightarrow \mu_j,\ \forall j \in \mathcal{I} \\ [\hat{z}_2]_j - [z_2]_j \leq 0 &\Rightarrow \tau_j,\ \forall j \in \mathcal{I} \\ \left((u_j - \ell_j)\,[z_2]_j - u_j[\hat{z}_2]_j\right) \leq -u_j\ell_j &\Rightarrow \lambda_j,\ \forall j \in \mathcal{I} \end{aligned} \tag{A.7}$$

note that we do not define explicit dual variables for $[z_2]_j = 0$ and $[z_2]_j = [\hat{z}_2]_j$ as we can easily eliminate them. We write the Lagrangian as follows:

$$\begin{aligned} \mathcal{L}(z, \hat{z}, \nu, \delta, \lambda, \tau, \mu, \psi) = &- \left(W_1^\top \nu_2 + \psi\right)^\top z_1 - \sum_{j \in \mathcal{I}} \left(\mu_j + \tau_j - \lambda_j\,(u_j - \ell_j) + \left[W_2^\top \nu_3\right]_j\right) [z_2]_j \\ &+ \sum_{j \in \mathcal{I}} \left(\tau_j - \lambda_j u_j + [\nu_2]_j\right) [\hat{z}_2]_j + (c + \nu_3)^\top \hat{z}_3 - \sum_{i=1}^{2} \nu_{i+1}^\top b_i \\ &+ \sum_{j \in \mathcal{I}} \lambda_j u_j \ell_j + \psi^\top x + \psi^\top \delta + \sum_{j \in \mathcal{I}^-} [\hat{z}_2]_j [\nu_2]_j \\ &+ \sum_{j \in \mathcal{I}^+} [z_2]_j \left([\nu_2]_j - [W_2^\top \nu_3]_j\right) \\ \text{s.t. } &\tilde{x} \in \mathcal{B}_\epsilon(x) \end{aligned} \tag{A.8}$$

and we take the infimum w.r.t. $z, \hat{z}, \delta$:

$$\begin{aligned} \inf_{z, \hat{z}, \delta} \mathcal{L}(z, \hat{z}, \nu, \delta, \lambda, \tau, \mu, \psi) = &- \inf_{z_2} \sum_{j \in \mathcal{I}} \left(\mu_j + \tau_j - \lambda_j\,(u_j - \ell_j) + \left[W_2^\top \nu_3\right]_j\right) [z_2]_j \\ &+ \inf_{\hat{z}_2} \sum_{j \in \mathcal{I}} \left(\tau_j - \lambda_j u_j + [\nu_2]_j\right) [\hat{z}_2]_j + \inf_{\hat{z}_3} (c + \nu_3)^\top z_3 - \sum_{i=1}^{2} \nu_{i+1}^\top b_i \\ &+ \sum_{j \in \mathcal{I}} \lambda_j u_j \ell_j + \psi^\top x + \inf_{\tilde{x} \in \mathcal{B}_\epsilon(x)} \psi^\top \delta - \inf_{z_1} \left(W_1^\top \nu_2 + \psi\right)^\top z_1 \\ &+ \inf_{\hat{z}_2} \sum_{j \in \mathcal{I}^-} [\hat{z}_2]_j [\nu_2]_j + \inf_{z_2} \sum_{j \in \mathcal{I}^+} [z_2]_j \left([\nu_2]_j - [W_2^\top \nu_3]_j\right) \end{aligned} \tag{A.9}$$

Now, we can compute the infimum for the $\psi^\top \delta$ term:

$$\inf_{\tilde{x} \in \mathcal{B}_\epsilon(x)} \psi^\top \delta = \inf_{\|\beta\|_1 \leq \epsilon} \psi_1 \cdot \beta = -\epsilon \cdot \|\psi_1\|_1 \tag{A.10}$$

and since for all the other terms the infimum of a linear function is $-\infty$, except in the special case when it is identically zero, the infimum of $\mathcal{L}(\cdot)$

becomes:

$$\inf_{z,\hat{z},\delta} \mathcal{L}(.) = \begin{cases} -\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j\in\mathcal{I}} \lambda_j u_j \ell_j + \psi^\top x - \epsilon \left\|\psi_1\right\|_1 & \text{if conditions} \\ -\infty & \text{else} \end{cases} \tag{A.11}$$

where the conditions to satisfy are:

$$\begin{aligned} &\nu_3 = -c \\ &W_1^\top \nu_2 = -\psi \\ &[\nu_2]_j = 0, j \in \mathcal{I}_i^- \\ &[\nu_2]_j = \left[W_2^\top \nu_3\right]_j, j \in \mathcal{I}_i^+ \\ &\left.\begin{aligned} \left(u_j - \ell_j\right)\lambda_j - \mu_j - \tau_j &= \left[W_2^\top \nu_3\right]_j \\ [\nu_2]_j &= u_j\lambda_j - \tau_j \end{aligned}\right\} j \in \mathcal{I} \\ &\lambda, \tau, \mu \geq 0 \end{aligned} \tag{A.12}$$

Thus, we can rewrite the dual problem as follows:

$$\begin{aligned} \max_{\nu,\psi,\lambda,\tau,\mu} \; & -\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j\in\mathcal{I}} \lambda_j u_j \ell_j + \psi^\top x - \epsilon \left\|\psi_1\right\|_1 \\ \text{s.t.} \quad & \nu_3 = -c \\ & W_1^\top \nu_2 = -\psi \\ & [\nu_2]_j = 0, j \in \mathcal{I}_i^- \\ & [\nu_2]_j = \left[W_2^\top \nu_3\right]_j, j \in \mathcal{I}_i^+ \\ & \left.\begin{aligned} \left(u_j - \ell_j\right)\lambda_j - \mu_j - \tau_j &= \left[W_2^\top \nu_3\right]_j \\ [\nu_2]_j &= u_j\lambda_j - \tau_j \end{aligned}\right\} j \in \mathcal{I} \\ & \lambda, \tau, \mu \geq 0 \end{aligned} \tag{A.13}$$

Note that the dual variable λ corresponds to the upper bounds in the convex ReLU relaxation, while μ and τ correspond to the lower bounds. By the complementarity property, we know that at the optimal solution, these variables will be zero if the ReLU constraint is non-tight, or non-zero if the ReLU constraint is tight. Since the upper and lower bounds cannot be tight simultaneously, either λ or $\mu + \tau$ must be zero. This means that at the optimal solution to the dual problem we can decompose $[W_2^\top \nu_3]_j$ into positive and negative parts since $(u_j - \ell_j)\lambda_j \geq 0$ and $\tau_j + \mu_j \geq 0$:

$$\begin{aligned} (u_j - \ell_j)\lambda_j &= [W_2^\top \nu_3]_j^+ \\ \tau_j + \mu_j &= [W_2^\top \nu_3]_j^- \end{aligned} \tag{A.14}$$

combining this with the constraint $[\nu_2]_j = u_j\lambda_j - \tau_j$ leads to

$$[\nu_2]_j = \frac{u_j}{u_j - \ell_j}[W_2^\top \nu_3]_j^+ - \alpha_j[W_2^\top \nu_3]_j^- \tag{A.15}$$

for $j \in \mathcal{I}$ and $0 \leq \alpha_j \leq 1$. Hence, we have that:

$$\lambda_j = \frac{u_j}{u_j - \ell_j}[\hat{\nu}_2]_j^+ \tag{A.16}$$

Now, we denote $\hat{\nu}_1 = -\psi$ to make our notation consistent, and putting all of this together the dual objective becomes:

$$\begin{aligned} -\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j\in\mathcal{I}} \lambda_j u_j \ell_j + \psi^\top x - \epsilon \left\|\psi_1\right\|_1 &= -\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j\in\mathcal{I}} \frac{u_j\ell_j}{u_j - \ell_j}[\hat{\nu}_2]_j^+ - \hat{\nu}_1^\top x - \epsilon \left\|[\hat{\nu}_1]_1\right\|_1 \\ &= -\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j\in\mathcal{I}} \ell_j[\nu_2]_j^+ - \hat{\nu}_1^\top x - \epsilon \left\|[\hat{\nu}_1]_1\right\|_1 \end{aligned} \tag{A.17}$$

and the final dual problem:

$$\begin{array}{ll} \max\limits_{\nu,\hat{\nu}} & -\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j\in\mathcal{I}} \ell_j[\nu_2]_j^+ - \hat{\nu}_1^\top x - \epsilon \left\|[\hat{\nu}_1]_1\right\|_1 \\ \text{s.t.} & \nu_3 = -c \\ & \hat{\nu}_2 = W_2^\top \nu_3 \\ & [\nu_2]_j = 0,\ j \in \mathcal{I}^- \\ & [\nu_2]_j = [\hat{\nu}_2]_j,\ j \in \mathcal{I}^+ \\ & [\nu_2]_j = \frac{u_j}{u_j - \ell_j}[\hat{\nu}_2]_j^+ - \alpha_j[\hat{\nu}_2]_j^-,\ j \in \mathcal{I} \\ & \hat{\nu}_1 = W_1^\top \nu_2 \end{array} \tag{A.18}$$

□

A.3 Proof of Proposition 4.3

Proposition A.2 *Consider the neural network f_θ defined in Equation (4.12) and the threat model $\mathcal{B}_\epsilon$ defined in Equation (4.1). Let L be the binary cross-entropy loss function, as defined in Equation (4.13). Then, we have:*

$$\begin{aligned} &\nabla_{\theta_1} \max_{x+\delta\in\mathcal{B}_\epsilon(x)} L(f_\theta(x+\delta), y) \\ &= -\operatorname{sgn}(y)\sigma\left(-J_\epsilon(x,y)\right) \begin{cases} a(x_1 - \epsilon\operatorname{sgn}(\theta_1))\mathbf{1}\{\ell > 0\} & \text{if } a\operatorname{sgn}(y) > 0 \\ a(x_1 + \epsilon\operatorname{sgn}(\theta_1))\mathbf{1}\{u > 0\} & \text{if } a\operatorname{sgn}(y) < 0 \end{cases} \end{aligned}$$

where $\ell = \theta^\top x - \epsilon|\theta_1|$ and $u = \theta^\top x + \epsilon|\theta_1|$ are respectively lower and upper bounds on the ReLU *inputs.*

Proof Given a data point x with known label $y \in \{-1, 1\}$, when L is the binary cross-entropy loss function we have:

$$\max_{x+\delta\in\mathcal{B}_\epsilon(x)} L(f_\theta(x+\delta), y) = L\left(\operatorname{sgn}(y) \min_{x+\delta\in\mathcal{B}_\epsilon(x)} \operatorname{sgn}(y) f_\theta(x+\delta),\ y\right) \tag{A.19}$$

For our simplified network we can analytically compute a closed form solution of the minimisation problem:

$$\begin{aligned} J_\epsilon &:= \min_{x+\delta\in\mathcal{B}_\epsilon(x)} \operatorname{sgn}(y)\left(b + a\,\mathrm{ReLU}\left(\theta^\top(x+\delta)\right)\right) \\ &= \begin{cases} \operatorname{sgn}(y)\left(b + a\max(0,\ell)\right) & \text{if } a\operatorname{sgn}(y) > 0 \\ \operatorname{sgn}(y)\left(b + a\max(0,u)\right) & \text{if } a\operatorname{sgn}(y) < 0 \end{cases} \\ &= \begin{cases} \operatorname{sgn}(y)\left(b + a\max(0,\ell)\right) & \text{if } a\operatorname{sgn}(y) > 0 \\ \operatorname{sgn}(y)\left(b + a\max(0,u)\right) & \text{if } a\operatorname{sgn}(y) < 0 \end{cases} \end{aligned} \tag{A.20}$$

where $\ell = \theta^\top x - \epsilon|\theta_1|$ and $u = \theta^\top x + \epsilon|\theta_1|$ are respectively lower and upper bounds on the pre-activations. Thus, we can compute the gradients for adversarial training w.r.t the signal parameter:

$$\frac{\partial}{\partial\theta_1} J_\epsilon = \begin{cases} \operatorname{sgn}(y) a (x_1 - \epsilon\operatorname{sgn}(\theta_1)) \mathbf{1}\{\ell > 0\} & \text{if } a\operatorname{sgn}(y) > 0 \\ \operatorname{sgn}(y) a (x_1 + \epsilon\operatorname{sgn}(\theta_1)) \mathbf{1}\{u > 0\} & \text{if } a\operatorname{sgn}(y) < 0 \end{cases} \tag{A.21}$$

and applying the chain-rule we have:

$$\frac{\partial}{\partial\theta_1} L\left(\operatorname{sgn}(y) J_\epsilon, y\right) \tag{A.22}$$

$$= \frac{\partial}{\partial J_\epsilon} L\left(\operatorname{sgn}(y) J_\epsilon, y\right) \cdot \frac{\partial}{\partial\theta_1} J_\epsilon \tag{A.23}$$

$$= \operatorname{sgn}(y)\left[\sigma\left(\operatorname{sgn}(y) J_\epsilon\right) - \mathbf{1}\{y = 1\}\right] \cdot \frac{\partial}{\partial\theta_1} J_\epsilon \tag{A.24}$$

$$= -\operatorname{sgn}(y)\sigma\left(-J_\epsilon\right) \begin{cases} a(x_1 - \epsilon\operatorname{sgn}(\theta_1))\mathbf{1}\{\ell > 0\} & \text{if } a\operatorname{sgn}(y) > 0 \\ a(x_1 + \epsilon\operatorname{sgn}(\theta_1))\mathbf{1}\{u > 0\} & \text{if } a\operatorname{sgn}(y) < 0 \end{cases} \tag{A.25}$$

where in the last equality we use a known property of the sigmoid function, $\sigma(x) = 1 - \sigma(-x)$.

Along the same lines we derive the gradient w.r.t. the non-signal weights ($2 \le k \le d$):

$$\frac{\partial}{\partial\theta_k} J_\epsilon = \begin{cases} \operatorname{sgn}(y) a x_k \mathbf{1}\{\ell > 0\} & \text{if } a\operatorname{sgn}(y) > 0 \\ \operatorname{sgn}(y) a x_k \mathbf{1}\{u > 0\} & \text{if } a\operatorname{sgn}(y) < 0 \end{cases} \tag{A.26}$$

Finally, applying the chain-rule we have:

$$\frac{\partial}{\partial \theta_k} L\left(\operatorname{sgn}(y) J_\epsilon, y\right) \tag{A.27}$$

$$= \frac{\partial}{\partial J_\epsilon} L\left(\operatorname{sgn}(y) J_\epsilon, y\right) \cdot \frac{\partial}{\partial \theta_k} J_\epsilon \tag{A.28}$$

$$= \operatorname{sgn}(y)\left[\sigma\left(\operatorname{sgn}(y) J_\epsilon\right) - \mathbf{1}\{y = 1\}\right] \cdot \frac{\partial}{\partial \theta_k} J_\epsilon \tag{A.29}$$

$$= -\operatorname{sgn}(y)\sigma\left(-J_\epsilon\right) \begin{cases} a x_k \mathbf{1}\{\ell > 0\} & \text{if } a\operatorname{sgn}(y) > 0 \\ a x_k \mathbf{1}\{u > 0\} & \text{if } a\operatorname{sgn}(y) < 0 \end{cases} \tag{A.30}$$

□

A.4 Proof of Proposition 4.4

Proposition A.3 *Consider the neural network f_θ defined in Equation (4.12) and the threat model $\mathcal{B}_\epsilon$ defined in Equation (4.1). Let L be the binary cross-entropy loss function, as defined in Equation (4.13). Further, we define $\ell = \theta^\top x - \epsilon|\theta_1|$ and $u = \theta^\top x + \epsilon|\theta_1|$ respectively as lower and upper bounds on the* ReLU *inputs.*

Let $\mathcal{I}^\star = \{(x, y) : 0 \in [\ell, u] \wedge a\operatorname{sgn}(y) > 0\}$. Then, for data points in $\mathcal{I}^\star$, we have that AT and COAP gradients differ:

$$\nabla_{\theta_1} J_\epsilon(x, y) \neq \nabla_{\theta_1} \widetilde{J}_\epsilon(x, y) \quad \forall (x, y) \in \mathcal{I}^\star$$

and COAP gradient is given by:

$$\nabla_{\theta_1} L(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x, y), y)$$
$$= -\frac{a\operatorname{sgn}(y)\sigma\left(-\widetilde{J}_\epsilon(x, y)\right)}{2\epsilon}\left(\frac{\ell}{\|\theta_1\|_1}(x_1 + \epsilon\operatorname{sgn}(\theta_1)) + u\frac{x_1\|\theta_1\|_1 - \theta^\top x\operatorname{sgn}(\theta_1)}{\theta_1^2}\right)$$

Further, for data points that are not in $\mathcal{I}^\star$ we have that AT and COAP gradients are equivalent:

$$\nabla_{\theta_1} J_\epsilon(x, y) = \nabla_{\theta_1} \widetilde{J}_\epsilon(x, y) \quad \forall (x, y) \notin \mathcal{I}^\star$$

Proof For the sake of clarity, we report here the definition of COAP objective from section 4.1.

$$\widetilde{J}_\epsilon(x, y) = -\sum_{i=1}^{2} \nu_{i+1}^\top b_i + \sum_{j \in \mathcal{I}} \ell_j [\nu_2]_j^+ - \hat{\nu}_1^\top x - \epsilon\left\|[\hat{\nu}_1]_1\right\|_1 \tag{A.31}$$

Further, recall that the dual variables ν are given by the following equations:

$$\begin{aligned}
&\nu_3 = -c \\
&\hat{\nu_2} = W_2^\top \nu_3 \\
&[\nu_2]_j = 0,\ j \in \mathcal{I}^- \\
&[\nu_2]_j = [\hat{\nu_2}]_j,\ j \in \mathcal{I}^+ \\
&[\nu_2]_j = \tfrac{u_j}{u_j - \ell_j}[\hat{\nu_2}]_j^+ - \alpha_j [\hat{\nu_2}]_j^-,\ j \in \mathcal{I} \\
&\hat{\nu_1} = W_1^\top \nu_2
\end{aligned} \tag{A.32}$$

where $\mathcal{I}^-, \mathcal{I}^+$ and $\mathcal{I}$ denote the sets of activations in the hidden layer where ℓ and u are both negative, both positive and span zero, respectively.

First, we consider the case when the neuron is always dead, i.e., $\ell < u < 0$. The dual variables are:

$$\begin{aligned}
\nu_3 &= -\operatorname{sgn}(y) \\
\hat{\nu_2} &= -a \operatorname{sgn}(y) \\
\nu_2 &= 0 \\
\hat{\nu_1} &= 0
\end{aligned} \tag{A.33}$$

Hence, AT and COAP objectives are equal in this case:

$$\widetilde{J_\epsilon} = \operatorname{sgn}(y) b = J_\epsilon \tag{A.34}$$

where the last equality follows from Equation (A.20).

Next, we consider the case when the neuron is always active, i.e., $0 < \ell < u$. The dual variables are:

$$\begin{aligned}
\nu_3 &= -\operatorname{sgn}(y) \\
\hat{\nu_2} &= -a \operatorname{sgn}(y) \\
\nu_2 &= -a \operatorname{sgn}(y) \\
\hat{\nu_1} &= -a \operatorname{sgn}(y) \cdot \theta
\end{aligned} \tag{A.35}$$

and the dual objective becomes:

$$\widetilde{J_\epsilon} = -\nu_3^\top b - \hat{\nu_1}^\top x - \epsilon \left\| [\hat{\nu_1}]_1 \right\|_1 \tag{A.36}$$

$$= \operatorname{sgn}(y) \left(b + a(\theta^\top x) \right) - \epsilon \| a \operatorname{sgn}(y) \theta_1 \| \tag{A.37}$$

$$= \begin{cases} \operatorname{sgn}(y)\,(b + a\ell) & \text{if } a \operatorname{sgn}(y) > 0 \\ \operatorname{sgn}(y)\,(b + au) & \text{if } a \operatorname{sgn}(y) < 0 \end{cases} \tag{A.38}$$

$$= J_\epsilon \tag{A.39}$$

where the last equality follows from the fact that $0 < \ell < u$.

Finally, we consider the case when the neuron is in the activation set $\mathcal{I}$, i.e., $\ell < 0 < u$. The dual variables are:

$$\begin{aligned} \nu_3 &= -\operatorname{sgn}(y) \\ \hat{\nu}_2 &= -a \operatorname{sgn}(y) \\ \nu_2 &= -a \operatorname{sgn}(y) \frac{u}{2\epsilon \|\theta_1\|_1} \\ \hat{\nu}_1 &= -a \operatorname{sgn}(y) \frac{u}{2\epsilon \|\theta_1\|_1} \cdot \theta \end{aligned} \tag{A.40}$$

Here we have two cases, when $\hat{\nu}_2 > 0$ we can rewrite the dual objective as:

$$\widetilde{J}_\epsilon = \operatorname{sgn}(y)\,(b + au) = J_\epsilon \tag{A.41}$$

and the two objectives coincide.

When $\nu_2 < 0$ we can rewrite the dual objective as:

$$\widetilde{J}_\epsilon = \operatorname{sgn}(y) \left(b + \frac{au\ell}{2\epsilon \|\theta_1\|_1} \right) \neq J_\epsilon \tag{A.42}$$

Hence, the only case when COAP gradient differs from AT gradient is when $\nu_2 < 0$ and the neuron belongs to the activation set $\mathcal{I}$.

We compute the partial derivative w.r.t. the signal parameter θ_1 in this case, by the chain rule we have:

$$\frac{\partial}{\partial \theta_1} L\left(\operatorname{sgn}(y) \cdot \widetilde{J}_\epsilon, y\right) \tag{A.43}$$

$$= \frac{\partial}{\partial \widetilde{J}_\epsilon} L\left(\operatorname{sgn}(y) \cdot \widetilde{J}_\epsilon, y\right) \cdot \frac{\partial}{\partial \theta_1} \widetilde{J}_\epsilon \tag{A.44}$$

$$= \operatorname{sgn}(y) \left[\sigma\left(\operatorname{sgn}(y) \cdot \widetilde{J}_\epsilon\right) - \mathbf{1}\{y = 1\} \right] \cdot \frac{\partial}{\partial \theta_1} \widetilde{J}_\epsilon \tag{A.45}$$

$$= -\frac{a \operatorname{sgn}(y) \sigma\left(-\widetilde{J}_\epsilon\right)}{2\epsilon} \left(\frac{\ell}{\|\theta_1\|_1} (x_1 + \epsilon \operatorname{sgn}(\theta_1)) + u \frac{x_1 \|\theta_1\|_1 - \theta^\top x \operatorname{sgn}(\theta_1)}{\theta_1^2} \right) \tag{A.46}$$

and finally we compute the partial derivative w.r.t. the non-signal parameters

θ_k $(2 \leq k \leq d)$:

$$\frac{\partial}{\partial \theta_k} L\left(\operatorname{sgn}(y) \cdot \widetilde{J}_\epsilon, y\right) \tag{A.47}$$

$$= \frac{\partial}{\partial \widetilde{J}_\epsilon} L\left(\operatorname{sgn}(y) \cdot \widetilde{J}_\epsilon, y\right) \cdot \frac{\partial}{\partial \theta_k} \widetilde{J}_\epsilon \tag{A.48}$$

$$= \operatorname{sgn}(y)\left[\sigma\left(\operatorname{sgn}(y) \cdot \widetilde{J}_\epsilon\right) - \mathbf{1}\{y = 1\}\right] \cdot \frac{\partial}{\partial \theta_k} \widetilde{J}_\epsilon \tag{A.49}$$

$$= -\frac{a x_k \operatorname{sgn}(y) \sigma\left(-\widetilde{J}_\epsilon\right) \theta^\top x}{\epsilon \left\|\theta_1\right\|_1} \tag{A.50}$$

□

A.5 Proof of Lemma 4.5

Lemma A.4 *Let f_θ be the neural network defined in Equation (4.12) and $\mathcal{B}_\epsilon$ the threat model defined in Equation (4.1). We define the robust risk $\mathbf{R}_\epsilon$ of f_θ as follows:*

$$\mathbf{R}_\epsilon(\theta) := \mathbb{P}_{(x,y)}\left[\exists z \in \mathcal{B}_\epsilon(x) : y \neq \operatorname{sgn}\left(f_\theta(z)\right)\right] \tag{4.17}$$

Then, $\mathbf{R}_\epsilon(\theta)$ is monotonically decreasing in θ_1.

Proof We assume, without loss of generality, that $\theta_1 > 0$, and since a and b are not trainable parameters we must have $a > 0$ and $b < 0$ to solve the classification problem. The robust risk is then equal to:

$$\begin{aligned}
\mathbf{R}_\epsilon(\theta) &:= \mathbb{P}_{(x,y)}\left[\exists z \in \mathcal{B}_\epsilon(x) : y \neq \operatorname{sgn}\left(f_\theta(z)\right)\right] \\
&= \frac{1}{2}\left(\mathbb{P}_x\left[a \operatorname{ReLU}\left(\theta^\top x\right) + b < 0 \mid y = 1\right] + \mathbb{P}_x\left[a \operatorname{ReLU}\left(\theta^\top x\right) + b > 0 \mid y = -1\right]\right)
\end{aligned}$$

further, we can remove the $\operatorname{ReLU}(\cdot)$ using the fact that $\frac{b}{a} < 0$:

$$\begin{aligned}
\mathbf{R}_\epsilon(\theta) &= \frac{1}{2}\left(\mathbb{P}_x\left[\{x : \theta^\top x + \frac{b}{a} < 0 \vee \theta^\top x < 0\} \mid y = 1\right] + \mathbb{P}_x\left[\{x : \theta^\top x + \frac{b}{a} > 0\} \mid y = -1\right]\right) \\
&= \frac{1}{2}\left(\mathbb{P}_x\left[\theta^\top x + \frac{b}{a} < 0 \mid y = 1\right] + \mathbb{P}_x\left[\theta^\top x + \frac{b}{a} > 0 \mid y = -1\right]\right) \\
&= \frac{1}{2}\left(\mathbb{P}_x\left[\sum_{i=2}^{d} x_i \theta_i < -\theta_1(\gamma - \epsilon) - \frac{b}{a}\right] + \mathbb{P}_x\left[\sum_{i=2}^{d} x_i \theta_i > \theta_1(\gamma - \epsilon) - \frac{b}{a}\right]\right)
\end{aligned}$$

Recall now that for the linearly separable distribution we have:

$$\sum_{i=2}^{d} x_i \theta_i \sim \mathcal{N}\left(0, \sigma^2 \left\|\theta_{2:d}\right\|^2\right)$$

Therefore, we can replace the probabilities with the standard normal CDF Φ:

$$\mathbf{R}_\epsilon(\theta) = \frac{1}{2}\left(\Phi\left(-\frac{(\gamma - \epsilon)\theta_1}{\sigma \left\|\theta_{2:d}\right\|_2} - \frac{b}{a\sigma \left\|\theta_{2:d}\right\|_2}\right) + \Phi\left(-\frac{(\gamma - \epsilon)\theta_1}{\sigma \left\|\theta_{2:d}\right\|_2} + \frac{b}{a\sigma \left\|\theta_{2:d}\right\|_2}\right)\right)$$

hence $\mathbf{R}_\epsilon(\theta)$ is monotonically decreasing in θ_1 and the statement follows. □

A.6 Proof of Theorem 4.6

Theorem 4.6 *Let $\bar{\theta}$ and $\tilde{\theta}$ be the network parameters after one step of gradient descent with respect to AT and COAP objectives. Let,*

$$\frac{\|\theta_{2:d}\|_2}{\|\theta_1\|_2} > \sqrt{\frac{24\gamma^3}{\sigma^2}} \quad \textit{and} \quad \frac{2}{3}\gamma < \epsilon < \gamma \tag{4.18}$$

where θ are the network parameters at initialization. Then, COAP yields higher robust risk than AT:

$$\mathbf{R}_\epsilon(\tilde{\theta}) > \mathbf{R}_\epsilon(\bar{\theta}) \tag{4.19}$$

Proof First we assume, without loss of generality, that at initialisation $\theta_1 > 0$, and since a and b are not trainable parameters we must have $a > 0$ and $b < 0$ to include the ground truth in our hypothesis class.

Let J_ϵ be the adversarial training inner maximisation as defined in Equation (4.15). Then, AT solves the following optimisation problem:

$$\min_\theta \mathbb{E}_{(x,y)} \left[L\left(\sigma\left(\operatorname{sgn}(y) J_\epsilon(x,y)\right), y\right)\right] \tag{A.51}$$

Similarly, let $\widetilde{J}_\epsilon$ be the COAP dual approximation to the inner maximization described in Equation (4.16). Then, COAP solves the following optimisation problem:

$$\min_\theta \mathbb{E}_{(x,y)} \left[L\left(\sigma\left(\operatorname{sgn}(y) \widetilde{J}_\epsilon\right), y\right)\right] \tag{A.52}$$

Since we are only training the signal parameter θ_1, after one gradient descent step, we have:

$$\left\|\bar{\theta}_{2:d}\right\|_2 = \left\|\tilde{\theta}_{2:d}\right\|_2 \tag{A.53}$$

Further, from Lemma 4.5 we know that AT yields smaller robust risk than COAP if the following holds:

$$\left\|\bar{\theta}_1\right\|_2 > \left\|\tilde{\theta}_1\right\|_2 \implies \mathbf{R}_\epsilon(\tilde{\theta}) > \mathbf{R}_\epsilon(\bar{\theta}) \tag{A.54}$$

which, after one step of gradient descent, is equivalent to:

$$\mathbb{E}_{(x,y)} \left[\nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) J_\epsilon(x,y)\right), y\right)\right] < \mathbb{E}_{(x,y)} \left[\nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) \widetilde{J}_\epsilon(x,y)\right), y\right)\right]$$

Now recall from Propositions 4.3 and 4.4 that the gradients of AT and COAP differ only on the set $\mathcal{I}^\star$. In particular, we have that:

$$(x,y) \notin \mathcal{I}^\star \implies \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) J_\epsilon(x,y)\right), y\right) = \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) \widetilde{J}_\epsilon(x,y)\right), y\right) < 0$$

and

$$(x,y) \in \mathcal{I}^\star \implies 0 = \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y) J_\epsilon(x,y)\right), y\right) \neq \nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x,y)\right), y\right)$$

Hence, for our purpose we need to show that:

$$\mathbb{E}_{(x,y)}\left[\nabla_{\theta_1} L\left(\sigma\left(\operatorname{sgn}(y)\widetilde{J}_\epsilon(x,y)\right), y\right) \mid (x,y) \in \mathcal{I}^\star\right] > 0 \tag{A.55}$$

Our strategy will be to lower-bound the expectation in Equation (A.55) with some strictly positive quantity. We define

$$Z = \sum_{i=2}^{d} \theta_i x_i \tag{A.56}$$

and plug-in the gradient computed in Proposition 4.4:

$$\begin{aligned}
&\mathbb{E}_{(x,y)}\left[\nabla_{\theta_1} L\left(\operatorname{sgn}(y)\sigma\left(\widetilde{J}_\epsilon(x,y)\right), y\right) \mid (x,y) \in \mathcal{I}^\star\right] \qquad \text{(A.57)}\\
&= \mathbb{E}_{(x,y)}\left[\frac{a\sigma\left(-\widetilde{J}_\epsilon(x,y)\right)}{2\epsilon}\left(-\frac{\ell}{\theta_1}(\gamma+\epsilon) + u\frac{\sum_{i=2}^{d} x_i\theta_i}{\theta_1^2}\right) \mid (x,y) \in \mathcal{I}^\star\right]\\
&= \frac{a}{2\theta_1\epsilon}\mathbb{E}_{(x,y)}\left[\sigma\left(-\widetilde{J}_\epsilon(x,y)\right)\left(-\ell(\gamma+\epsilon) + u\frac{Z}{\theta_1}\right) \mid (x,y) \in \mathcal{I}^\star\right]\\
&= \frac{a}{2\theta_1\epsilon}\mathbb{E}_{(x,y)}\left[\sigma\left(-\widetilde{J}_\epsilon(x,y)\right) u\frac{Z}{\theta_1} - \sigma\left(-\widetilde{J}_\epsilon(x,y)\right)\ell(\gamma+\epsilon) \mid (x,y) \in \mathcal{I}^\star\right]
\end{aligned}$$

Now, we observe that Z is always negative on the set $\mathcal{I}^\star$, since we need to satisfy the constraint $\ell < 0 < u$:

$$(x,y) \in \mathcal{I}^\star \implies -\theta_1(\gamma+\epsilon) < \sum_{i=2}^{d} \theta_i x_i < -\theta_1(\gamma-\epsilon) < 0 \tag{A.58}$$

Further, from Equation (A.42) we have:

$$(x,y) \in \mathcal{I}^\star \implies \sigma\left(-\widetilde{J}_\epsilon(x,y)\right) \geq \frac{1}{2} \tag{A.59}$$

Combining these two observations we can lower-bound the expectation:

$$\begin{aligned}
&\mathbb{E}_{(x,y)}\left[\nabla_{\theta_1} L\left(\operatorname{sgn}(y)\sigma\left(\widetilde{J}_\epsilon(x,y)\right), y\right) \mid (x,y) \in \mathcal{I}^\star\right] \qquad \text{(A.60)}\\
&= \frac{a}{2\theta_1\epsilon}\mathbb{E}_{(x,y)}\left[\sigma\left(-\widetilde{J}_\epsilon(x,y)\right) u\frac{Z}{\theta_1} - \sigma\left(-\widetilde{J}_\epsilon(x,y)\right)\ell(\gamma+\epsilon) \mid (x,y) \in \mathcal{I}^\star\right]\\
&\geq \frac{a}{2\theta_1\epsilon}\mathbb{E}_{(x,y)}\left[u\frac{Z}{\theta_1} - \frac{\gamma+\epsilon}{2}\ell \mid (x,y) \in \mathcal{I}^\star\right]
\end{aligned}$$

Now, we need to show that this lower-bound is strictly positive:

$$\mathbb{E}_{(x,y)}\left[u\frac{Z}{\theta_1} - \frac{\gamma+\epsilon}{2}\ell \mid (x,y) \in \mathcal{I}^\star\right] > 0 \tag{A.61}$$

Note that, we can further expand this expression:

$$\begin{aligned}&\mathbb{E}_{(x,y)}\left[u\frac{Z}{\theta_1} - \frac{\gamma+\epsilon}{2}\ell \mid (x,y) \in \mathcal{I}^\star\right] \\ &= -(\gamma^2-\epsilon^2)\theta_1^2 + (\gamma+\epsilon)\theta_1\mathbb{E}\left[Z \mid (x,y) \in \mathcal{I}^\star\right] + 2\mathbb{E}\left[Z^2 \mid (x,y) \in \mathcal{I}^\star\right]\end{aligned} \tag{A.62}$$

Further, $Z \mid (x,y) \in \mathcal{I}^\star$ is distributed as a truncated normal with:

$$\alpha = -\frac{\theta_1(\gamma+\epsilon)}{\sigma\left\|\theta_{2:d}\right\|_2} \text{ and } \beta = -\frac{\theta_1(\gamma-\epsilon)}{\sigma\left\|\theta_{2:d}\right\|_2} \tag{A.63}$$

Hence, we can plug in the expectations of the truncated normal distribution to obtain the following:

$$\begin{aligned}&-(\gamma^2-\epsilon^2)\theta_1^2 + \theta_1(\gamma+\epsilon)\mathbb{E}\left[Z \mid (x,y) \in \mathcal{I}^\star\right] + 2\mathbb{E}\left[Z^2 \mid (x,y) \in \mathcal{I}^\star\right] \\ &= -(\gamma^2-\epsilon^2)\theta_1^2 + 2\sigma^2\left\|\theta_{2:d}\right\|_2^2 + \sigma\left\|\theta_{2:d}\right\|_2\theta_1\frac{(\gamma-3\epsilon)\phi(\beta) - (\gamma+\epsilon)\phi(\alpha)}{\Phi(\beta)-\Phi(\alpha)} \\ &\propto -(\gamma^2-\epsilon^2) + 2\sigma^2r^2 + \sigma r\frac{(\gamma-3\epsilon)\phi(\beta) - (\gamma+\epsilon)\phi(\alpha)}{\Phi(\beta)-\Phi(\alpha)} \\ &= -f(r)\end{aligned} \tag{A.64}$$

where we define $r = \frac{\left\|\theta_{2:d}\right\|_2}{\left\|\theta_1\right\|_2}$. Now, under our assumptions, from Lemma B.3 we have:

$$f(r) < 0,\ \forall r > \sqrt{\frac{24\gamma^3}{\sigma^2}} \tag{A.65}$$

which concludes the proof. □

Appendix B

Auxiliary lemmas

B.1 Upper bound on the exponential function

Lemma B.1 *Let $f : \mathbb{R} \to \mathbb{R}$ be the function defined by $f(x) = \exp(x)$. When $x \leq 0$ and n is even we have:*

$$f(x) \leq 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} \tag{B.1}$$

Proof Let $g : (-\infty, 0] \to \mathbb{R}$ be the function defined by

$$g(x) = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} - \exp(x) \tag{B.2}$$

□

Since $g(x) \to \infty$ as $x \to -\infty$, g must attain an absolute minimum somewhere on the interval $(-\infty, 0]$. Now, differentiating we have:

- If g has an absolute minimum at 0 , then for all $x, g(x) \geq g(0) = 1 - \exp(0) = 0$, so we are done.
- If g has an absolute minimum at y for some $y < 0$, then $g'(y) = 0$. But differentiating,

  $$g'(y) = 1 + y + \frac{y^2}{2!} + \cdots + \frac{y^{n-1}}{(n-1)!} - \exp(y) = g(y) - \frac{y^n}{n!}.$$

  Therefore, for any x,

  $$g(x) \geq g(y) = \frac{y^n}{n!} + g'(y) = \frac{y^n}{n!} > 0,$$

  since n is even.

B.2 Lower bound on the difference of Gaussian CDFs

Lemma B.2 *Let $f : \mathbb{R}^2 \to \mathbb{R}$ be the function defined by $f(x, y) = \Phi(y) - \Phi(x)$. When $x < y < 0$ we have:*

$$\phi(0)\left(y - x + \frac{x^3}{6}\right) \leq \Phi(y) - \Phi(x) \tag{B.3}$$

where Φ and ϕ are respectively the CDF and PDF of the standard Gaussian distribution.

Proof First, we want to prove that $\frac{2x}{\sqrt{\pi}}$ is a lower bound for the error function $\operatorname{erf}(x)$ when $x \leq 0$. That is, we want to show that $f(x) \geq 0$ where $f : (-\infty, 0] \to \mathbb{R}$ is the function defined by:

$$f(x) = \operatorname{erf}(x) - \frac{2x}{\sqrt{\pi}} \tag{B.4}$$

Since f is continuous and $f(x) \to \infty$ as $x \to -\infty$, f must attain an absolute minimimum on the interval $(-\infty, 0]$. Now, differentiating we have:

$$f'(x) = \frac{2}{\sqrt{\pi}} \exp(-x^2) - \frac{2}{\sqrt{\pi}} \tag{B.5}$$

hence f attains an absolute minimum at 0 and we have $f(x) \geq f(0) = 0$.
Next, we show that $\frac{2}{\sqrt{\pi}}(x - x^3/3)$ is an upper bound for $\operatorname{erf}(x)$ when $x \leq 0$. Let $g : (-\infty, 0] \to \mathbb{R}$ the function defined by:

$$g(x) = \frac{2}{\sqrt{\pi}}(x - x^3/3) - \operatorname{erf}(x) \tag{B.6}$$

Similarly, since g is continuous and $g(x) \to \infty$ as $x \to -\infty$, g must attain an absolute minimimum on the interval $(-\infty, 0]$. Now, differentiating we have:

$$g'(x) = \frac{2}{\sqrt{\pi}}(1 - x^2 - \exp(-x^2)) \tag{B.7}$$

hence g attains an absolute minimum at 0 and we have $g(x) \geq g(0) = 0$.
Now, since $a < b < 0$ we can use the erf bounds derived above:

$$\Phi(b) - \Phi(a) = \frac{1}{2}\left(\operatorname{erf}(b/\sqrt{2}) - \operatorname{erf}(a/\sqrt{2})\right) \tag{B.8}$$

$$\geq \frac{1}{\sqrt{\pi}}\left(\frac{b}{\sqrt{2}} - \frac{a}{\sqrt{2}} + \frac{a^3}{6\sqrt{2}}\right) \tag{B.9}$$

$$= \phi(0)\left(b - a + \frac{a^3}{6}\right) \tag{B.10}$$

which concludes the proof. □

B.3 Upper bound on the ratio of Gaussian PDFs and CDFs

Lemma B.3 *Suppose $f : \mathbb{R} \to \mathbb{R}$ is defined as follows:*

$$f(r) = \gamma^2 - \epsilon^2 - 2\sigma^2 r^2 - \sigma r \frac{(\gamma - 3\epsilon)\phi(\beta) - (\gamma + \epsilon)\phi(\alpha)}{\Phi(\beta) - \Phi(\alpha)} \tag{B.11}$$

where $\alpha := -\frac{\gamma+\epsilon}{r\sigma}$, $\beta := -\frac{\gamma-\epsilon}{r\sigma}$, Φ and ϕ are respectively the standard Gaussian CDF and PDF.

Assume that:

$$\frac{5 + 2\sqrt{3}}{13}\gamma < \epsilon < \gamma \tag{B.12}$$

Then, we have:

$$f(r) < 0, \quad \forall r > \sqrt{\frac{24\gamma^3}{\sigma^2}} \tag{B.13}$$

Proof We begin by providing a lower bound on the difference of gaussian cdfs. Applying Lemma B.2 with $x = \alpha$ and $y = \beta$ we have:

$$\Phi(\beta) - \Phi(\alpha) \geq \left(\frac{2\epsilon}{r\sigma} - \frac{(\gamma + \epsilon)^3}{6\sigma^3 r^3}\right)\phi(0), \quad \alpha < \beta < 0 \tag{B.14}$$

Next, we can upper-bound f:

$$f(r) \leq \gamma^2 - \epsilon^2 - 2\sigma^2 r^2 - \sigma r \frac{(\gamma - 3\epsilon)\phi(\beta) - (\gamma + \epsilon)\phi(\alpha)}{\left(\frac{2\epsilon}{\sigma r} - \frac{(\gamma+\epsilon)^3}{6\sigma^3 r^3}\right)\phi(0)} \tag{B.15}$$

$$\leq \gamma^2 - \epsilon^2 - 2\sigma^2 r^2 - \sigma^2 r^2 \frac{(\gamma - 3\epsilon)\phi(0) - (\gamma + \epsilon)\phi(\alpha)}{\left(2\epsilon - \frac{(\gamma+\epsilon)^3}{6r^2\sigma^2}\right)\phi(0)} \tag{B.16}$$

$$= \gamma^2 - \epsilon^2 - 2\sigma^2 r^2 - \sigma^2 r^2 \frac{(\gamma - 3\epsilon) - (\gamma + \epsilon)\exp(-\alpha^2/2)}{2\epsilon - \frac{(\gamma+\epsilon)^3}{6\sigma^2 r^2}} \tag{B.17}$$

Now, we use the upper-bound for the exponential function from Lemma B.1 with $n = 2$:

$$\exp(x) \leq 1 + x - x^2/2, \quad \forall x \leq 0 \tag{B.18}$$

and substituting it back into our upper-bound for f we get:

$$f(r) \leq \gamma^2 - \epsilon^2 - 2\sigma^2 r^2 - \sigma^2 r^2 \frac{(\gamma - 3\epsilon) - (\gamma + \epsilon)(1 - \frac{(\gamma+\epsilon)^2}{2r^2\sigma^2} + \frac{(\gamma+\epsilon)^4}{8r^4\sigma^4})}{2\epsilon - \frac{(\gamma+\epsilon)^3}{6r^2\sigma^2}} \tag{B.19}$$

which can be further simplified:

$$f(r) \leq \gamma^2 - \epsilon^2 - 2\sigma^2 r^2 - \sigma^2 r^2 \frac{(\gamma - 3\epsilon) - (\gamma + \epsilon)(1 - \frac{(\gamma+\epsilon)^2}{2r^2\sigma^2} + \frac{(\gamma+\epsilon)^4}{8r^4\sigma^4})}{2\epsilon - \frac{(\gamma+\epsilon)^3}{6r^2\sigma^2}} \tag{B.20}$$

$$= \frac{(\gamma - 7\epsilon)(\gamma + \epsilon)^4 + 4r^2\sigma^2(\gamma + \epsilon)(\gamma^2 - 10\gamma\epsilon + 13\epsilon^2)}{4(\gamma + \epsilon)^3 - 48r^2\sigma^2\epsilon} \tag{B.21}$$

$$= u(r) \tag{B.22}$$

and we have that for $\epsilon > \frac{5+2\sqrt{3}}{13}\gamma$ and $r > \sqrt{\max\left(\frac{(7\epsilon-\gamma)(\gamma+\epsilon)^4}{4\sigma^2(\gamma^2-10\gamma\epsilon+13\epsilon^2)}, \frac{(\gamma+\epsilon)^3}{12\sigma^2\epsilon}\right)}$ the upper bound is negative, i.e. $u(r) < 0$. Finally, for the sake of clarity we can further simplify the condition on r:

$$r > \sqrt{\frac{24\gamma^3}{\sigma^2}} > \sqrt{\max\left(\frac{(7\epsilon - \gamma)(\gamma + \epsilon)^4}{4\sigma^2(\gamma^2 - 10\gamma\epsilon + 13\epsilon^2)}, \frac{(\gamma + \epsilon)^3}{12\sigma^2\epsilon}\right)} \tag{B.23}$$

which concludes the proof. □

Appendix C

Experimental details

C.1 Real-world experiments with ℓ_2-ball perturbations

C.1.1 Datasets

Below we provide a brief description of the computer vision datasets used throughout Chapter 3.

MNIST The MNIST dataset [LeCun et al., 1998] is a collection of handwritten digits that is commonly used for training machine learning models. The dataset contains $n = 70000$ black and white images with size $28x28$, divided in 10 different classes. The ten different represent handwritten digits from 0 to 9. Further, the dataset is split into a training set, which contains 60000 images, and a test set, which contains 10000 images. In Figure C.1 we show samples from the 10 different classes of the MNIST dataset.

Figure C.1: Samples from the MNIST dataset. Source: Wikipedia

CIFAR-10 The CIFAR-10 dataset is a collection of images that are commonly used to train machine learning models. The dataset contains $n = 60000$ color

images with size 32x32 divided in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset is perfectly balanced, i.e., there are 6,000 images of each class. Further, the dataset is split into a training set, which contains 50000 images, and a test set, which contains 10000 images. In Figure C.2 we show samples from the 10 different classes of the CIFAR-10 dataset.

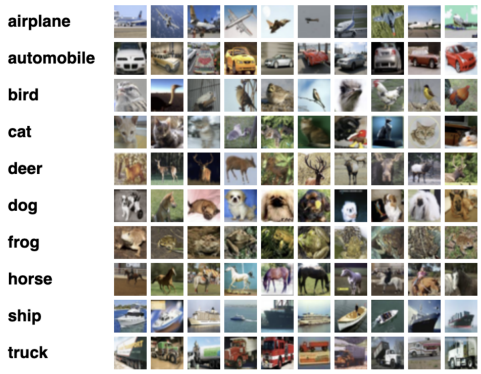


Figure C.2: Samples from the CIFAR-10 dataset. Source: Krizhevsky [2009]

Tiny ImageNet The Tiny ImageNet dataset [Le and Yang, 2015] is a subset of the ImageNet dataset in the famous ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The dataset contains $n = 100000$ color images with size 64×64, divided into 200 classes. The dataset is perfectly balanced, i.e., there are 500 images of each class. Further, the dataset is equally split into a training set, which contains 50000 images, and a test set, which contains 50000 images. In Figure C.3 we show samples from the 200 different classes of the Tiny ImageNet dataset.

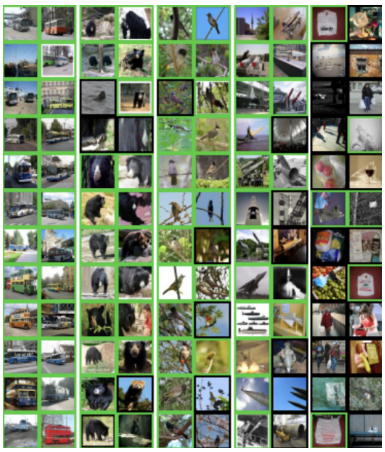


Figure C.3: Samples from the Tiny ImageNet dataset. Source: Chrabaszcz et al. [2017]

C.1.2 Network architectures

Below we provide a brief description of the computer vision architectures used throughout Chapter 3.

Convolutional Neural Network [LeCun et al., 1998] Convolutional Neural Networks (CNNs) are feed-forward neural networks consisting of both fully connected and convolutional layers. The novelty in convolutional layers is that they compute a convolution between the input image and multiple learnable kernels. During the forward pass, we convolve each kernel across the width and height of the input image and compute dot products between the entries of the kernel and the input. We illustrate this simple idea in Figure C.4.

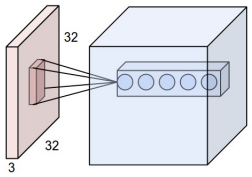


Figure C.4: An example input image in red (e.g. a 32x32x3 CIFAR-10 image), and an example of neurons in the first convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input image. Each neuron is associated with a different filter. Source: https://cs231n.github.io/convolutional-networks/

Residual Network [He et al., 2015] As we increase the depth of CNNs a degradation problem has been exposed: accuracy saturates with increasing network depth and then degrades rapidly. This is due to the optimisation problem becoming much harder for deeper networks. He et al. [2015] propose Residual Networks (ResNet) to learn deeper CNNs without a performance drop. In particular, suppose you want to learn a mapping $h(x)$. Instead of learning the direct mapping, we fit another mapping $f(x) := h(x) - x$ and we then reconstruct the original mapping simply by $h(x) = f(x) + x$. We illustrate this idea in Figure C.5.

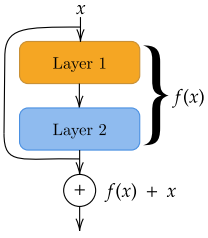


Figure C.5: Illustration of a residual connection. Source: Debenedetti [2022]

C.1.3 Training details

Below we provide complete experimental details to reproduce Figures 3.1 to 3.6.

Model architectures For MNIST, we train the CNN architecture with four convolutional layer and two fully connected layers of 512 units introduced in Wong et al. [2018]. We report the architectural details in Table C.1. For CIFAR-10, we train the residual network (ResNet) with the same structure used in Wong et al. [2018]; we use 1 residual block with 16, 16, 32, and 64 filters. For Tiny ImageNet, we train a WideResNet. Following Xu et al. [2020] we use 3 wide basic blocks with a widen factor of 10.

CNN
CONV 32 $3 \times 3 + 1$
CONV 32 $4 \times 4 + 2$
CONV 64 $3 \times 3 + 1$
CONV 64 $4 \times 4 + 2$
FC 512
FC 512

Table C.1: MNIST model architecture. All layers are followed by $\mathrm{ReLU}(\cdot)$ activations. The last fully connected layer is omitted. "CONV k $w \times h + s$" corresponds to a 2D convolutional layer with k filters of size $w \times h$ using a stride of s in both dimensions. "FC n" corresponds to a fully connected layer with n outputs.

Dataset preprocessing For MNIST, we use full 28×28 images without any augmentations and normalisation. For CIFAR-10, we use random horizontal flips and random crops as data augmentation, and normalise images according to per-channel statistics. For Tiny ImageNet, we use random crops of 56 $\times$ 56 and random flips during training. During testing, we use a central 56 *times* 56 crop. We also normalise images according to per-channel statistics.

Robust evaluation We consider ℓ_2-ball perturbations. We evaluate the robust error using the most expensive version of AutoAttack (AA+) [Croce and Hein, 2020b]. Specifically, we include the following attacks: untargeted APGD-CE (5 restarts), untargeted APGD-DLR (5 restarts), untargeted APGD-DLR (5 restarts), Square Attack (5000 queries), targeted APGD-DLR (9 target classes) and targeted FAB (9 target classes).

AT training details For MNIST, we train 100 epochs using Adam optimiser [Kingma and Ba, 2015] with a learning rate of 0.001, momentum of 0.9 and a batch size of 128; we reduce the learning rate by a factor 0.1 at epochs

40 and 80. For CIFAR-10 with ResNet, we train 150 epochs using SGD with a learning rate of 0.05 and a batch size of 128; we reduce the learning rate by a factor 0.1 at epochs 80 and 120. For Tiny Imagenet and CIFAR-10 with Wide-Resnet we train 200 epochs using SGD with a learning rate of 0.1 and a batch size of 512; we reduce the learning rate by a factor 0.1 at epochs 100 and 150. For the inner optimisation of all models and datasets, adversarial examples are generated with 10 iterations of Auto-PGD [Croce and Hein, 2020b].

COAP training details We follow the settings proposed by the authors and report them here. For MNIST, we use the Adam optimiser [Kingma and Ba, 2015] with a learning rate of 0.001 and a batch size of 50. We schedule ϵ starting from 0.01 to the desired value over the first 20 epochs, after which we decay the learning rate by a factor of 0.5 every 10 epochs for a total of 60 epochs. For CIFAR-10, we use the SGD optimiser with a learning rate of 0.05 and a batch size of 50. We schedule ϵ starting from 0.001 to the desired value over the first 20 epochs, after which we decay the learning rate by a factor of 0.5 every 10 epochs for a total of 60 epochs. For all datasets and models, we use random projection of 50 dimensions. For all experiments, we use the implementation provided in Wong et al. [2018].

CROWN-IBP training details We follow the settings proposed by the authors and report them here. For MNIST, we train 200 epochs with a batch size of 256. We use Adam optimiser [Kingma and Ba, 2015] and set the learning rate to 5×10^{-4}. We warm up with 10 epochs of regular training, and gradually ramp up ϵ_{train} from 0 to ϵ in 50 epochs. We reduce the learning rate by a factor 0.1 at epoch 130 and 190. For CIFAR-10, we train 2000 epochs with a batch size of 256, and a learning rate of 5×10^{-4}. We warm up for 100 epochs, and ramp-up ϵ for 800 epochs. Learning rate is reduced by a factor 0.1 at epoch 1400 and 1700. For Tiny ImageNet, we train 600 epochs with batch size 128. The first 100 epochs are clean training, then we gradually increase ϵ_{train} with a schedule length of 400. For all datasets, an hyper-parameter β to balance LiRPA bounds and IBP bounds for the output layer is gradually decreased from 1 to 0 (1 for only using LiRPA bounds and 0 for only using IBP bounds), with the same schedule of ϵ. For all experiments, we use the implementation provided in the auto LiRPA library [Xu et al., 2020].

C.2 Synthetic experiments with ℓ_2-ball perturbations

Below we provide complete experimental details to reproduce Figures 3.7 to 3.9.

Data generation For the spheres dataset, we generate a random $x \in \mathbb{R}^d$ where $\|x\|_2$ is either R_1 or R_{-1} , with equal probability assigned to each norm. We associate with each x a label y such that $y = -1$ if $\|x\|_2 = R_{-1}$ and $y = 1$ if $\|x\|_2 = R_1$. We can sample uniformly from this distribution by sampling $z \sim \mathcal{N}(0, I_d)$ and then setting $x = \frac{z}{\|z\|_2} R_{-1}$ or $x = \frac{z}{\|z\|_2} R_1$. For the linearly separable distribution we set $d = 1000$, $n = 50$, $n_{\text{test}} = 10^5$, $\gamma = 6$. For the concentric spheres distribution we set $d = 100$, , $n = 50$, $n_{\text{test}} = 10^5$, $\gamma_{\min} = 1$ and $\gamma_{\max} = 12$.

Model and hyper-parameters For all the experiments, we use a MLP architecture with $W = 100$ neurons in each hidden layer and ReLU $(\cdot)$ activation functions. We use PyTorch SGD optimiser with a momentum of 0.95 and train the network for 150 epochs. We sweep over the learning rate $\eta \in \{0.1, 0.01, 0.001\}$ and for each perturbation budget, we choose the one that minimises robust error on the test set and interpolates the training set.

Robust evaluation We consider ℓ_2-ball perturbations. We evaluate robust error at test-time using Auto-PGD [Croce and Hein, 2020b] with 100 iterations and 5 random restarts. We use both the cross-entropy and difference of logits loss to prevent gradient masking. We use the implementation provided in AutoAttack [Croce and Hein, 2020b] with minor adjustments to allow for non-image inputs.

Training paradigms For standard training (ST), we train the network to minimise the cross-entropy loss. For adversarial training (AT) [Madry et al., 2018, Goodfellow et al., 2015] we train the network to minimise the robust cross-entropy loss. At each epoch, we search for adversarial examples using Auto-PGD [Croce and Hein, 2020b] with a budget of 10 steps and 1 random restart. Then, we update the weights using a gradient with respect to the adversarial examples. For convex outer adversarial polytope (COAP) [Wong and Kolter, 2018, Wong et al., 2018]. We train the network to minimise the upper-bound on the robust error. Our implementation is based on the code released by the authors.

C.3 Synthetic experiments with signal-directed perturbations

Below we provide detailed experimental details to reproduce Figure 4.1.

Data generation For the linearly separable distribution we set $d = 1000$, $n_{\text{test}} = 10^5$, $\gamma = 6$.

Model and hyper-parameters For all the experiments, we use the one hidden layer architecture defined in Equation (4.2) with 100 neurons. We use PyTorch SGD optimiser and train all networks for 100 epochs. We sweep over the learning rate $\eta \in \{0.1, 0.01, 0.001\}$ and for each perturbation budget, we choose the one that interpolates the training set and minimises robust error on the test set.

Robust evaluation We perform all the attacks to evaluate robust risk at test-time using exact line search; this is computationally tractable since the attacks are directed along one dimension.

Training paradigms For standard training (ST), wee train the network to minimise the cross-entropy loss. For adversarial training (AT) [Madry et al., 2018, Goodfellow et al., 2015], we train the network to minimise the robust binary cross-entropy loss. At each epoch, we compute an exact adversarial example using line search and update the weights using a gradient with respect to this example. For convex outer adversarial polytope (COAP) [Wong and Kolter, 2018, Wong et al., 2018], at each epoch, we compute upper and lower bounds u and ℓ as described in Proposition 4.1. We then train the network to minimise the upper bound on robust error from Theorem 4.2.

Appendix D

Bonus: Tiramisù recipe

We present an authentic Tiramisù recipe in Algorithm 1. This might even be considered as a contribution of the thesis, as it strives to mitigate the reproducibility crisis of good Italian desserts. Enjoy the Tiramisù.

Algorithm 1: Tiramisù

Ingredients:
ceramic baking pan;
electric mixer ;
350gr mascarpone ;
120gr sugar ;
60g egg yolk ;
90g egg white ;
30g cocoa powder ;
400ml espresso coffee;
200gr savoiardi;

Result: 1 baking pan of Tiramisù (16x22cm)

Recipe:
Prepare 400ml of good espresso and let it cool down in a container ;
Separate the egg whites and yolks into two separate bowls ;
while *whites cream is not white and stiff* **do**
    Whip the egg whites ;
    Gradually add 60 gr of sugar ;
end
Add 60gr of sugar into the yolks bowl ;
while *yolks cream is not yellow and smooth* **do**
    Whip the egg yolks ;
end
Put 350gr of mascarpone into a new bowl ;
while *mascarpone is not smooth and creamy* **do**
    Gently whip the mascarpone ;
end
Add the yolks cream into the mascarpone bowl;
while *mascarpone-yolks cream is not smooth and uniformly mixed* **do**
    Whip mascarpone and yolks;
    (Bonus) Add some Baileys for the "Tiramisù ubriaco" version;
end
Add some whites cream into the mascarpone-yolks bowl;
while *mascarpone-yolks-whites cream is not smooth and uniformly mixed* **do**
    Gently mix with a wooden spoon with bottom-up movements;
    Gradually add more whites cream;
end
while *baking pan is not full* **do**
    Dip the savoiardi in coffee and arrange them to cover the surface of the baking pan;
    Gently spread the cream on the savoiardi until they are fully covered – be generous here;
    Sprinkle with cocoa powder q.b.;
end

Bibliography

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search. In *Proceedings of the European Conference on Computer Vision*, 2020.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the International Conference on Machine Learning*, 2018.

Jimmy Ba, Murat A. Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional Asymptotics of Feature Learning: How One Gradient Step Improves the Representation, 2022. arXiv:2205.01445.

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 *Machine Learning and Knowledge Discovery in Databases - European Conference*, 2013.

Avrim Blum, Travis Dick, Naren Manoj, and Hongyang Zhang. Random Smoothing Might be Unable to Certify L\(ınfty\) Robustness for High-Dimensional Images. *Journal of Machine Learning Research*, 2020.

Nicholas Carlini, Guy Katz, Clark Barrett, and David L. Dill. Provably Minimally-Distorted Adversarial Examples, 2018. arXiv:1709.10207.

Niladri S. Chatterji, Philip M. Long, and Peter L. Bartlett. When Does Gradient Descent with Logistic Loss Find Interpolating Two-Layer Networks? *Journal of Machine Learning Research*, 2021.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum Resilience of Artificial Neural Networks. In *Proceedings of the International Symposium of Automated Technology for Verification and Analysis*, 2017.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets. 2017. arXiv: 1707.08819.

Jacob Clarysse, Julia Hörrmann, and Fanny Yang. Why adversarial training can hurt robust accuracy, 2022. arXiv:2203.02006.

Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of the International Conference on Machine Learning*, 2019.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning*, 2008.

Francesco Croce and Matthias Hein. Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack. In *Proceedings of the International Conference on Machine Learning*, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the International Conference on Machine Learning*, 2020b.

Amit Daniely and Eran Malach. Learning Parities with Neural Networks. In *Advances in Neural Information Processing Systems*, 2020.

Edoardo Debenedetti. Adversarially robust vision transformers. Master's thesis, Swiss Federal Institue of Technology, Lausanne (EPFL), 2022.

John C. Duchi and Hongseok Namkoong. Learning models with uniform performance via distributionally robust optimization. arxiv:1810.08750, 2018.

Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output Range Analysis for Deep Feedforward Neural Networks. In *Proceedings of the International Symposium of NASA Formal Methods*, 2018.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli. A Dual Approach to Scalable Verification of Deep Networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 214–226. ACM, 2012.

Rüdiger Ehlers. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In *Proceedings of the International Symposium of Automated Technology for Verification and Analysis*, 2017.

Ecenaz Erdemir, Jeffrey Bickford, Luca Melis, and Sergül Aydöre. Adversarial Robustness with Non-uniform Perturbations. In *Advances in Neural Information Processing Systems*, 2021.

Samuel G. Finlayson, John D. Bowers, Joichi Ito, Jonathan L. Zittrain, Andrew L. Beam, and Isaac S. Kohane. Adversarial attacks on medical machine learning. *Science*, (6433), March 2019.

Spencer Frei, Niladri S. Chatterji, and Peter L. Bartlett. Random Feature Amplification: Feature Learning and Generalization in Neural Networks, May 2022. arXiv:2202.07626.

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian J. Goodfellow. Adversarial Spheres. 2018. arXiv: 1801.02774.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proceedings of the International Conference on Learning Representations*, 2015.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Arthur Mann, and Pushmeet Kohli. Scalable Verified Training for Provably Robust Image Classification. In *International Conference on Computer Vision*, 2019.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

Úrsula Hébert-Johnson, Michael P. Kim, Omer Reingold, and Guy N. Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *Proceedings of the International Conference on Machine Learning*, 2018.

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 *Proceedings of the International Conference of Computer Aided Verification*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *citeseer*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 2012.

Aounon Kumar, Alexander Levine, Tom Goldstein, and Soheil Feizi. Curse of Dimensionality on Randomized Smoothing for Certifiable Robustness. In *Proceedings of the International Conference on Machine Learning*, 2020.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, (11), 1998.

Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified Adversarial Robustness with Additive Noise. In *Advances in Neural Information Processing Systems*, 2019.

Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward ReLU neural networks. *arXiv:1706.07351*, 2017.

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*, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of the International Conference on Learning Representations*, 2018.

Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *Proceedings of the International Conference on Machine Learning*, 2018.

Jeet Mohapatra, Ching-Yun Ko, Lily Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Hidden Cost of Randomized Smoothing. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2021.

Marius Mosbach, Maksym Andriushchenko, Thomas Alexander Trost, Matthias Hein, and Dietrich Klakow. Logit Pairing Methods Can Fool Gradient-Based Attacks. 2018. arXiv: 1810.12042.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, 2019.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified Defenses against Adversarial Examples. In *Proceedings of the International Conference on Learning Representations*, 2018.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and Mitigating the Tradeoff between Robustness

and Accuracy. In *Proceedings of the International Conference on Machine Learning*, 2020.

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*, 2019.

Amartya Sanyal, Yaxi Hu, and Fanny Yang. How unfair is private learning? In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2022.

Karsten Scheibler, Leonore Winterer, Ralf Wimmer, and Bernd Becker. Towards verification of artificial neural networks. In *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, 2015.

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*, 2019.

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

Vincent Tjeng, Kai Yuanqing Xiao, and Russ Tedrake. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *Proceedings of the International Conference on Learning Representations*, 2019.

Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On Adaptive Attacks to Adversarial Example Defenses. In *Advances in Neural Information Processing Systems*, 2020.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. In *Proceedings of the International Conference on Learning Representations*, 2019.

Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. Simulation-based Adversarial Test Generation for Autonomous Vehicles with Machine Learning Components. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2018.

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 *Proceedings of the International Conference on Machine Learning*, 2018.

Eric Wong and J. Zico Kolter. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proceedings of the International Conference on Machine Learning*, 2018.

Eric Wong, Frank R. Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, 2018.

Han Xu, Xiaorui Liu, Yaxin Li, Anil K. Jain, and Jiliang Tang. To be Robust or to be Fair: Towards Fairness in Adversarial Training. In *Proceedings of the International Conference on Machine Learning*, 2021.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond. In *Advances in Neural Information Processing Systems*, 2020.

Greg Yang, Tony Duan, J. Edward Hu, Hadi Salman, Ilya P. Razenshteyn, and Jerry Li. Randomized Smoothing of All Shapes and Sizes. In *Proceedings of the International Conference on Machine Learning*, pages 10693–10705, 2020.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. In *Proceedings of the International Conference on Machine Learning*, 2019.

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*, 2018.

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

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

CERTIFIED DEFENCES BASED ON CONVEX RELAXATIONS HURT GENERALISATION

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
DE BARTOLOMEIS	PIERSILVIO

With my signature I confirm that

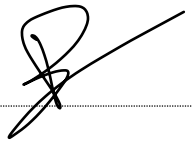
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

28/10/2022

Signature(s)



For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.