Convex Formulation of Robust Two-layer Neural Network Training

Yatong Bai [*1] **Tanmay Gautam** [†2] **Yu Gai** [‡2] **Somayeh Sojoudi** [§1,2]

[1]Department of Mechanical Engineering, University of California, Berkeley, USA
[2]Department of Electrical Engineering and Computer Science, University of California, Berkeley, USA

Abstract

Recent work has shown that the training of a two-layer, scalar-output fully-connected neural network with ReLU activations can be reformulated as a finite-dimensional convex program. Leveraging this result, we derive convex optimization approaches to solve the "adversarial training" problem, which trains neural networks that are robust to adversarial input perturbations. These convex problems are derived for the cases when hinge loss and squared loss between the network output and the target are used to calculate the training cost. Our work provides an alternative adversarial training method over the current approximation methods, such as Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). We demonstrate in different experiments that the proposed method achieves a significantly higher adversarial robustness than existing training methods.

1 INTRODUCTION

Over the past decade, deep learning has become one of the most prominent subfields of machine learning. Neural networks are used in a variety of applications, ranging from natural language processing to computer vision and reinforcement learning [Madry et al., 2018, Goodfellow et al., 2016, Krizhevsky et al., 2012, Mnih et al., 2013]. They are also studied for safety-critical systems, such as autonomous driving [Bojarski et al., 2016]. As neural networks form the backbone of modern-day technology, it is critical to guarantee their safety.

[*]Yatong Bai <yatong_bai@berkeley.edu>
[†]Tanmay Gautam <tgautam23@berkeley.edu>
[‡]Yu Gai <yu_gai@berkeley.edu>
[§]Somayeh Sojoudi <sojoudi@berkeley.edu>

Adversarial robustness has emerged as a powerful framework for evaluating the safety of machine learning models [Madry et al., 2018]. In the field of computer vision, for instance, it has been shown that slight manipulations in the input images can elicit misclassifications in neural networks with high confidence [Szegedy et al., 2014, Moosavi-Dezfooli et al., 2016, Goodfellow et al., 2015]. Thus, adversarial robustness is crucial to safety-critical technologies such as autonomous driving, where highly susceptible models could result in grave consequences [Kurakin et al., 2017].

1.1 RELATED WORK AND OVERVIEW

Recent work has focused on enforcing adversarial robustness on neural networks. While there have been different studies on robustness certification [Anderson et al., 2020, Ma and Sojoudi, 2020], researchers have also been working extensively on training classifiers whose predictions are robust to input perturbations [Kurakin et al., 2017, Goodfellow et al., 2015, Huang et al., 2015]. "Adversarial training" is one of the most effective and reliable methods to train robust classifiers, compared with other methods such as obfuscated gradients [Athalye et al., 2018].

The training of neural networks usually relies on Stochastic Gradient Descent (SGD), which only guarantees convergence to a local minimum. While it has been shown that gradient descent can converge to the optimal point for two-layer ReLU neural networks when they are wide enough [Lacotte and Pilanci, 2020, Du et al., 2019] or when the inputs follow a Gaussian distribution [Brutzkus and Globerson, 2017], spurious local minima can still exist in general applications.

To overcome the issue of arriving at spurious local minima using SGD, the existing works in the literature has considered convexifying the neural network training problem [Bengio et al., 2006, Bach, 2017]. More recently, Pilanci and Ergen [2020] proposed a convex optimization problem with the same global minimum as the non-convex cost func-

tion for a two-layer fully-connected ReLU neural network. While the explicit focus is on the case of squared loss, their analysis extends to arbitrary convex loss functions. Our experiments (Table 1) show that solving the convex problem returns a lower loss than applying SGD to the non-convex cost function, but makes classifiers more susceptible to adversarial input perturbations.

In this paper, we build upon the aforementioned work to develop convex robust optimization problems for the "adversarial training" problem, specifically focusing on the cases of hinge loss and squared loss. We also validate the performance of our proposed algorithms with experiments.

A summary of our contributions is as follows. In section 2, we formulate a robust minimax training problem for a two-layer ReLU network with scalar output trained with any convex loss function. We also discuss the practicality of the aforementioned convex training algorithm proposed in Pilanci and Ergen [2020]. In section 3, we leverage some convexification results to form a minimax problem with a convex objective as an upper-bound on the robust problem. In sections 4 and 5, we study the special cases of the hinge loss and the squared loss. We exploit the structures of these loss functions to arrive at convex optimization formulations for adversarial training. In section 6, we verify the effectiveness of the derived upper-bound problems by comparing numerical test results with their nominal counterparts. We also compare them with conventional gradient-based adversarial training algorithms.

2 PROBLEM FORMULATION

2.1 NOTATIONS

Throughout this work, let $\mathbb{R}^n$ be the set of $n \times 1$ real vectors. We focus on fully-connected neural networks with one ReLU-activated hidden layer and scalar output, defined as

$$\widehat{y} = \sum_{j=1}^{m} (Xu_j)_+ \alpha_j,$$

where

- $X \in \mathbb{R}^{n \times d}$ is the data matrix with n data points in $\mathbb{R}^d$ and $\widehat{y} \in \mathbb{R}^n$ is the vector of predictions (outputs) of the neural network. We denote the target output used for training as $y \in \mathbb{R}^n$.
- $u_1, \ldots, u_m \in \mathbb{R}^d$ are the weight vectors of each of the m neurons in the hidden layer while $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$ are the weights of the output layer.
- $(\cdot)_+ = \max\{0, \cdot\}$ is the ReLU activation function.

Let $\|\cdot\|_p$ denote the ℓ_p-norm within $\mathbb{R}^n$ and $\odot$ denote the Hadamard product. For $P \in \mathbb{N}_+$, we define $[P]$ as the set $\{a \in \mathbb{N}_+ | a \leq P\}$, where $\mathbb{N}_+$ is the set of positive integer numbers. For $q \in \mathbb{R}^n$, $\text{sgn}(q) \in \mathbb{R}^n$ denotes the sign of each entry of q. $[q \geq 0]$ denotes a boolean vector in $\{0,1\}^n$ with ones at the locations of the nonnegative entries of q and zeros otherwise. $\text{Diag}(q)$ denotes a diagonal matrix $Q \in \mathbb{R}^{n \times n}$, where $Q_{ii} = q_i$ for all i, and $Q_{ij} = 0$ for all $i \neq j$. The symbol $\mathbf{1}$ defines a column vector with all entries being 1. For $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, the inequality $a \geq b$ means that $a_i \geq b$ for all $i \in [n]$. $\Pi_{\mathscr{S}}(\cdot)$ denotes the projection onto the set $\mathscr{S}$ and $|\mathscr{S}|$ denotes the cardinality of the set $\mathscr{S}$.

2.2 CONVEX NEURAL-NETWORK TRAINING

We define the problem of training the above neural network with a regularized convex loss function $\ell(\widehat{y}, y)$ as:

$$\min_{(u_j, \alpha_j)_{j=1}^m} \ell\left(\sum_{j=1}^{m} (Xu_j)_+ \alpha_j, y \right) + \frac{\beta}{2} \sum_{j=1}^{m} \left(\|u_j\|_2^2 + \alpha_j^2 \right). \quad (1)$$

where β is a nonnegative regularization parameter. Consider a set of diagonal matrices $\{\text{Diag}([Xu \geq 0]) | u \in \mathbb{R}^d\}$, and let the distinct elements of this set be denoted as $D_1, \ldots, D_P$. The constant P corresponds to the total number of partitions of $\mathbb{R}^d$ by hyperplanes passing through the origin that are also perpendicular to the rows of X [Pilanci and Ergen, 2020]. Intuitively, P can be regarded as the number of possible ReLU activation patterns associated with X.

Consider the convex optimization problem (2):

$$\begin{aligned} &\min_{(v_i, w_i)_{i=1}^P} \ell\left(\sum_{i=1}^{P} D_i X (v_i - w_i), y \right) + \beta \sum_{i=1}^{P} \left(\|v_i\|_2 + \|w_i\|_2 \right) \\ &\text{s.t. } (2D_i - I_n) X v_i \geq 0,\ (2D_i - I_n) X w_i \geq 0,\ \forall i \in [P] \end{aligned} \quad (2)$$

and let $(v_i^\star, w_i^\star)_{i=1}^P$ denote a solution of this problem. Define $m^\star$ as $|\{i : v_i^\star \neq 0\}| + |\{i : w_i^\star \neq 0\}|$. The next theorem explains the relationship between the non-convex training problem (1) and the convex problem (2).

Theorem 1 (Pilanci and Ergen [2020])**.** *Given an arbitrary convex loss function $\ell(\cdot, y)$, the non-convex problem (1) has the same optimal objective as the convex problem (2) provided that $m \geq m^\star$. Moreover, the optimal neural network weights $(u_j^\star, \alpha_j^\star)_{j=1}^m$ can be recovered using the formulas*

$$\begin{aligned} (u_{j_{1i}}^*, \alpha_{j_{1i}}^*) &= \left(\frac{v_i^*}{\sqrt{\|v_i^*\|_2}}, \sqrt{\|v_i^*\|_2} \right) \quad \text{if } v_i^* \neq 0 \\ (u_{j_{2i}}^*, \alpha_{j_{2i}}^*) &= \left(\frac{w_i^*}{\sqrt{\|w_i^*\|_2}}, -\sqrt{\|w_i^*\|_2} \right) \quad \text{if } w_i^* \neq 0. \end{aligned} \quad (3)$$

where the remaining $m - m^\star$ neurons are chosen to have zero weights.

The worst-case computational complexity of solving (2) for the case of squared loss is $\mathscr{O}(d^3 r^3 (\frac{n}{r})^{3r})$ using standard interior-point solvers. Here, r is the rank of the data matrix X and in many cases $r = d$. Thus, the complexity is polynomial in n, an exponential improvement over previous algorithms [Pilanci and Ergen, 2020], but is exponential in r.

2.3 PRACTICAL CONVEX TRAINING

The $\mathcal{O}\big(d^3r^3(\frac{n}{r})^{3r}\big)$ complexity of solving (2) is still prohibitively high for many applications. For example, d and r are equal to $3\times 32\times 32 = 3072$ for the CIFAR-10 and CIFAR-100 image classification datasets. Such high complexity is due to the large number of D matrices, upper-bounded by $2r\big(\frac{e(n-1)}{r}\big)^r$ [Pilanci and Ergen, 2020]. A practical algorithm is to use only a subset of the diagonal matrices, as shown in Alg 1.

Algorithm 1 Practical training

1: **for** $i = 1$ to P_{sample} **do**
2: $\quad a_i \sim \mathcal{N}(0, I_d)$
3: $\quad D_i \leftarrow \text{Diag}([Xa_i \geq 0])$
4: **end for**
5: Select all unique matrices $D_1 \dots D_{P_s}$ from all D_i's.
6: Solve

$$\min_{(v_i,w_i)_{i=1}^{P_s}} \ell\Big(\sum_{i=1}^{P_s} D_i X(v_i - w_i), y\Big) + \beta \sum_{i=1}^{P_s} \big(\|v_i\|_2 + \|w_i\|_2\big)$$
$$\text{s.t. } (2D_i - I_n)Xv_i \geq 0,\ (2D_i - I_n)Xw_i \geq 0,\ \forall i \in [P_s]. \tag{4}$$

7: Recover $u_1, \dots, u_{m_s}$ and $\alpha_1, \dots, \alpha_{m_s}$ from the solution $(v_{s_i}^\star, w_{s_i}^\star)_{i=1}^{P_s}$ of (4) using (3).

Alg 1 is no longer deterministic due to the stochastic sampling of the D matrices, and always yields solutions that upper-bound those of (2). However, we have observed in different experiments that even when the number of D matrices used in the constraints of (2) is significantly smaller than the total number of feasible ReLU activation patterns (P), Alg 1 still returns optimized loss values with low mean and variance. For instance, consider a randomly-generated dataset with $n = 40$ and $d = 2$. The upper bound of the number of ReLU activation patterns is $4\big(\frac{e(39)}{2}\big)^2 = 11239$. We ran Alg 1 to train neural networks using hinge loss with $4, 8, 16, \dots, 2048$ D matrices[1] and compared the optimized loss. We repeated this experiment 15 times for each setting, and plotted the loss in Figure 1. The error bars show the loss values achieved in the best and the worst runs. When there are more than 128 D matrices (much less than the theoretical bound), Alg 1 yields consistent and favorable results. Further increasing the number of D matrices does not produce a significantly lower loss.

2.4 ADVERSARIAL TRAINING

Goodfellow et al. [2015] proposes that a classifier is considered robust against adversarial perturbations if it assigns the same label to all inputs within an ℓ_∞ bound, where the radius ε of the norm ball is small enough to be neglected by other means. Thus, the uncertainty set can be defined as

$$\mathcal{X} = \Big\{ X + \Delta \in \mathbb{R}^{n\times d} \Big| \ \Delta = [\delta_1, \dots, \delta_n]^\top, \delta_k \in \mathbb{R}^d, \|\delta_k\|_\infty \leq \varepsilon, \forall k \in [n] \Big\} \tag{5}$$

As proposed in Madry et al. [2018], one common method for training robust classifiers is to minimize the maximum loss within the perturbation set by solving the following minimax problem:

$$\min_{(u_j,\alpha_j)_{j=1}^m} \begin{pmatrix} \max_{\Delta: X+\Delta\in\mathcal{X}} \ell\Big(\sum_{j=1}^m \big((X+\Delta)u_j\big)_+ \alpha_j, y\Big) \\ +\frac{\beta}{2}\sum_{j=1}^m \big(\|u_j\|_2^2 + \alpha_j^2\big) \end{pmatrix} \tag{6}$$

This process of "training with adversarial data" is often referred to as "adversarial training", as opposed to "standard training" that trains on clean data. In practice, Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) are commonly used to numerically solve the inner maximization of (6) and generate adversarial examples [Madry et al., 2018]. More specifically, FGSM generates adversarial examples using

$$\tilde{x} = x + \varepsilon \cdot \text{sgn}\Big(\nabla_x \ell\big(\sum_{j=1}^m (x^\top u_j)_+ \alpha_j, y\big)\Big). \tag{7}$$

Since FGSM is a one-shot method that assumes linearity, it may miss the worst-case adversarial input. PGD better explores the nonlinear landscape of the problem and is capable of generating "universal" first-order adversaries by running the iterations

$$x^{t+1} = \Pi_{\mathcal{X}}\left(x^t + \gamma \cdot \text{sgn}\Big(\nabla_x \ell\big(\sum_{j=1}^m (x^\top u_j)_+ \alpha_j, y\big)\Big)\right) \tag{8}$$

for $t = 0, 1, \dots$, where x^t is the perturbed data vector at iteration t, x^0 is the clean data x, and $\gamma > 0$ is the step size.

Adversarial training in the prior literature has often been achieved by augmenting the training data with adversarial examples. In this work, we leverage Theorem 1 to recharacterize (6) as robust, convex upper-bound problems that can be minimized globally. We use numerical experiments to verify the improved robustness of the neural networks trained with the proposed convex problems against adversarial examples generated by (7) and (8).

3 CONVEX ADVERSARIAL TRAINING

We first propose the following theorem about adversarial training involving general convex loss functions. The proof is provided in section A.1.

[1] P_{sample} was set to 81920, and the sampling was terminated when enough D matrices were generated. β was chosen as 10^{-4}.

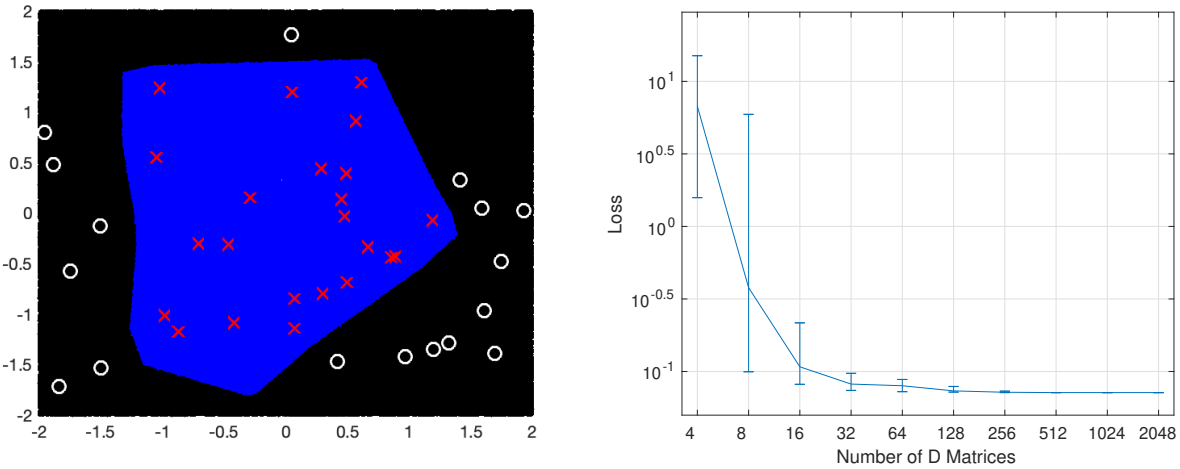

Figure 1: The left figure is a randomized 2-dimensional dataset. The right figure is the optimized training loss for each P_{sample}. The red crosses are positive training points and the white circles are negative training points. The region classified as positive is in blue, whereas the negative region is in black. When P_{sample} reaches 128, the mean and variance of the optimized loss becomes very small.

Consider the optimization problem (9):

$$\min_{(v_i,w_i)_{i=1}^{\widehat{P}}} \begin{pmatrix} \max_{\Delta: X+\Delta\in\mathscr{U}} \ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i), y\right) \\ +\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right) \end{pmatrix} \tag{9a}$$

$$\text{s.t.} \min_{\Delta: X+\Delta\in\mathscr{U}} (2D_i-I_n)(X+\Delta)v_i \geq 0, \ \forall i\in[\widehat{P}] \tag{9b}$$

$$\min_{\Delta: X+\Delta\in\mathscr{U}} (2D_i-I_n)(X+\Delta)w_i \geq 0, \ \forall i\in[\widehat{P}] \tag{9c}$$

where $\mathscr{U}$ is any convex additive perturbation set and $D_1,\ldots,D_{\widehat{P}}$ are all distinct diagonal matrices $\text{Diag}([(X+\Delta)u\geq 0])$ that can be obtained for $\forall u\in\mathbb{R}^d$ and *all* $\Delta: X+\Delta\in\mathscr{U}$.

Theorem 2. *The optimization problem (9) provides an upper-bound on the non-convex adversarial training problem (6), and can be used to train a robust neural network.*

In light of Theorem 2, we use optimization (9) as a surrogate for optimization (6) to train the neural network. We will show that the new problem can be efficiently solved in important cases.

The robust constraints (9b) and (9c) force all points within the perturbation set to be feasible. Intuitively, if $\Delta^\star_{\text{rob}}$ and $(v^\star_{\text{rob}_i}, w^\star_{\text{rob}_i})_{i=1}^{\widehat{P}}$ denote a solution to (9), and $(u^\star_{\text{rob}_j})_{j=1}^{\widehat{m}^\star}$ denote the hidden layer weights of the neural network recovered from $(v^\star_{\text{rob}_i}, w^\star_{\text{rob}_i})_{i=1}^{\widehat{P}}$ using (3), then for every $j\in[\widehat{m}^\star]$, (9b) and (9c) force the ReLU activation pattern $\text{sgn}\left((X+\Delta)u^\star_{\text{rob}_j}\right)$ to stay the same for all Δ such that $X+\Delta\in\mathscr{U}$. Moreover, $\Delta^\star_{\text{rob}}$ attains the maximum loss (worst-case adversary) in the recovered neural network.

Pilanci and Ergen [2020] shows that $P\leq 2r(\frac{e(n-1)}{r})^r$ in (2). Since (9) includes all D matrices in (2), we have $\widehat{P}\geq P$. While $\widehat{P}$ is upper-bounded by 2^n in the worst case, ε is often small, and we thus expect $\widehat{P}$ to be relatively close to P.

Corollary 2.1. *For the perturbation set $\mathscr{X}$, the constraints (9b) and (9c) can be equivalently replaced by*

$$\begin{aligned} (2D_i-I_n)Xv_i &\geq \varepsilon\|v_i\|_1, \quad \forall i\in[\widehat{P}] \\ (2D_i-I_n)Xw_i &\geq \varepsilon\|w_i\|_1, \quad \forall i\in[\widehat{P}] \end{aligned} \tag{10}$$

The proof of the corollary is provided in section A.2. Note that the left-hand side of the inequalities are vectors in $\mathbb{R}^n$ whereas the right-hand side are scalars. Each vector element should be greater than or equal to the scalar.

4 CONVEX HINGE LOSS ADVERSARIAL TRAINING

While the inner maximization of the robust problem (9) is still hard to solve in general, it is tractable for some loss functions. The simplest case is the piecewise-linear hinge loss $\ell(\widehat{y},y)=(1-\widehat{y}\cdot y)_+$. Hinge loss is widely used for classification. Here we focus on binary classification with $y\in\{-1,1\}^n$. [2]

Consider an ℓ_2 regularized two-layer neural network trained with the hinge loss:

$$\min_{(u_j,\alpha_j)_{j=1}^m} \begin{pmatrix} \frac{1}{n}\cdot\mathbf{1}^\top\left(\mathbf{1}-y\odot\sum_{j=1}^m(Xu_j)_+\alpha_j\right)_+ \\ +\frac{\beta}{2}\sum_{j=1}^m\left(\|u_j\|_2^2+\alpha_j^2\right) \end{pmatrix} \tag{11}$$

[2] It is straightforward to extend the analysis in this section to any convex piecewise-affine loss functions. Other ℓ_p norm-bounded additive perturbation sets can be similarly analyzed, as shown in A.6.

The robust formulation against ℓ_∞-bounded adversarial data uncertainty $\mathscr{X}$ is:

$$\min_{(u_j,\alpha_j)_{j=1}^m} \begin{pmatrix} \max_{\Delta: X+\Delta\in\mathscr{X}} \frac{1}{n}\cdot \mathbf{1}^\top \Big(\mathbf{1} - y\odot \sum_{j=1}^m \big((X+\Delta)u_j\big)_+\alpha_j\Big)_+ \\ +\ \frac{\beta}{2}\sum_{j=1}^m \left(\|u_j\|_2^2+\alpha_j^2\right) \end{pmatrix} \tag{12}$$

Applying Theorem 2 and Corollary 2.1 to the case of hinge loss leads to the following optimization as an upper bound on (12).

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} & \begin{pmatrix} \max_{\Delta: X+\Delta\in\mathscr{X}} \frac{1}{n}\cdot \mathbf{1}^\top \Big(\mathbf{1} - y\odot \sum_{i=1}^{\widehat{P}} D_i (X+\Delta)(v_i-w_i)\Big)_+ \\ +\ \beta\sum_{i=1}^{\widehat{P}} \left(\|v_i\|_2+\|w_i\|_2\right) \end{pmatrix} \\ \text{s.t. } & (2D_i-I_n)Xv_i \geq \varepsilon\|v_i\|_1,\quad \forall i\in[\widehat{P}] \\ & (2D_i-I_n)Xw_i \geq \varepsilon\|w_i\|_1,\ \forall i\in[\widehat{P}] \end{aligned} \tag{13}$$

Instead of enumerating an infinite number of points in $\mathscr{X}$, we only need to enumerate all vertices of $\mathscr{X}$, which is finite. This is because the solution $\Delta^\star_{\text{hinge}}$ to the inner maximum always occurs at a vertex of $\mathscr{X}$, as will be shown in Theorem 3. Solving the inner maximization of (13) in closed form leads us to Theorem 3. The proof is provided in section A.3.

Theorem 3. *For the binary classification problem, the inner maximum of (13) is attained at* $\Delta^\star_{hinge} = -\varepsilon\cdot sgn\Big(\sum_{i=1}^{\widehat{P}} D_i y (v_i-w_i)^\top\Big)$*, and the bi-level optimization problem (13) is equivalent to the classic optimization:*

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} & \begin{pmatrix} \frac{1}{n}\sum_{k=1}^n \Big(1 - y_k\sum_{i=1}^{\widehat{P}} d_{ik}x_k^\top(v_i-w_i) + \varepsilon\Big|\Big|\sum_{i=1}^{\widehat{P}} d_{ik}(v_i-w_i)\Big|\Big|_1\Big)_+ \\ +\ \beta\sum_{i=1}^{\widehat{P}} \left(\|v_i\|_2+\|w_i\|_2\right) \end{pmatrix} \\ \text{s.t. } & (2D_i-I_n)Xv_i \geq \varepsilon\|v_i\|_1,\quad \forall i\in[\widehat{P}] \\ & (2D_i-I_n)Xw_i \geq \varepsilon\|w_i\|_1,\ \forall i\in[\widehat{P}] \end{aligned} \tag{14}$$

where d_{ik} is the k^{th} diagonal element of D_i.

The problem (14) is a finite-dimensional convex optimization program that upper-bounds (12), which can be considered as the robust counterpart of (11). We can thus solve it to robustly train the neural network. The neural network weights can be recovered from the solution of (14) via the formulas given in (3). The ℓ_1 norm term in (14) explains the regularization effect of adversarial training.

4.1 APPROXIMATE ALGORITHM FOR CONVEX ADVERSARIAL TRAINING

In terms of algorithm implementation, we use a subset of D matrices similarly to the strategy rendered in Alg 1. For the robust training case, since the D matrices depend on the perturbation Δ, we also add randomness to the data matrix X in the sampling process, which leads to Algorithm 2. P_{sample} and S are preset parameters that determine the number of random weight samples, which upper-bounds the number of D matrices in the optimization problem to be used.

Algorithm 2 Practical adversarial training

for $i=1$ to P_{sample} **do**
  $a_i \sim \mathscr{N}(0, I_d)$
  $D_{i1} \leftarrow \text{Diag}([Xa_i \geq 0])$
  for $j=2$ to S **do**
    $R_{ij} \leftarrow [r_1,\ldots,r_d]$, where $r_h \sim \mathscr{N}(\mathbf{0}, I_n), \forall h\in[d]$
    $\overline{X}_{ij} \leftarrow X + \varepsilon\cdot\text{sgn}(R_{ij})$
    $D_{ij} \leftarrow \text{Diag}([\overline{X}_{ij}a_i \geq 0])$
  end for
end for
Select all unique matrices $D_1,\ldots,D_{P_s}$ from all D_{ij}'s.
Solve

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{P_s}} & \begin{pmatrix} \frac{1}{n}\sum_{k=1}^n \Big(1 - y_k\sum_{i=1}^{P_s} d_{ik}x_k^\top(v_i-w_i) + \varepsilon\Big|\Big|\sum_{i=1}^{P_s} d_{ik}(v_i-w_i)\Big|\Big|_1\Big)_+ \\ +\ \beta\sum_{i=1}^{P_s} \left(\|v_i\|_2+\|w_i\|_2\right) \end{pmatrix} \\ \text{s.t. } & (2D_i-I_n)Xv_i \geq \varepsilon\|v_i\|_1,\quad \forall i\in[P_s] \\ & (2D_i-I_n)Xw_i \geq \varepsilon\|w_i\|_1,\ \forall i\in[P_s] \end{aligned} \tag{15}$$

Recover $u_1,\ldots,u_{m_s}$ and $\alpha_1,\ldots,\alpha_{m_s}$ from the solution $(v^\star_{\text{rob}s_i}, w^\star_{\text{rob}s_i})_{i=1}^{P_s}$ of (15) using (3).

5 CONVEX SQUARED LOSS ADVERSARIAL TRAINING

Squared loss $\ell(\widehat{y},y) = (\widehat{y}-y)^2$ is another commonly used loss function for machine learning that has a relatively simple structure. It is widely used for regression tasks, and can be applied for classification as well.

Consider the non-convex training problem of a two-layer neural network with ReLU activations trained with the ℓ_2-regularized squared loss:

$$\min_{(u_j,\alpha_j)_{j=1}^m} \frac{1}{2}\left|\left|\sum_{j=1}^m (Xu_j)_+\alpha_j - y\right|\right|_2^2 + \frac{\beta}{2}\sum_{j=1}^m \left(\|u_j\|_2^2+\alpha_j^2\right). \tag{16}$$

Coupling this nominal problem with uncertainty set $\mathscr{X}$

allows us to formulate the robust counterpart of (16) as

$$p^\star = \min_{(u_j,\alpha_j)_{j=1}^m} \begin{pmatrix} \max_{\Delta: X+\Delta\in\mathscr{X}} \frac{1}{2}\left\| \sum_{j=1}^m \left((X+\Delta)u_j\right)_+ \alpha_j - y \right\|_2^2 \\ + \frac{\beta}{2}\sum_{j=1}^m \left(\|u_j\|_2^2 + \alpha_j^2\right) \end{pmatrix} \tag{17}$$

By the application of Theorem 2 and Corollary 2.1, it follows that the following adversarial training problem provides an upper bound on $p^\star$:

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} & \begin{pmatrix} \max_{\Delta: X+\Delta\in\mathscr{X}} \frac{1}{2}\left\| \sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i) - y \right\|_2^2 \\ + \beta \sum_{i=1}^{\widehat{P}} \left(\|v_i\|_2 + \|w_i\|_2\right) \end{pmatrix} \\ \text{s.t.} \quad & (2D_i - I_n)Xv_i \geq \varepsilon\|v_i\|_1, \quad \forall i \in [\widehat{P}] \\ & (2D_i - I_n)Xw_i \geq \varepsilon\|w_i\|_1, \ \forall i \in [\widehat{P}]. \end{aligned} \tag{18}$$

5.1 ROBUST SOCP UPPER BOUND PROBLEM

We now exploit the structure of (18) and reformulate it as the following robust second-order cone program (SOCP):

$$\begin{aligned} \min_{(v_i,w_i,b_i,c_i)_{i=1}^{\widehat{P}},a} \quad & a + \beta \sum_{i=1}^{\widehat{P}} (b_i + c_i) \\ \text{s.t.} \ & (2D_i - I_n)Xv_i \geq \varepsilon\|v_i\|_1, \quad \forall i \in [\widehat{P}], \\ & (2D_i - I_n)Xw_i \geq \varepsilon\|w_i\|_1, \ \forall i \in [\widehat{P}], \\ & \|v_i\|_2 \leq b_i, \|w_i\|_2 \leq c_i, \quad \forall i \in [\widehat{P}], \\ & \max_{\Delta: X+\Delta\in\mathscr{X}} \left\| \begin{bmatrix} \sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i) - y \\ 2a - \frac{1}{4} \end{bmatrix} \right\|_2 \leq 2a + \frac{1}{4}, \\ & \qquad\qquad \forall i \in [\widehat{P}]. \end{aligned} \tag{19}$$

Solving the maximization over Δ in closed form leads to the next result.

Theorem 4. *The optimization problem (18) is equivalent to the convex program:*

$$\begin{aligned} \min_{(v_i,w_i,b_i,c_i)_{i=1}^{\widehat{P}},a,z} \quad & a + \beta \sum_{i=1}^{\widehat{P}} (b_i + c_i) \\ \text{s.t.} \ & z_k \geq \left| \sum_{i=1}^{\widehat{P}} D_{ik} x_k^T (v_i - w_i) - y_k \right| \\ & \qquad + \varepsilon \left\| \sum_{i=1}^{\widehat{P}} D_{ik}(v_i - w_i) \right\|_1, \forall k \in [n] \\ & z_{n+1} \geq \left|2a - \tfrac{1}{4}\right|, \ \|z\|_2 \leq 2a + \tfrac{1}{4} \\ & (2D_i - I_n)Xv_i \geq \varepsilon\|v_i\|_1, \quad \forall i \in [\widehat{P}] \\ & (2D_i - I_n)Xw_i \geq \varepsilon\|w_i\|_1, \ \forall i \in [\widehat{P}] \\ & \|v_i\|_2 \leq b_i, \ \|w_i\|_2 \leq c_i, \quad \forall i \in [\widehat{P}] \end{aligned} \tag{20}$$

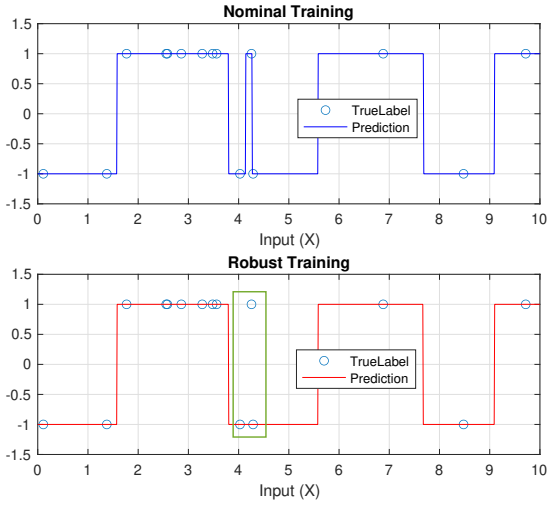

Figure 2: Visualization of binary decision boundaries in 1-dimensional space. The positive point in the highlighted region was considered as an outlier and ignored by Alg 2.

The proof of Theorem 4 can be found in section A.4.

Problem (20) is a convex optimization problem that can be used to train robust neural networks. However, directly using (20) for adversarial training can be intractable due to the large number of constraints that arise when we take into account all D matrices that can be obtained for all Δ such that $X+\Delta \in \mathscr{X}$. To this end, an approximate training algorithm can be used where we sample a subset of the diagonal matrices $D_1, D_2, \ldots, D_{\widehat{P}}$. The practical training algorithm follows the procedure in Alg 2 to approximately solve (20).

6 NUMERICAL EXPERIMENTS

6.1 HINGE LOSS UPPER-BOUND PROBLEM

To visualize the decision boundaries, we ran Alg 1 and Alg 2 on an 1-dimensional binary classification dataset [3], where the training data included 15 randomly generated points. A bias term was included by concatenating a column of ones to the data matrix X. The training labels were also randomly generated such that $y \in \{-1,+1\}$[15]. As shown in Figure 2, in the 1-dimensional space, the adversarial training algorithm acts as a regularizer, ignores points with conflicting perturbation sets, and ensures all points in the perturbation set to be predicted as the same class. The parameters $\varepsilon = 0.03$ and $\beta = 10^{-8}$ were used for this experiment.

Similar experiments were done on 2-dimensional datasets. 34 random points were generated in $[-2,2]\times[-2,2]$. The

[3] Experiments were conducted with CVX [Grant and Boyd, 2014] and CVXPY [Agrawal et al., 2018, Diamond and Boyd, 2016] with the SeDuMi [Sturm, 1999] and MOSEK [ApS, 2019] solvers.

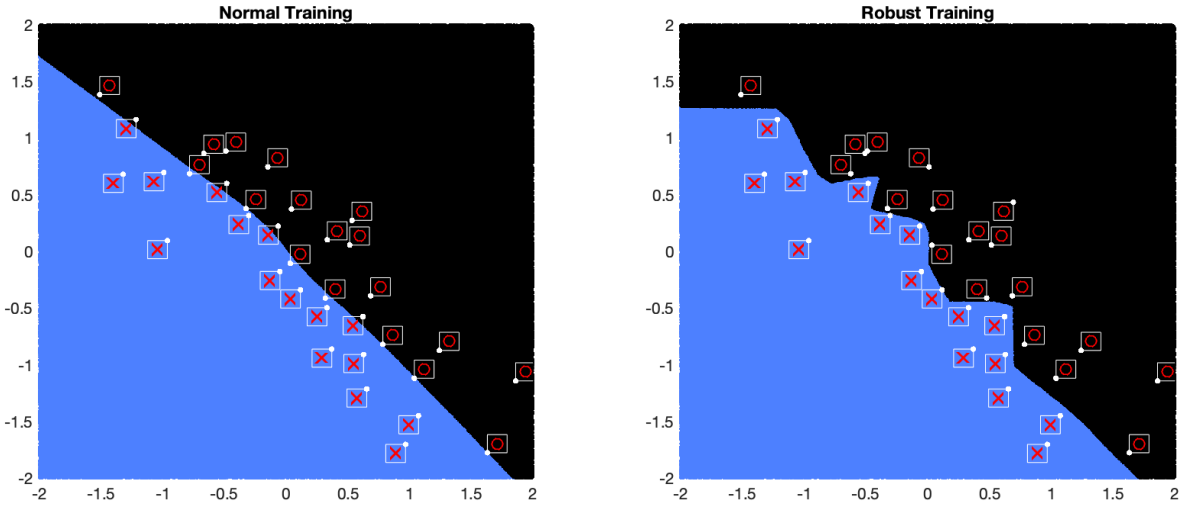

Figure 3: Visualization of binary decision boundaries in 2-dimensional space. The red crosses are positive training points while the red circles are negative points. The region classified as positive is in blue whereas the negative region is in black. The white box around each training data is the ℓ_∞ perturbation bound. The white dot at a vertex of each box is the worst-case perturbation. Our proposed adversarial training method fitted the perturbation boxes while standard training fitted the points.

bias term was again introduced. The algorithm was run with the parameters $P_s = 360$, $\beta = 10^{-9}$ and $\varepsilon = 0.08$. Figure 3 shows the decision boundary, visualising that adversarial training fits the perturbation boxes.

To verify the real-world performance, we compared the performances of Alg 1 and Alg 2 with the "Mammographic Masses" dataset from the UCI Machine Learning Repository [Dua and Graff, 2017]. We removed instances containing NaNs and randomly selected 70% of the data for training and 30% of the data for testing, resulting in $n = 581$ and $d = 5$. Both algorithms were run 15 times with the parameters $P_s = 80$ and $\varepsilon = 0.005$. [4]

The average accuracy was 82.97% for Alg 2 and 79.44% for Alg 1. This result shows that adversarial training with small ε can be used as a regularizer to alleviate overfitting and thereby improve accuracy even on clean test data, as suggested in Kurakin et al. [2017].

We then used FGSM and PGD methods to generate adversarial examples to verify the robustness of the model against adversaries. We also compared Alg 1 and Alg 2 with the widely-used gradient-based method: use PGD to solve the inner maximum of (12) and use gradient descent back propagation to solve the outer minimization. [5]

Hinge loss has a flat part that has zero gradient. To generate adversarial examples even in this part, we treat it as "leaky hinge loss" via the model $\max(\zeta(1-\hat{y}\cdot y), 1-\hat{y}\cdot y)$, where $\zeta \to 0^+$. Hence, the FGSM calculation (7) evaluates to

$$\tilde{x} = x - \varepsilon \cdot \operatorname{sgn}\Big(y \cdot \textstyle\sum_{j:\, x^\top u_j \geq 0} (u_j \alpha_j)\Big).$$

Similarly, the PGD method (8) evaluates to

$$x^{t+1} = \Pi_{\mathscr{X}}\Big(x^t - \gamma \cdot \operatorname{sgn}\big(y \cdot \textstyle\sum_{j:\, x^\top u_j \geq 0} (u_j \alpha_j)\big)\Big).$$

In the following experiments, we use $\gamma = \varepsilon/30$ and run PGD for 40 steps.

We ran the algorithm on the "Mammographic Masses" dataset 20 times. We also selected $\varepsilon = 0.12$ and $P_s = 120$, corresponding to neural network widths of at most 240. The results are shown in Table 1.

Table 1: Average prediction accuracies on clean data, FGSM adversaries, and PGD adversaries. Alg 1 proposed by Pilanci and Ergen [2020] and Alg 2 (this work) corresponds to Cvx-standard and Cvx-robust, respectively.

METHOD	CLEAN	FGSM-ADV.	PGD-ADV.
GD-STANDARD	81.14 %	57.83 %	52.75 %
GD-PGD	81.35 %	75.82 %	74.00 %
CVX-STANDARD	79.80 %	43.41 %	31.14 %
CVX-ROBUST	80.00 %	75.72 %	75.68 %

Table 1 shows that the classifiers trained with Alg 1 are more susceptible to FGSM and PGD adversaries than those trained with classical gradient descent method. Our proposed Alg 2 addressed this issue and significantly improved the accuracy on data with adversaries generated by both

[4] For all experiments with the "Mammographic Masses" and the "MNIST" datasets, β was chosen as 10^{-4} and z-score standardization was performed.

[5] We used a modified version of [Tromgy, 2019] for the implementation of gradient-based algorithms.

Table 2: Average optimized objective and CPU time.

METHOD	OBJECTIVE	CPU TIME (S)
GD-NOMINAL	.4144	4.996
GD-PGD	.6955	143.0
CVX-STANDARD (ALG 1)	.2428	36.92
CVX-ROBUST (ALG 2)	.9663	38.26

Table 3: Average prediction accuracy on clean data and adversarial data over 7 runs using the MNIST database.

METHOD	CLEAN	FGSM-ADV.	PGD-ADV.
CVX-STANDARD	96.47 %	38.83 %	28.20 %
CVX-ROBUST	93.44 %	81.19 %	80.98 %

FGSM and PGD. Alg 2 also outperforms the PGD adversarial training method when predicting PGD adversaries.

We also present the training objective and CPU time in Table 2. Alg 1 returned a lower loss than gradient descent methods. However, such advantage in optimization was not reflected in prediction accuracy. Alg 2 returned a higher objective than the gradient-based adversarial training method, which is expected since Alg 2 solves the upper-bound problem.

Table 2 also shows that PGD adversarial training can be slow due to the iterative process of generating adversarial training examples, and Alg 2 was much faster.

Another set of comparisons between standard and adversarial training was performed on a subset of the MNIST database of handwritten digits for binary classification between digits "2" and "8" [LeCun et al., 2010]. The images were downsampled to 10×10 and thus $d = 100$. The parameters were $n = 3000$, $\varepsilon = \frac{16}{255}$ and $P_s = 36$. The results are provided in Table 3.

6.2 SQUARED LOSS UPPER-BOUND PROBLEM

The performance of upper bound problem (20) was compared with the standard training problem (2) on a contrived 1-dimensional dataset.

Figure 4 shows the true relationship between the 1-dimensional data X and its corresponding label y. Throughout this experiment, training datasets were constructed by uniformly sampling 8 points from this distribution. Test datasets were similarly constructed by uniformly sampling 100 points. A bias term was included by concatenating a column of ones to the training data vector.

The training and testing procedure was repeated for 100 trials on the standard training problem (Alg 1). Figure 5 reports the average test mean square error (MSE) over all trials. For the robust problem, we varied the allowed perturbation $\varepsilon = 0.1, \ldots, 0.9$. The training and testing procedure was carried out for 10 trials for each ε. Figure 5 reports the average test MSE for each ε setting.

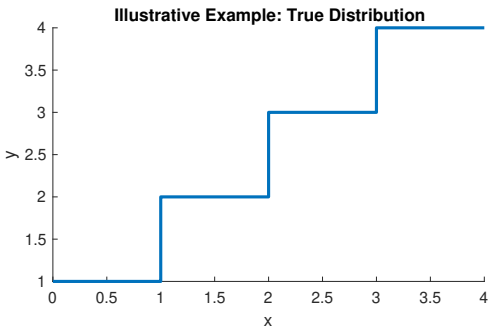

Figure 4: True relationship between data (x) and target (y) used in the illustrative example in section 6.2. Training (with $n = 8$ points) and test (with $n = 100$ points) sets are uniformly sampled from the distribution.

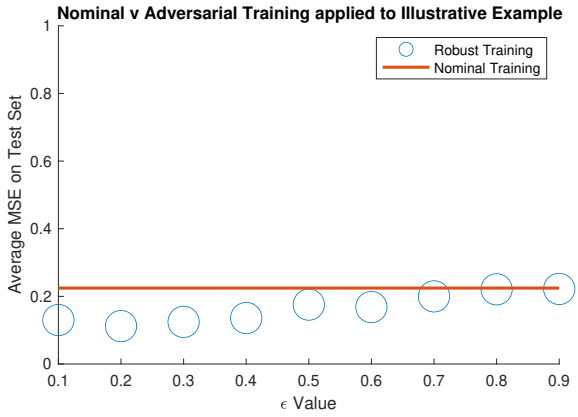

Figure 5: This plot shows that the robust training approach (20) outperforms the standard approach for different $\varepsilon \in \{0.1, ..., 0.9\}$ on the dataset studied in section 6.2.

We note that the robust training procedure outperforms the nominal counterpart for all reported ε values. We further observe that the average MSE is the lowest at $\varepsilon \approx 0.2$, and approaches the MSE of the standard training procedure for smaller or larger ε. This behavior arises as the robust problem attempts to account for all points within the uncertainty interval around the sampled training points. When ε is sufficiently small, the robust problem approaches the standard training problem. Larger values of ε cause the uncertainty interval to overestimate the constant regions of the true distribution of the considered dataset. This, in turn, increases the average MSE.

7 CONCLUDING REMARKS

In this work, we studied the adversarial training of two-layer fully-connected scalar-output ReLU neural networks,

and formulated a robust minimax training problem for such neural networks under arbitrary convex loss functions. We then focused on the specific cases of the hinge and squared loss and developed robust convex optimization formulations. Such robust optimization problems are convex and thus can be easily optimized globally, and their structures explain the regularization effect of adversarial training. Through numerical experiments on various datasets, we have shown that the proposed robust training problems are able to fit the perturbation boxes around the training data points and improve prediction accuracy on test data containing adversaries.

References

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1): 42–60, 2018.

Brendon G. Anderson, Ziye Ma, Jingqi Li, and Somayeh Sojoudi. Tightened convex relaxations for neural network robustness certification. In *59th IEEE Conference on Decision and Control, CDC 2020, Jeju Island, South Korea, December 14-18, 2020*, pages 2190–2197. IEEE, 2020. doi: 10.1109/CDC42340.2020.9303750.

MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

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

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

Yoshua Bengio, Nicolas Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, pages 123–130. MIT Press, 2006.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 605–614. JMLR.org, 2017.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, March 2014.

Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015.

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

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Jonathan Lacotte and Mert Pilanci. All local minima are global for two-layer relu neural networks: The hidden convex optimization landscape. *CoRR*, abs/2006.05900, 2020.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]*, 2, 2010.

Ziye Ma and Somayeh Sojoudi. Strengthened SDP verification of neural network robustness via non-convex cuts. *CoRR*, abs/2010.08603, 2020. URL `https://arxiv.org/abs/2010.08603`.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2574–2582. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.282.

Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7695–7705. PMLR, 13–18 Jul 2020.

J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Tromgy. simple-neural-networks. `https://github.com/tromgy/simple-neural-networks`, 2019.

A APPENDIX

A.1 PROOF OF THEOREM 2

We first reiterate the following result proposed by Pilanci and Ergen [2020] as the tool of the proof.

Lemma 5. *For a given data matrix X and $(v_i, w_i)_{i=1}^P$, if $(2D_i - I_n)Xv_i \geq 0$ and $(2D_i - I_n)Xw_i \geq 0$ for all $i \in [P]$, then we can recover the corresponding neural network weights $(u_{v,w_j}, \alpha_{v,w_j})_{j=1}^{m^\star}$ using the formulas in (3), and*

$$\ell\Big(\sum_{i=1}^{P} D_i X(v_i - w_i), y\Big) + \beta \sum_{i=1}^{P} \big(\|v_i\|_2 + \|w_i\|_2\big) = \ell\Big(\sum_{j=1}^{m^\star} (Xu_{v,w_j})_+ \alpha_{v,w_j}, y\Big) + \frac{\beta}{2} \sum_{j=1}^{m^\star} \big(\|u_{v,w_j}\|_2^2 + \alpha_{v,w_j}^2\big). \tag{21}$$

Theorem 1 implies that (1) has the same objective value as the following finite dimensional convex optimization problem:

$$\begin{aligned} q^\star = \min_{(v_i, w_i)_{i=1}^P} \ell\Big(\sum_{i=1}^{P} D_i X(v_i - w_i), y\Big) + \beta \sum_{i=1}^{P} \big(\|v_i\|_2 + \|w_i\|_2\big) \\ \text{s.t. } (2D_i - I_n)Xv_i \geq 0,\ (2D_i - I_n)Xw_i \geq 0,\ \forall i \in [P] \end{aligned} \tag{22}$$

where $D_1, \ldots, D_P$ are all matrices in the set of matrices $\mathscr{D}$, which is defined as the set of all distinct diagonal matrices $\mathrm{Diag}([Xu \geq 0])$ that can be obtained for all possible $u \in \mathbb{R}^d$. We recall that the optimal neural network weights can be recovered using (3).

Consider the optimization problem (23)

$$\begin{aligned} \widetilde{q}^\star = \min_{(v_i, w_i)_{i=1}^{\widetilde{P}}} \ell\Big(\sum_{i=1}^{\widetilde{P}} D_i X(v_i - w_i), y\Big) + \beta \sum_{i=1}^{\widetilde{P}} \big(\|v_i\|_2 + \|w_i\|_2\big) \\ \text{s.t. } (2D_i - I_n)Xv_i \geq 0,\ (2D_i - I_n)Xw_i \geq 0,\ \forall i \in [\widetilde{P}] \end{aligned} \tag{23}$$

where additional D matrices named $D_{P+1}, \ldots, D_{\widetilde{P}}$, are introduced. These additional matrices are still diagonal, with each entry being either 0 or 1 while they do not belong to $\mathscr{D}$. They represent "infeasible hyperplanes" that cannot be achieved by the sign pattern of Xu for any $u \in \mathbb{R}^d$.

Lemma 6. *$\widetilde{q}^\star = q^\star$. The optimization problem (23) has the same optimal objective as (22).*

The proof of Lemma 6 is given in section A.5.

The robust minimax training problem (6) considers an uncertain data matrix $X + \Delta$. Different $X + \Delta$ within the uncertainty set $\mathscr{U}$ can result in different D matrices. Now, we define $\widehat{\mathscr{D}} = \bigcup_\Delta \mathscr{D}_\Delta$, where $\mathscr{D}_\Delta$ is the set of diagonal matrices for a particular Δ such that $X + \Delta \in \mathscr{U}$. By construction, we have $\mathscr{D}_\Delta \subseteq \widehat{\mathscr{D}}$ for every Δ such that $X + \Delta \in \mathscr{U}$. Thus, if we define $D_1, \ldots, D_{\widehat{P}}$ as all matrices in $\widehat{\mathscr{D}}$, then for every Δ with the property $X + \Delta \in \mathscr{U}$, the optimization problem

$$\begin{aligned} \min_{(v_i, w_i)_{i=1}^{\widehat{P}}} \ell\Big(\sum_{i=1}^{\widehat{P}} D_i (X + \Delta)(v_i - w_i), y\Big) + \beta \sum_{i=1}^{\widehat{P}} (\|v_i\|_2 + \|w_i\|_2) \\ \text{s.t. } (2D_i - I_n)(X + \Delta)v_i \geq 0,\ (2D_i - I_n)(X + \Delta)w_i \geq 0,\ \forall i \in [\widehat{P}] \end{aligned} \tag{24}$$

is equivalent to

$$\min_{(u_j, \alpha_j)_{j=1}^m} \ell\Big(\sum_{j=1}^{m} ((X + \Delta)u_j)_+ \alpha_j, y\Big) + \frac{\beta}{2} \sum_{j=1}^{m} \big(\|u_j\|_2^2 + \alpha_j^2\big)$$

as long as $m \geq \widehat{m}^\star$ with $\widehat{m}^\star = |\{i : v_i^\star(\Delta) \neq 0\}| + |\{i : w_i^\star(\Delta) \neq 0\}|$, where $(v_i^\star(\Delta), w_i^\star(\Delta))_{i=1}^{\widehat{P}}$ denotes an optimal point to (24).

Now, we focus on the minimax training problem with convex objective given by

$$\min_{(v_i, w_i)_{i=1}^{\widehat{P}} \in \mathscr{F}} \left(\begin{aligned} \max_{\Delta : X + \Delta \in \mathscr{U}} \ell\Big(\sum_{i=1}^{\widehat{P}} D_i (X + \Delta)(v_i - w_i), y\Big) + \beta \sum_{i=1}^{\widehat{P}} \big(\|v_i\|_2 + \|w_i\|_2\big) \\ \text{s.t. } (2D_i - I_n)(X + \Delta)v_i \geq 0,\ (2D_i - I_n)(X + \Delta)w_i \geq 0,\ \forall i \in [\widehat{P}] \end{aligned} \right), \tag{25}$$

where $\mathscr{F}$ is defined as

$$\left\{(v_i,w_i)_{i=1}^{\widehat{P}} \;\middle|\; \exists \Delta : X+\Delta \in \mathscr{U} \;\text{ s.t. } (2D_i-I_n)(X+\Delta)v_i \geq 0,\ (2D_i-I_n)(X+\Delta)w_i \geq 0,\ \forall i \in [\widehat{P}]\right\}.$$

The introduction of the feasible set $\mathscr{F}$ is to avoid the situation where the inner maximization over Δ is infeasible and the objective becomes $-\infty$, leaving the outer minimization problem unbounded.

Moreover, consider the following problem:

$$\begin{aligned}\min_{(v_i,w_i)_{i=1}^{\widehat{P}}} \quad & \left(\ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta^\star_{v,w})(v_i-w_i),y\right)+\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right)\right) \\ \text{s.t. } & (2D_i-I_n)(X+\Delta^\star_{v,w})v_i \geq 0,\ (2D_i-I_n)(X+\Delta^\star_{v,w})w_i \geq 0,\ \forall i \in [\widehat{P}]\end{aligned} \tag{26}$$

where $\Delta^\star_{v,w}$ is the optimal point for $\max_{\Delta: X+\Delta\in\mathscr{U}} \ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i),y\right)$. Note that the inequality constraints are dropped for the maximization here compared to (25).

The optimization problem (25) gives a lower-bound on (26). To prove this, we first rewrite (26) as:

$$\min_{(v_i,w_i)_{i=1}^{\widehat{P}}} \; f((v_i,w_i)_{i=1}^{\widehat{P}})$$

$$\text{where } f((v_i,w_i)_{i=1}^{\widehat{P}}) = \begin{cases} \ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta^\star_{v,w})(v_i-w_i),y\right) & (2D_i-I_n)(X+\Delta^\star_{v,w})v_i \geq 0,\ \forall i\in[\widehat{P}] \\ \qquad +\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right), & (2D_i-I_n)(X+\Delta^\star_{v,w})w_i \geq 0,\ \forall i\in[\widehat{P}] \\ +\infty, & \text{otherwise.}\end{cases}$$

Now we analyze (25). Consider two cases:

Case 1: For some $(v_i,w_i)_{i=1}^{\widehat{P}}$, $\Delta^\star_{v,w}$ is optimal for the inner maximization of (25) and the inequality constraints are inactive. This happens whenever $\Delta^\star_{v,w}$ is feasible for the particular choice of $(v_i,w_i)_{i=1}^{\widehat{P}}$. In other words, $(2D_i-I_n)(X+\Delta^\star_{v,w})v_i \geq 0$ and $(2D_i-I_n)(X+\Delta^\star_{v,w})w_i \geq 0$ hold true for all $i\in[\widehat{P}]$. For these $(v_i,w_i)_{i=1}^{\widehat{P}}$, we have:

$$\begin{pmatrix}\max_{\Delta: X+\Delta\in\mathscr{U}} \ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i),y\right)+\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right) \\ \text{s.t. } (2D_i-I_n)(X+\Delta)v_i \geq 0,\ (2D_i-I_n)(X+\Delta)w_i \geq 0,\ \forall i\in[\widehat{P}]\end{pmatrix} = \begin{pmatrix}\ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta^\star_{v,w})(v_i-w_i),y\right) \\ \qquad +\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right)\end{pmatrix}$$

Case 2: For some other $(v_i,w_i)_{i=1}^{\widehat{P}}$, $\Delta^\star_{v,w}$ is infeasible, but some other Δ within the perturbation bound satisfies the inequality constraints. Suppose that among the feasible Δ's,

$$\begin{aligned}\widetilde{\Delta}^\star_{v,w} = \operatorname*{arg\,max}_{\Delta: X+\Delta\in\mathscr{U}} \; & \ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i),y\right)+\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right) \\ \text{s.t. } & (2D_i-I_n)(X+\Delta)v_i \geq 0,\ (2D_i-I_n)(X+\Delta)w_i \geq 0,\ \forall i\in[\widehat{P}].\end{aligned}$$

In this case,

$$\begin{pmatrix}\max_{\Delta: X+\Delta\in\mathscr{U}} \ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i),y\right)+\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right) \\ \text{s.t. } (2D_i-I_n)(X+\Delta)v_i \geq 0,\ (2D_i-I_n)(X+\Delta)w_i \geq 0,\ \forall i\in[\widehat{P}]\end{pmatrix} = \begin{pmatrix}\ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\widetilde{\Delta}^\star_{v,w})(v_i-w_i),y\right) \\ \qquad +\beta\sum_{i=1}^{\widehat{P}}\left(\|v_i\|_2+\|w_i\|_2\right)\end{pmatrix}$$

Case 3: For all other $(v_i,w_i)_{i=1}^{\widehat{P}}$, the objective value is $+\infty$ since they do not belong to $\mathscr{F}$.

Therefore, (25) can be rewritten as

$$\min_{(v_i,w_i)_{i=1}^{\widehat{P}}} g((v_i,w_i)_{i=1}^{\widehat{P}}), \text{ where}$$

$$g((v_i,w_i)_{i=1}^{\widehat{P}}) = \begin{cases} \ell\Big(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta^\star_{v,w})(v_i-w_i), y\Big) & (2D_i-I_n)(X+\Delta^\star_{v,w})v_i \geq 0,\ \forall i \in [\widehat{P}] \\ \quad +\beta\sum_{i=1}^{\widehat{P}}\big(\|v_i\|_2+\|w_i\|_2\big), & (2D_i-I_n)(X+\Delta^\star_{v,w})w_i \geq 0,\ \forall i \in [\widehat{P}] \\ & \exists j : (2D_j-I_n)(X+\Delta^\star_{v,w})v_j < 0 \\ \ell\Big(\sum_{i=1}^{\widehat{P}} D_i(X+\widetilde{\Delta}^\star_{v,w})(v_i-w_i), y\Big) & \text{or } (2D_j-I_n)(X+\Delta^\star_{v,w})w_j < 0 \\ \quad +\beta\sum_{i=1}^{\widehat{P}}\big(\|v_i\|_2+\|w_i\|_2\big), & \exists \Delta :\ (2D_i-I_n)(X+\Delta)v_i \geq 0,\ \forall i \in [\widehat{P}] \\ & \quad (2D_i-I_n)(X+\Delta)w_i \geq 0,\ \forall i \in [\widehat{P}] \\ +\infty, & \text{otherwise} \end{cases}$$

Hence, $g((v_i,w_i)_{i=1}^{\widehat{P}}) = f((v_i,w_i)_{i=1}^{\widehat{P}})$ for all $(v_i,w_i)_{i=1}^{\widehat{P}}$ that belong to the first and the third case. $g((v_i,w_i)_{i=1}^{\widehat{P}}) < f((v_i,w_i)_{i=1}^{\widehat{P}})$ for $(v_i,w_i)_{i=1}^{\widehat{P}}$ belonging to the second case. Thus, $\min_{(v_i,w_i)_{i=1}^{\widehat{P}}} g((v_i,w_i)_{i=1}^{\widehat{P}}) \leq \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} f((v_i,w_i)_{i=1}^{\widehat{P}})$. This concludes that (25) is a lower-bound to (26).

Let $(v^\star_{\text{minimax}_i}, w^\star_{\text{minimax}_i})_{i=1}^{\widehat{P}}$ denote an optimal point for (26). It is possible that for some $\Delta : X+\Delta \in \mathscr{U}$, the constraints $(2D_i-I_n)(X+\Delta)v^\star_{\text{minimax}_i}, \geq 0$ and $(2D_i-I_n)(X+\Delta)w^\star_{\text{minimax}_i}, \geq 0$ are not satisfied for all $i \in [\widehat{P}]$. In light of Lemma 5, at those Δ where such constraints are violated, the convex problem (26) does not reflect the cost of the neural network. For these infeasible Δ, the input-label pairs $(X+\Delta, y)$ can have high cost in the neural network and potentially become the worst-case adversary. However, these Δ are ignored in (26) due to the infeasibility. Since adversarial training aims to minimize the cost over the worst-case adversaries generated upon the training data whereas (26) may sometimes miss the worst-case adversaries, (26) does not fully accomplish the task of adversarial training. In fact, by applying Theorem 1 and Lemma 6, it can be verified that (25) and (26) are lower-bounds to (6) as long as $m \geq \widehat{m}^\star$:

$$\begin{aligned} &\min_{(u_j,\alpha_j)_{j=1}^m} \left(\max_{\Delta : X+\Delta \in \mathscr{U}} \ell\Big(\sum_{j=1}^m \big((X+\Delta)u_j\big)_+ \alpha_j, y \Big) + \frac{\beta}{2}\sum_{j=1}^m \Big(\|u_j\|_2^2 + \alpha_j^2 \Big) \right) \\ \geq &\min_{(u_j,\alpha_j)_{j=1}^m} \ell\Big(\sum_{j=1}^m \big((X+\Delta^\star_{v,w})u_j\big)_+ \alpha_j, y \Big) + \frac{\beta}{2}\sum_{j=1}^m \Big(\|u_j\|_2^2 + \alpha_j^2 \Big) \\ = &\begin{pmatrix} \min\limits_{(v_i,w_i)_{i=1}^{\widehat{P}}} \ell\Big(\sum\limits_{i=1}^{\widehat{P}} D_i(X+\Delta^\star_{v,w})(v_i-w_i), y \Big) + \beta\sum\limits_{i=1}^{\widehat{P}}\big(\|v_i\|_2+\|w_i\|_2\big) \\ \text{s.t. } (2D_i-I_n)(X+\Delta^\star_{v,w})v_i \geq 0,\ (2D_i-I_n)(X+\Delta^\star_{v,w})w_i \geq 0,\ \forall i \in [\widehat{P}] \end{pmatrix}. \end{aligned}$$

To address the feasibility issue, we can apply robust optimization techniques (Boyd and Vandenberghe [2004] section 4.4.2) and replace the constraints in (26) with robust convex constraints, which will lead to (9). Let $\big((v^\star_{\text{rob}_i}, w^\star_{\text{rob}_i})_{i=1}^{\widehat{P}}, \Delta^\star_{\text{rob}}\big)$ denote an optimal point of (9) and let $(u^\star_{\text{rob}_j}, \alpha^\star_{\text{rob}_j})_{j=1}^{\widehat{m}^\star}$ be the neural network weights recovered from $(v^\star_{\text{rob}_i}, w^\star_{\text{rob}_i})_{i=1}^{\widehat{P}}$ with (3). In light of Lemma 5, since the constraints $(2D_i-I_n)(X+\Delta)v^\star_{\text{rob}_i} \geq 0$ and $(2D_i-I_n)(X+\Delta)w^\star_{\text{rob}_i} \geq 0$ for all $i \in [\widehat{P}]$ apply to all $X+\Delta \in \mathscr{U}$, all $X+\Delta \in \mathscr{U}$ satisfy the equality

$$\ell\Big(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v^\star_{\text{rob}_i} - w^\star_{\text{rob}_i}), y \Big) + \beta\sum_{i=1}^{\widehat{P}}\big(\|v^\star_{\text{rob}_i}\|_2 + \|w^\star_{\text{rob}_i}\|_2\big) = \ell\Big(\sum_{j=1}^{\widehat{m}^\star} \big((X+\Delta)u^\star_{\text{rob}_j}\big)_+ \alpha^\star_{\text{rob}_j}, y \Big) + \frac{\beta}{2}\sum_{j=1}^{\widehat{m}^\star}\big(\|u^\star_{\text{rob}_j}\|_2^2 + \alpha^{\star 2}_{\text{rob}_j}\big).$$

Thus, since

$$\Delta^\star_{\text{rob}} = \operatorname*{arg\,max}_{\Delta : X+\Delta \in \mathscr{U}} \ell\Big(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v^\star_{\text{rob}_i} - w^\star_{\text{rob}_i}), y \Big) + \beta\sum_{i=1}^{\widehat{P}}\big(\|v^\star_{\text{rob}_i}\|_2 + \|w^\star_{\text{rob}_i}\|_2\big),$$

we have

$$\Delta^\star_{\text{rob}} = \operatorname*{arg\,max}_{\Delta : X+\Delta \in \mathscr{U}} \ell\Big(\sum_{j=1}^{\widehat{m}^\star} \big((X+\Delta)u^\star_{\text{rob}_j}\big)_+ \alpha^\star_{\text{rob}_j}, y \Big) + \frac{\beta}{2}\sum_{j=1}^{\widehat{m}^\star}\big(\|u^\star_{\text{rob}_j}\|_2^2 + \alpha^{\star 2}_{\text{rob}_j}\big),$$

giving rise:

$$
\begin{aligned}
&\ell\left(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta^\star_{\text{rob}})(v^\star_{\text{rob}_i}-w^\star_{\text{rob}_i}),y\right)+\beta\sum_{i=1}^{\widehat{P}}\left(\|v^\star_{\text{rob}_i}\|_2+\|w^\star_{\text{rob}_i}\|_2\right)\\
=&\ell\left(\sum_{j=1}^{\widehat{m}^\star}\left((X+\Delta^\star_{\text{rob}})u^\star_{\text{rob}_j}\right)_+\alpha^\star_{\text{rob}_j},y\right)+\frac{\beta}{2}\sum_{j=1}^{\widehat{m}^\star}\left(\|u^\star_{\text{rob}_j}\|_2^2+\alpha^{\star 2}_{\text{rob}_j}\right)\\
=&\max_{\Delta:X+\Delta\in\mathscr{U}}\ell\left(\sum_{j=1}^{\widehat{m}^\star}\left((X+\Delta)u^\star_{\text{rob}_j}\right)_+\alpha^\star_{\text{rob}_j},y\right)+\frac{\beta}{2}\sum_{j=1}^{\widehat{m}^\star}\left(\|u^\star_{\text{rob}_j}\|_2^2+\alpha^{\star 2}_{\text{rob}_j}\right)\\
\geq&\min_{(u_j,\alpha_j)_{j=1}^{\widehat{m}^\star}}\left(\max_{\Delta:X+\Delta\in\mathscr{U}}\ell\left(\sum_{j=1}^{\widehat{m}^\star}\left((X+\Delta)u_j\right)_+\alpha_j,y\right)+\frac{\beta}{2}\sum_{j=1}^{\widehat{m}^\star}\left(\|u_j\|_2^2+\alpha_j^2\right)\right)
\end{aligned}
$$

Therefore, (9) is an upper-bound to (6).

While there are an infinite number of points in the uncertainty set, which makes enumerating all points intractable, one can only enumerate all possible locations of the worst-case adversarial perturbation $\Delta^{\star\star}_{v,w}$. For some loss functions, the possible locations are finitely many points. Moreover, note that the number of D matrices is trivially upper-bounded by 2^n.

A.2 PROOF OF COROLLARY 2.1

Define $E_i = 2D_i - I_n$ for all $i \in [\widehat{P}]$. Note that each E_i is a diagonal matrix, and its diagonal elements are either -1 or 1. Therefore, for each $i \in [\widehat{P}]$, we can analyze the robust constraint $\min_{\Delta:X+\Delta\in\mathscr{U}} E_i(X+\Delta)v_i \geq 0$ element-wise (for each data point). Let e_{ik} denote the k^{th} diagonal element of E_i and $\delta_{ik}^\top$ denote the k^{th} element of Δ that appears in the i^{th} constraint. We then have:

$$
\left(\min_{\|\delta_{ik}\|_\infty\leq\varepsilon} e_{ik}(x_k^\top+\delta_{ik}^\top)v_i\right)=\left(e_{ik}x_k^\top v_i+\min_{\|\delta_{ik}\|_\infty\leq\varepsilon} e_{ik}\delta_{ik}^\top v_i\right)\geq 0 \tag{27}
$$

The minimum of the above optimization problems are achieved at $\delta^{\star\star}_{ik} = \varepsilon\cdot\text{sgn}(e_{ik}v_i) = \varepsilon\cdot e_{ik}\cdot\text{sgn}(v_i)$.

Note that as ε approaches 0, $\delta^{\star\star}_{ik}$ and $\Delta^\star_{\text{rob}}$ in Theorem 2 both approach 0, which means that the gap between the convex robust problem (14) and the non-convex adversarial training problem (12) diminishes. Plugging $\delta^{\star\star}_k$ into (27) yields that

$$
\left(e_{ik}x_k^\top v_i-\varepsilon\|e_{ik}v_i\|_1\right)=\left(e_{ik}x_k^\top v_i-\varepsilon\|v_i\|_1\right)\geq 0.
$$

Vertically concatenating $e_{ik}x_k^\top v_i-\varepsilon\|v_i\|_1\geq 0$ for all $i\in[\widehat{P}]$ gives the vectorized representation $E_iXv_i-\varepsilon\|v_i\|_1\geq 0$, which leads to (10).

Since the constraints on w are exactly the same, we also have that $\min_{\Delta:X+\Delta\in\mathscr{U}} E_i(X+\Delta)w_i\geq 0$ is equivalent to $E_iXw_i-\varepsilon\|w_i\|_1\geq 0$ for each $i\in[\widehat{P}]$.

A.3 PROOF OF THEOREM 3

The regularization term is independent from Δ. Thus, it can be ignored for the purpose of analyzing the inner maximization. Note that each D_i is diagonal, and its diagonal elements are either 0 or 1. Therefore, the inner maximization of (5) can be analyzed element-wise (cost of each data point).

The maximization problem of the loss at each data point is:

$$
\max_{\|\delta_k\|_\infty\leq\varepsilon}\left(1-y_k\sum_{i=1}^{P} d_{ik}(x_k^\top+\delta_k^\top)(v_i-w_i)\right)_+ \tag{28}
$$

where d_{ik} is the k^{th} diagonal element of D_i and $\delta_k^\top$ is the k^{th} row of Δ. One can write:

$$
\begin{aligned}
&\max_{\|\delta_k\|_\infty \le \varepsilon} \left(1 - y_k \sum_{i=1}^{P} d_{ik}(x_k^\top + \delta_k^\top)(v_i - w_i)\right)_+ \\
=&\left(\max_{\|\delta_k\|_\infty \le \varepsilon} 1 - y_k \sum_{i=1}^{P} d_{ik}(x_k^\top + \delta_k^\top)(v_i - w_i)\right)_+ \\
=&\left(1 - y_k \sum_{i=1}^{P} d_{ik} x_k^\top (v_i - w_i) - \min_{\|\delta_k\|_\infty \le \varepsilon} \delta_k^\top y_k \sum_{i=1}^{P} d_{ik}(v_i - w_i)\right)_+
\end{aligned}
$$

The optimal solution to $\min_{\|\delta_k\|_\infty \le \varepsilon} \delta_k^\top y_k \sum_{i=1}^{P} d_{ik}(v_i - w_i)$ is $\delta^\star_{\text{hinge}_k} = -\varepsilon \cdot \operatorname{sgn}\Big(y_k \sum_{i=1}^{P} d_{ik}(v_i - w_i)^\top\Big)$, or equivalently:

$$
\Delta^\star_{\text{hinge}} = -\varepsilon \cdot \operatorname{sgn}\Big(\sum_{i=1}^{P} D_i y (v_i - w_i)^\top\Big).
$$

Plug $\delta^\star_{\text{hinge}_k}$ into (28), and the optimization problem (28) reduces to:

$$
\begin{aligned}
&\left(1 - y_k \sum_{i=1}^{P} d_{ik} x_k^\top (v_i - w_i) + \varepsilon \left\| y_k \sum_{i=1}^{P} d_{ik}(v_i - w_i) \right\|_1\right)_+ \\
=&\left(1 - y_k \sum_{i=1}^{P} d_{ik} x_k^\top (v_i - w_i) + \varepsilon |y_k| \left\| \sum_{i=1}^{P} d_{ik}(v_i - w_i) \right\|_1\right)_+ .
\end{aligned}
$$

Therefore, the overall loss function is:

$$
\frac{1}{n} \sum_{k=1}^{n} \left(1 - y_k \sum_{i=1}^{P} d_{ik} x_k^\top (v_i - w_i) + \varepsilon |y_k| \left\| \sum_{i=1}^{P} d_{ik}(v_i - w_i) \right\|_1\right)_+ .
$$

In the case of binary classification, $y = \{-1, 1\}^n$, and thus $|y_k| = 1$ for all $k \in [n]$. Therefore, the above is equivalent to

$$
\frac{1}{n} \sum_{k=1}^{n} \left(1 - y_k \sum_{i=1}^{P} d_{ik} x_k^\top (v_i - w_i) + \varepsilon \left\| \sum_{i=1}^{P} d_{ik}(v_i - w_i) \right\|_1\right)_+ \tag{29}
$$

which is the objective of (14). This completes the proof.

A.4 PROOF OF THEOREM 4

In this part, we establish the equivalence between (19) and (20). To this end, we consider the constraints of (19) and argue that these can be recast as the constraints given in (20). One can write:

$$
\begin{aligned}
&\max_{\Delta: X+\Delta \in \mathscr{X}} \left\| \begin{bmatrix} \sum_{i=1}^{\widehat{P}} D_i (X+\Delta)(v_i - w_i) - y \\ 2a - \frac{1}{4} \end{bmatrix} \right\|_2 \le 2a + \frac{1}{4} \\
\iff &\max_{\|\delta_k\|_\infty \le \varepsilon,\ \forall k \in [n]} \left\| \begin{bmatrix} \sum_{i=1}^{\widehat{P}} d_{i1}(x_1^\top - \delta_1^\top)(v_i - w_i) - y_1 \\ \sum_{i=1}^{\widehat{P}} d_{i2}(x_2^\top - \delta_2^\top)(v_i - w_i) - y_2 \\ \vdots \\ \sum_{i=1}^{\widehat{P}} d_{in}(x_n^\top - \delta_n^\top)(v_i - w_i) - y_n \\ 2a - \frac{1}{4} \end{bmatrix} \right\|_2 \le 2a + \frac{1}{4} \\
\iff &\max_{\|\delta_k\|_\infty \le \varepsilon,\ \forall k \in [n]} \left(\sum_{k=1}^{n} \Big(\sum_{i=1}^{\widehat{P}} d_{ik}(x_k^\top - \delta_k^\top)(v_i - w_i) - y_k \Big)^2 + \Big(2a - \frac{1}{4}\Big)^2 \right)^{\frac{1}{2}} \le 2a + \frac{1}{4}
\end{aligned}
$$

where d_{ik} is the k^{th} diagonal element of D_i and $\delta_k^\top$ is the k^{th} row of Δ. The above constraints can be rewritten by introducing slack variables $z \in \mathbb{R}^{n+1}$ as

$$z_k \geq \left| \sum_{i=1}^{\widehat{P}} d_{ik} x_k^\top (v_i - w_i) - y_k \right| + \varepsilon \left\| \sum_{i=1}^{\widehat{P}} d_{ik}(v_i - w_i) \right\|_1, \ \forall k \in [n]$$
$$z_{n+1} \geq \left| 2a - \tfrac{1}{4} \right|, \quad \|z\|_2 \leq 2a + \tfrac{1}{4}.$$

■

A.5 PROOF OF LEMMA 6

According to Pilanci and Ergen [2020], recovering the neural network weights by plugging (3) in (22) leads to

$$\begin{aligned} q^\star &= \min_{(v_i, w_i)_{i=1}^P} \ell\left(\sum_{i=1}^P D_i X (v_i - w_i), y \right) + \beta \sum_{i=1}^P \left(\|v_i\|_2 + \|w_i\|_2 \right) \\ &= \min_{(u_j, \alpha_j)_{j=1}^{m^\star}} \ell\left(\sum_{j=1}^{m^\star} (X u_j)_+ \alpha_j, y \right) + \frac{\beta}{2} \sum_{j=1}^{m^\star} \left(\|u_j\|_2^2 + \alpha_j^2 \right) \end{aligned}$$

Similarly, we can recover the neural network weights from the solution $(\widetilde{v}_i^\star, \widetilde{w}_i^\star)_{i=1}^{\widetilde{P}}$ of (23) using:

$$(\widetilde{u}_{j_{1i}}, \widetilde{\alpha}_{j_{1i}}) = \left(\frac{\widetilde{v}_i^\star}{\sqrt{\|\widetilde{v}_i^\star\|_2}}, \sqrt{\|\widetilde{v}_i^\star\|_2} \right), \quad (\widetilde{u}_{j_{2i}}, \widetilde{\alpha}_{j_{2i}}) = \left(\frac{\widetilde{w}_i^\star}{\sqrt{\|\widetilde{w}_i^\star\|_2}}, -\sqrt{\|\widetilde{w}_i^\star\|_2} \right), \quad \forall i \in [\widetilde{P}]. \tag{30}$$

Unlike (3), zero weights are not discarded in (30). For simplicity, we use $\widetilde{u}_1, \ldots, \widetilde{u}_{\widetilde{m}^\star}$ to refer to the hidden layer weights and use $\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_{\widetilde{m}^\star}$ to refer to the output layer weights recovered using (30). Since $(\widetilde{v}_i^\star, \widetilde{w}_i^\star)_{i=1}^{\widetilde{P}}$ is a solution to (23), it satisfies $(2D_i - I_n) X \widetilde{v}_i^\star \geq 0$ and $(2D_i - I_n) X \widetilde{w}_i^\star \geq 0$ for all $i \in [\widetilde{P}]$. Thus, we can apply Lemma 5 and obtain:

$$\begin{aligned} \widetilde{q}^\star =& \ell\left(\sum_{i=1}^{\widetilde{P}} D_i X (\widetilde{v}_i^\star - \widetilde{w}_i^\star), y \right) + \beta \sum_{i=1}^{\widetilde{P}} \left(\|\widetilde{v}_i^\star\|_2 + \|\widetilde{w}_i^\star\|_2 \right) \\ =& \ell\left(\sum_{j=1}^{\widetilde{m}^\star} (X \widetilde{u}_j^\star)_+ \alpha_j, y \right) + \frac{\beta}{2} \sum_{j=1}^{\widetilde{m}^\star} \left(\|\widetilde{u}_j^\star\|_2^2 + \widetilde{\alpha}_j^{\star 2} \right) \\ \geq & \min_{(u_j, \alpha_j)_{j=1}^{\widetilde{m}^\star}} \ell\left(\sum_{j=1}^{\widetilde{m}^\star} (X u_j)_+ \alpha_j, y \right) + \frac{\beta}{2} \sum_{j=1}^{\widetilde{m}^\star} \left(\|u_j\|_2^2 + \alpha_j^2 \right) \end{aligned}$$

Since $\widetilde{P} \geq P$, $m^\star \leq 2P$ and $\widetilde{m}^\star = 2\widetilde{P}$, we have $\widetilde{m}^\star \geq m^\star$. Therefore, according to section 2 and Theorem 6 of [Pilanci and Ergen, 2020], we have

$$\begin{aligned} q^\star &= \min_{(u_j, \alpha_j)_{j=1}^{m^\star}} \ell\left(\sum_{j=1}^{m^\star} (X u_j)_+ \alpha_j, y \right) + \frac{\beta}{2} \sum_{j=1}^{m^\star} \left(\|u_j\|_2^2 + \alpha_j^2 \right) \\ &= \min_{(u_j, \alpha_j)_{j=1}^{\widetilde{m}^\star}} \ell\left(\sum_{j=1}^{\widetilde{m}^\star} (X u_j)_+ \alpha_j, y \right) + \frac{\beta}{2} \sum_{j=1}^{\widetilde{m}^\star} \left(\|u_j\|_2^2 + \alpha_j^2 \right) \\ &\leq \widetilde{q}^\star \end{aligned}$$

The above shows that a neural network with more than m neurons in the hidden layer will yield the same loss as the neural network with m neurons when optimized.

Note that (23) can always attain $q^\star$ by simply plugging in the optimal solution of (22) and assigning 0 to all other additional v_i and w_i, implying that $q^\star \geq \widetilde{q}^\star$. Since $q^\star$ is both an upper bound and a lower bound of $\widetilde{q}^\star$, we have $\widetilde{q}^\star = q^\star$, proving that as long as all matrices in $\mathscr{D}$ are included, the existence of redundant matrices does not change the optimal objective value.

A.6 ℓ_P NORM-BOUNDED PERTURBATION SET FOR HINGE LOSS

Theorem 3 and can be extended to the following ℓ_p norm-bounded perturbation set:

$$\widetilde{\mathscr{X}} = \{X + \Delta \in \mathbb{R}^{n \times d} : \Delta = [\delta_1^\top; \ldots; \delta_n^\top],\ \|\delta_k\|_p \leq \varepsilon,\ \forall k \in [n]\}$$

In the case of performing binary classification with a hinge-lossed neural network, the convex upper bound adversarial training problem then becomes:

$$\begin{aligned} \min_{(v_i, w_i)_{i=1}^{\widehat{P}}} & \left(\frac{1}{n} \sum_{k=1}^{n} \max \left(0,\ 1 - y_k \sum_{i=1}^{P} d_{ik} x_k^\top (v_i - w_i) + \varepsilon \left\| \sum_{i=1}^{P} d_{ik} (v_i - w_i) \right\|_{p*} \right) + \beta \sum_{i=1}^{\widehat{P}} \left(\|v_i\|_2 + \|w_i\|_2 \right) \right) \\ \text{s.t.} & \quad (2D_i - I_n) X v_i \geq \varepsilon \|v_i\|_{p*}, \quad (2D_i - I_n) X w_i \geq \varepsilon \|w_i\|_{p*}, \quad \forall i \in [\widehat{P}] \end{aligned} \tag{31}$$

where $D_1, \ldots, D_{\hat{P}}$ are all distinct diagonal matrices $\mathrm{Diag}([Xu \geq 0])$ that can be obtained for all possible $u \in \mathbb{R}^d$ and all $X + \Delta$ at the *boundary* of $\widetilde{\mathscr{X}}$. Moreover, $\|\cdot\|_{p*}$ is the dual norm of $\|\cdot\|_p$.