# Practical Convex Formulations of One-hidden-layer Neural Network Adversarial Training

Yatong Bai[1], Tanmay Gautam[2], Yu Gai[3], Somayeh Sojoudi[4]

***Abstract*— As neural networks become more prevalent in safety-critical systems, ensuring their robustness becomes essential. "Adversarial training" is one of the most common methods for training neural networks to be robust to adversarial perturbations. Current adversarial training algorithms, such as fast gradient sign method (FGSM) and projected gradient descent (PGD), solve highly non-convex bi-level optimizations. These algorithms suffer from the lack of convergence guarantees and can exhibit an unstable behavior. A recent work has shown that the (non-robust) standard training formulation of a one-hidden-layer, scalar-output fully-connected neural network with rectified linear unit (ReLU) activations can be reformulated as a finite-dimensional convex program. This result enables the use of global optimization methods for this class of neural networks. In this paper, we leverage this "convex training" framework to tackle the problem of adversarial training. Unfortunately, the scale of the convex training program proposed in the literature grows exponentially in data size. We prove that a stochastic approximation procedure, previously known as a heuristic, yields high-quality solutions with a scale linear in training data size. With the complexity roadblock removed, we derive convex optimization models that efficiently perform adversarial training. Our convex methods provably produce an upper bound to the global optimum of the adversarial training objective and can be applied to both binary classification and regression. We demonstrate in experiments that the proposed method achieves a noticeably superior adversarial robustness and performance compared with the existing methods.**

## I. Introduction

The neural network, as one of the most powerful and popular machine learning tools, are vulnerable to adversarial attacks. In the field of computer vision, for instance, slight manipulations of the input images can elicit misclassifications in neural networks with high confidence [1]–[3]. Neural networks have found a wide range of applications especially in control theory [4], where robustness is a high priority. More recently, deep reinforcement learning has emerged as a powerful technique for the control of highly non-linear and complex systems [5], [6]. An adversarial attack on the underlying neural network may cause the control system to fail completely [7]. Thus, it is crucial to analyze the adversarial robustness of neural networks, especially when they are applied to safety-critical or mission-critical systems.

[1]Department of Mechanical Engineering, University of California, Berkeley. yatong_bai@berkeley.edu

[2]Department of Electrical Engineering and Computer Science, University of California, Berkeley. tgautam23@berkeley.edu

[3]Department of Electrical Engineering and Computer Science, University of California, Berkeley. yu_gai@berkeley.edu

[4]Department of Electrical Engineering and Computer Science and Department of Mechanical Engineering, University of California, Berkeley. sojoudi@berkeley.edu

While there have been studies on robustness certifications [8], [9], researchers have also been working extensively on training classifiers whose predictions are robust to input perturbations [3], [10], [11]. "Adversarial training" is one of the most effective methods to train robust classifiers, compared with other methods such as obfuscated gradients [12]. Adversarial training replaces the standard loss function with a robust loss function and solves a bi-level mini-max optimization. More recently, [13] used "random smoothing" to achieve robustness. However, this method is more suitable for defending $\ell_2$ attacks than defending the more common $\ell_\infty$ attacks [14]. Previously, researchers have applied convex relaxation techniques to adversarial training. These works obtain robust convex certificates that that provide an upper bound on the inner maximum of the adversarial training formulation and use weak duality to develop upper bound loss functions that can be directly optimized with back-propagation [15], [16]. While these works rely on convex relaxations, the resulted training formulations are still non-convex.

Training neural networks involves optimizing non-convex objectives. In practice, training usually relies on Stochastic Gradient Descent (SGD) back-propagation, which only guarantees convergence to a local minimum for non-convex programs. While gradient descent can converge to a global optimizer for one-hidden-layer ReLU networks when the considered network is wide enough [17], [18], spurious local minima still exist in general. Moreover, the non-convexity of the optimization landscape results in poor interpretability and excessive sensitivity to hyperparameters. These issues become worse when adversarial training is incorporated: adversarial training can be highly unstable in practice.

Convex programs have the favorable property that all local minima are global. To overcome the issue of arriving at spurious local minima when training neural networks, existing works have considered convexifying the neural network training problem [19], [20]. More recently, [21] proposed a convex optimization problem with the same global minimum as the non-convex cost function for a one-hidden-layer fully-connected ReLU neural network.

Therefore, extending "convex training" to adversarial training has become a natural solution to the optimization difficulty issues. Unfortunately, the size of the convex program proposed in [21] grows exponentially in the training data matrix rank, leading to exponential overall complexity. While [21] proposed a heuristic method to reduce the computation through approximation, it did not provide theoretical insights into the approximation quality. To bridge this gap, we theoretically bound the quality and show that the scale of

this approximation is linear in training data size.

With these roadblocks cleared, we build upon the aforementioned works to develop "convex adversarial training", explicitly focusing on the hinge loss (binary classification) and the squared loss (regression). We mathematically show that solving the proposed robust convex optimizations trains robust neural networks and empirically demonstrate the advantages over traditional methods. The theoretical analysis focuses on one-hidden-layer neural networks but can extend to more complex architectures.

Due to space restrictions, some of the proofs are moved to the appendix, which also includes additional numerical experiments.

## II. Background

### A. Notations

Throughout this work, we focus on fully-connected neural networks with one ReLU-activated hidden layer and a scalar output, defined as $\widehat{y} = \sum_{j=1}^{m}(Xu_j)_+\alpha_j$, where $X \in \mathbb{R}^{n\times d}$ is the input data matrix with $n$ data points in $\mathbb{R}^d$ and $\widehat{y} \in \mathbb{R}^n$ is the corresponding output vector. We denote the target output used for training as $y \in \mathbb{R}^n$. $u_1, \dots, u_m \in \mathbb{R}^d$ are the weight vectors of the $m$ neurons in the hidden layer while $\alpha_1, \dots, \alpha_m \in \mathbb{R}$ are the weights of the output layer. The symbol $(\cdot)_+ = \max\{0, \cdot\}$ indicates the ReLU activation.

Furthermore, let $\|\cdot\|_p$ denote the $\ell_p$-norm within $\mathbb{R}^n$ and $\odot$ denote the Hadamard product. For $P \in \mathbb{N}_+$, we define $[P]$ as the set $\{a \in \mathbb{N}_+ | a \le P\}$, where $\mathbb{N}_+$ is the positive integer set. For $q \in \mathbb{R}^n$, $\mathrm{sgn}(q) \in \mathbb{R}^n$ denotes the sign of each entry of $q$ and $[q \ge 0]$ denotes a boolean vector in $\{0,1\}^n$ that represents the non-negativeness of each entry of $q$. The symbol $\mathrm{diag}(q)$ 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 $i \ne j$. 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 $a \ge b$ means $a_i \ge b$ for all $i$. The notation $\Pi_{\mathcal{S}}(\cdot)$ denotes the projection onto the set $\mathcal{S}$ and $|\mathcal{S}|$ denotes the cardinality of the set. For $r \in \mathbb{R}^n$, $r \sim \mathcal{N}(0, I_n)$ indicates that $r$ is a standard normal random vector.

### B. Adversarial training

Adversarial training is the main problem under study in this paper. We now formally introduce this problem. A classifier is considered adversarially robust if it assigns the same label to all inputs within an $\ell_\infty$ bound with radius $\epsilon$ [3]. The perturbation set can be defined formally as

$$\mathcal{X} = \Big\{ X + \Delta \in \mathbb{R}^{n\times d} \;\Big|\; \Delta = [\delta_1, \dots, \delta_n]^\top, \delta_k \in \mathbb{R}^d, \|\delta_k\|_\infty \le \epsilon, \forall k \in [n] \Big\}. \quad (1)$$

One standard method for training robust classifiers minimizes the "robust cost", defined as the maximum loss within the perturbation. The process of "training with adversarial data" is often referred to as "adversarial training", as opposed to "standard training" that trains on clean data. Formally, this method solves the minimax problem

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

(see [11] for more details). In the prior literature, Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) are commonly used to numerically solve the inner maximization of (2) and generate adversarial examples [11]. The outer minimization of (2) is still solved with SGD back-propagation. We abbreviate these traditional methods as GD-FGSM and GD-PGD. More specifically, PGD generates adversarial examples $\tilde{x}$ by running the iterations

$$\tilde{x}^{t+1} = \Pi_{\mathcal{X}}\Big( \tilde{x}^t + \gamma \cdot \mathrm{sgn}\Big( \nabla_x \ell\big( \sum_{j=1}^{m} (x^\top u_j)_+\alpha_j, y \big) \Big) \Big) \quad (3)$$

for $t = 0, 1, \dots$, where $x^t$ is the perturbed data vector at iteration $t$, the initial vector $\tilde{x}^0$ is the unperturbed data $x$, $\Pi_{\mathcal{X}}$ denotes the projection onto the set $\mathcal{X}$, and $\gamma > 0$ is the step size. The projection step can be performed by simply clipping the coordinates that deviate more than $\epsilon$ from $x$. FGSM can be regarded as running PGD for a single step with a large step size.

In the previous literature, GD-FGSM and GD-PGD have demonstrated their capabilities of training robust networks in various settings [3], [10], [11], [22]. However, they suffer from major issues:

- **Poor interpretability:** With GD-PGD and GD-FGSM, it is hard to monitor the training status. For example, when the training loss is high, it is unclear whether a satisfactory robustness has been achieved (the inner maximization works well) or the training was unsuccessful (the outer minimization fails).
- **Sensitivity to hyperparameters:** The hyperparameters of GD-PGD include the number of epochs, batch size, and step size of SGD back-propagation (for outer minimization), and the step size and the number of steps of PGD (for inner maximization). Although the value of each parameter directly affects the the performance, it is challenging to design them. The method is also sensitive to the initializations.
- **Lack of optimality guarantees:** The inner maximization problem of (2) is non-concave, and the outer minimization is non-convex in general. Convergence guarantees are lacking for both subproblems.
- **Vanishing / exploding gradients:** For back-propagation, the gradient at shallower layers depends on the deeper layers, thus susceptible to vanishing or exploding gradients.

Moreover, iteratively solving the bi-level optimization (2) requires an algorithm with a computationally cumbersome nested loop structure.

### C. Convex training

Here, we introduce our main analysis framework – convex training. Consider the optimization for training a one-hidden-

layer network with a regularized convex loss $\ell(\widehat{y}, y)$:

$$\min_{(u_j,\alpha_j)_{j=1}^m} \ell\Big(\sum_{j=1}^m (Xu_j)_+\alpha_j, y\Big) + \frac{\beta}{2}\sum_{j=1}^m \big(\|u_j\|_2^2+\alpha_j^2\big), \quad (4)$$

where $\beta > 0$ is a regularization parameter. Consider a set of diagonal matrices $\{\mathrm{diag}([Xu \geq 0]) \mid u \in \mathbb{R}^d\}$, and denote the distinct elements of this set as $D_1, \ldots, D_P$. The constant $P$ is the total number of partitions of $\mathbb{R}^d$ by hyperplanes passing through the origin that are also perpendicular to the rows of $X$ [21]. Intuitively, the $D_i$ matrices represent the ReLU activation patterns associated with $X$.

Consider the convex optimization problem

$$\begin{aligned}\min_{(v_i,w_i)_{i=1}^P} \ell\Big(\sum_{i=1}^P D_iX(v_i-w_i), y\Big) + \beta\sum_{i=1}^P \big(\|v_i\|_2+\|w_i\|_2\big)\\ \text{s.t. } (2D_i-I_n)Xv_i \geq 0,\ (2D_i-I_n)Xw_i \geq 0,\ \forall i \in [P]\end{aligned} \quad (5)$$

and its dual formulation

$$\max_v -\ell^*(v) \quad \text{s.t. } |v^\top (Xu)_+| \leq \beta,\ \forall u : \|u\|_2 \leq 1, \quad (6)$$

where $\ell^*(v) = \max_z z^\top v - \ell(z, y)$ is the Fenchel conjugate function. Note that (6) is a convex semi-infinite program. The next theorem borrowed from [21, Theorem 6] explains the relationship between the non-convex training problem (4), the convex problem (5), and the dual problem (6) when the neural network is sufficiently wide.

**Theorem 1.** *Let $(v_i^\star, w_i^\star)_{i=1}^P$ denote a solution of (5) and define $m^\star$ as $|\{i : v_i^\star \neq 0\}| + |\{i : w_i^\star \neq 0\}|$. Suppose that the neural network width $m$ is at least $m^\star$, where $m^\star$ is upper-bounded by $n+1$. If the loss function $\ell(\cdot, y)$ is convex, then (4), (5), and (6) share the same optimal objective. The optimal network weights $(u_j^\star, \alpha_j^\star)_{j=1}^m$ can be recovered using the formulas*

$$\begin{aligned}(u_{j_{1i}}^\star, \alpha_{j_{1i}}^\star) &= \Big(\frac{v_i^\star}{\sqrt{\|v_i^\star\|_2}}, \sqrt{\|v_i^\star\|_2}\Big) \qquad \text{if } v_i^\star \neq 0;\\ (u_{j_{2i}}^\star, \alpha_{j_{2i}}^\star) &= \Big(\frac{w_i^\star}{\sqrt{\|w_i^\star\|_2}}, -\sqrt{\|w_i^\star\|_2}\Big) \quad \text{if } w_i^\star \neq 0.\end{aligned} \quad (7)$$

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

While Theorem 1 requires an over-parameterized neural network, convex training can be applied to networks much narrower than $m^\star$, as will be shown in Algorithm 1.

## III. Practical Convex Training

Unfortunately, the worst-case computational complexity of solving (5) is $\mathcal{O}\big(d^3r^3(\frac{n}{r})^{3r}\big)$ using standard interior-point solvers [21], prohibitively high for many practical applications. Here, $r$ is the rank of the data matrix $X$ and in many cases $r = d$. This high complexity makes convex training impractical. Before we use this framework to address the problems of adversarial training, we need to break down the complexity bottleneck of convex training.

The high complexity arises because the total number of $D_i$ matrices is upper-bounded by $\min\big\{2^n, 2r\big(\frac{e(n-1)}{r}\big)^r\big\}$. To reduce this number, [21, Remark 3.3] introduced Algorithm 1. Algorithm 1 approximately solves (5) by independently sampling a subset of the $D_i$ matrices. However, [21] did not provide theoretical insights regarding the approximation quality, and therefore the approximation remains a heuristic. The following theorem bridges this gap by providing a probabilistic bound on the suboptimality of the neural network trained with Algorithm 1.

**Algorithm 1** Practical convex training

1: Via $D_i \leftarrow \mathrm{diag}([Xa_i \geq 0])$ where $a_i \sim \mathcal{N}(0, I_d)$ i.i.d. for all $i$, generate $P_s$ distinct diagonal matrices.

2: Solve

$$\begin{aligned}p_{s1}^\star = \min_{(v_i,w_i)_{i=1}^{P_s}} &\begin{pmatrix}\ell\Big(\sum_{i=1}^{P_s} D_iX(v_i-w_i), y\Big)\\ +\beta\sum_{i=1}^{P_s}\big(\|v_i\|_2+\|w_i\|_2\big)\end{pmatrix}\\ \text{s.t. } &(2D_i-I_n)Xv_i \geq 0, \quad \forall i \in [P_s],\\ &(2D_i-I_n)Xw_i \geq 0, \quad \forall i \in [P_s];\end{aligned} \quad (8)$$

3: Recover $u_1, \ldots, u_{m_s}$ and $\alpha_1, \ldots, \alpha_{m_s}$ from the solution $(v_{s_i}^\star, w_{s_i}^\star)_{i=1}^{P_s}$ of (8) using (7).

**Theorem 2.** *Consider an additional diagonal matrix $D_{P_s+1}$ uniformly sampled, and then construct*

$$\begin{aligned}p_{s2}^\star = \min_{(v_i,w_i)_{i=1}^{P_s+1}} &\begin{pmatrix}\ell\Big(\sum_{i=1}^{P_s+1} D_iX(v_i-w_i), y\Big)\\ +\beta\sum_{i=1}^{P_s+1}\big(\|v_i\|_2+\|w_i\|_2\big)\end{pmatrix}\\ \text{s.t. } &(2D_i-I_n)Xv_i \geq 0, \quad \forall i \in [P_s+1],\\ &(2D_i-I_n)Xw_i \geq 0, \quad \forall i \in [P_s+1].\end{aligned} \quad (9)$$

*It holds that $p_{s2}^\star \leq p_{s1}^\star$. Furthermore, if $P_s \geq \frac{n+1}{\psi\xi} - 1$, where $\psi$ and $\xi$ are preset confidence level constants between 0 and 1, then with probability at least $1-\xi$, it holds that $\mathbb{P}\{p_{s2}^\star < p_{s1}^\star\} \leq \psi$.*

**Proof sketch:** It can be shown that a dual problem of (5) is an instance of "uncertain convex program (UCP)". Similarly, it can be shown that a dual problem of the approximation (8) is a "sampled convex program (SCP)", which relaxes the UCP by randomly dropping some of the constraints. The quality of the SCP relaxation can then be bounded using the result presented in [23]. □

The formal proof is presented in Appendix VIII-B. Intuitively, Theorem 2 states that independently sampling an additional $D_{P_s+1}$ matrix will not reduce the training cost with high probability. One can recursively apply this bound $T$ times to show that when $P_s$ is large, the solution with $P_s$ matrices is close to the solution with $P_s + T$ matrices for an arbitrary number $T$. So, the optimality gap due to sampling will be small, and the trained network is nearly optimal.

Compared with $P$, which is exponential in $r$, $P_s$ is on the order of $\frac{n}{\xi\phi}$, linear in $n$ and independent of $r$. When $r$ is large, solving the approximated formulation (8) is exponentially more efficient than solving (5). On the other hand, Algorithm 1 is no longer deterministic due to the stochastic sampling of the $D_i$ matrices, and yields upper bounds to the global optimum of (5). We have verified empirically (shown in

Section VII-A) that even when $P_s$ is much smaller than $P$, Algorithm 1 still reliably returns a low training cost.

## IV. Convex Adversarial Training

To conquer the drawbacks of traditional adversarial training, we leverage Theorem 1 to recast (2) as robust, convex upper bound problems that can be efficiently minimized globally. We first develop a result about adversarial training involving general convex loss functions.

The connection between the convex training objective (5) and the non-convex true training cost (4) holds only when the linear constraints in (5) are satisfied. For adversarial training, we need this connection to hold at all perturbed data matrices $X + \Delta \in \mathcal{X}$. Otherwise, if some matrix $X + \Delta$ violates the constraints, then this perturbation $\Delta$ can correspond to a low convex objective but a high actual loss. To ensure the meaningfulness of the convex reformulation throughout $\mathcal{X}$, we introduce the robust constraints (10b) and (10c).

Since the $D_i$ matrices in (5) reflect the ReLU patterns of $X$, the $D_i$ matrices can change as $X$ is perturbed. Therefore, we include all distinct diagonal matrices $\operatorname{diag}([(X+\Delta)u \geq 0])$ that can be obtained for all $u \in \mathbb{R}^d$ and *all* $\Delta : X + \Delta \in \mathcal{U}$, denoted as $D_1, \ldots, D_{\widehat{P}}$, where $\widehat{P}$ is the total number of such matrices. Since $D_1, \ldots, D_{\widehat{P}}$ include $D_1, \ldots, D_P$ in (5), we have $\widehat{P} \geq P$. While $\widehat{P}$ is at most $2^n$ in the worst case, since $\epsilon$ is often small, we expect $\widehat{P}$ to be relatively close to $P$, where $P \leq 2r\left(\frac{e(n-1)}{r}\right)^r$ as discussed above.

Finally, we replace the objective of (5) with its robust counterpart, giving rise to the optimization

$$\min_{(v_i,w_i)_{i=1}^{\widehat{P}}} \left( \begin{array}{l} \max\limits_{\Delta: X+\Delta \in \mathcal{U}} \ell\Big( \sum\limits_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i - w_i), y \Big) \\ +\beta \sum_{i=1}^{\widehat{P}} \left( \|v_i\|_2 + \|w_i\|_2 \right) \end{array} \right) \tag{10a}$$

$$\text{s.t.} \min_{\Delta: X+\Delta \in \mathcal{U}} (2D_i - I_n)(X+\Delta)v_i \geq 0, \ \forall i \in [\widehat{P}], \tag{10b}$$

$$\min_{\Delta: X+\Delta \in \mathcal{U}} (2D_i - I_n)(X+\Delta)w_i \geq 0, \ \forall i \in [\widehat{P}], \tag{10c}$$

where $\mathcal{U}$ is any convex additive perturbation set. The next theorem shows that (10) is an upper bound to the robust loss function (2).

**Theorem 3.** *Let $(v^\star_{rob_i}, w^\star_{rob_i})_{i=1}^{\widehat{P}}$ denote a solution of (10) and define $\widehat{m}^\star$ as $|\{i : v^\star_{rob_i} \neq 0\}| + |\{i : w^\star_{rob_i} \neq 0\}|$. When the network width $m$ satisfies $m \geq \widehat{m}^\star$, the optimization (10) provides an upper bound on the non-convex adversarial training problem (2). The robust network weights $(u^\star_{rob_j}, \alpha^\star_{rob_j})_{j=1}^{\widehat{m}}$ can be recovered using (7). Moreover, if $\Delta^\star_{rob}$ denotes a solution to the inner maximization in (10a), then $X + \Delta^\star_{rob}$ corresponds to the worst-case adversarial inputs for the recovered neural network.*

**Proof sketch:** Since the linear constraints in (5) are satisfied by all matrices $X + \Delta$, the relationship between (5) and (4) holds for all matrices $X + \Delta$. Thus, $\Delta^\star_{\text{rob}}$ is optimal for the inner maximization of (4). Since $(u^\star_{\text{rob}_j}, \alpha^\star_{\text{rob}_j})_{j=1}^{\widehat{m}}$ may not be optimal for the outer minimization of (4), (10) is an upper bound. □

**Algorithm 2** Practical convex adversarial training

**for** $i = 1$ to $P_a$ **do**
  $a_i \sim \mathcal{N}(0, I_d)$ i.i.d.
  $D_{i1} \leftarrow \operatorname{diag}([Xa_i \geq 0])$
  **for** $j = 2$ to $S$ **do**
    $R_{ij} \leftarrow [r_1, \ldots, r_d]$, where $r_h \sim \mathcal{N}(\mathbf{0}, I_n), \forall h$
    $D_{ij} \leftarrow \operatorname{diag}([\overline{X}_{ij} a_i \geq 0])$, where $\overline{X}_{ij} \leftarrow X + \epsilon \cdot \operatorname{sgn}(R_{ij})$
    Discard repeated $D_{ij}$ matrices
    **break if** $P_s$ distinct $D_{ij}$ matrices have been sampled
  **end for**
**end for**
Solve

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} & \left( \begin{array}{l} \max\limits_{\Delta: X+\Delta \in \mathcal{U}} \ell\Big( \sum\limits_{i=1}^{P_s} D_i(X+\Delta)(v_i - w_i), y \Big) \\ +\beta \sum_{i=1}^{P_s} \left( \|v_i\|_2 + \|w_i\|_2 \right) \end{array} \right) \\ \text{s.t.} & \min_{\Delta: X+\Delta \in \mathcal{U}} (2D_i - I_n)(X+\Delta)v_i \geq 0, \ \ \forall i \in [P_s] \\ & \min_{\Delta: X+\Delta \in \mathcal{U}} (2D_i - I_n)(X+\Delta)w_i \geq 0, \ \ \forall i \in [P_s] \end{aligned} \tag{12}$$

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 (12) using (7).

The formal proof provided in Appendix VIII-C. When the perturbation set is zero, Theorem 3 reduces to Theorem 1. Rather than an exact reformulation, (10) is an upper bound problem because the robust constraints (10b) and (10c) enforce that the ReLU activation pattern of the perturbed data $X + \Delta$ remains the same within $\mathcal{X}$, effectively reducing the feasible space of neural networks and causing suboptimality. The optimality gap between (10) and (2) is solely due to this suboptimality of the outer minimization, whereas the inner maximization is exact.

In light of Theorem 3, we use optimization (10) as a surrogate for optimization (2) to train the neural network. We will show that the new problem can be efficiently solved in important cases.

For the $\ell_\infty$ perturbation set $\mathcal{X}$, the constraints in (10b) and (10c) can be equivalently replaced by the algebraic constraints

$$\begin{aligned} (2D_i - I_n)Xv_i \geq \epsilon \|v_i\|_1, \quad \forall i \in [\widehat{P}], \\ (2D_i - I_n)Xw_i \geq \epsilon \|w_i\|_1, \quad \forall i \in [\widehat{P}]. \end{aligned} \tag{11}$$

To understand this, observe that for the $\ell_\infty$ set, (10b) and (10c) become linear programming (LP) subproblems. Solving the LPs in closed forms yields (11). The detailed derivation is provided in Appendix VIII-D.

### A. Practical algorithm for convex adversarial training

Since Theorem 2 does not rely on any assumption on the matrix $X$, it applies to an arbitrary matrix $X + \Delta$, and naturally extends to the convex adversarial training formulation (10). Therefore, an approximation to (10) can be applied to train robust neural networks with widths much less than $\widehat{m}^\star$. Similar to the strategy rendered in

Algorithm 1, we use a subset of the $D_i$ matrices for practical adversarial training. Since the $D_i$ matrices depend on the perturbation $\Delta$, we also add randomness to the data matrix $X$ in the sampling process to cover $D_i$ matrices associated with different perturbations, leading to Algorithm 2. $P_a$ and $S$ are preset parameters that control the number of times we run the random weight sampling procedure, with $P_a \cdot S \geq P_s$.

## V. Convex Hinge Loss Adversarial Training

While the inner maximization of the robust problem (10) is still hard to solve in general, it is tractable for some loss functions. The simplest case is the piecewise-linear hinge loss $\ell(\widehat{y}, y) = (1 - \widehat{y} \odot y)_+$, which is widely used for classification. Here, we focus on binary classification with $y \in \{-1, 1\}^n$.

Consider the adversarial training problem for a one-hidden-layer neural network with $\ell_2$ regularized hinge loss:

$$\min_{(u_j,\alpha_j)_{j=1}^m} \left( \begin{array}{l} \max\limits_{\Delta: X+\Delta \in \mathcal{X}} \dfrac{1}{n} \cdot \mathbf{1}^\top \left( \mathbf{1} - y \odot \sum\limits_{j=1}^m \left( (X+\Delta) u_j \right)_+ \alpha_j \right)_+ \\ + \frac{\beta}{2} \sum_{j=1}^m \left( \|u_j\|_2^2 + \alpha_j^2 \right) \end{array} \right) \tag{13}$$

Applying Theorem 3 leads to the following formulation as an upper bound on (13):

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} & \left( \begin{array}{l} \max\limits_{\Delta: X+\Delta \in \mathcal{X}} \dfrac{1}{n} \cdot \mathbf{1}^\top \left( \mathbf{1} - y \odot \sum\limits_{i=1}^{\widehat{P}} D_i (X+\Delta)(v_i - w_i) \right)_+ \\ + \beta \sum_{i=1}^{\widehat{P}} \left( \|v_i\|_2 + \|w_i\|_2 \right) \end{array} \right) \\ \text{s.t.} \quad & (2D_i - I_n) X v_i \geq \epsilon \|v_i\|_1, \quad \forall i \in [\widehat{P}], \\ & (2D_i - I_n) X w_i \geq \epsilon \|w_i\|_1, \quad \forall i \in [\widehat{P}]. \end{aligned} \tag{14}$$

When generating the $D_i$ matrices, instead of enumerating an infinite number of points in $\mathcal{X}$ as suggested in Theorem 3, we only need to enumerate all vertices of $\mathcal{X}$, which is finite. This is because the solution $\Delta^\star_{\text{hinge}}$ to the inner maximum is always at a vertex of $\mathcal{X}$, as will be shown in Theorem 4.

**Theorem 4.** *For binary classification, the inner maximum of (14) is attained at* $\Delta^\star_{hinge} = -\epsilon \cdot \operatorname{sgn}\left( \sum_{i=1}^{\widehat{P}} D_i y (v_i - w_i)^\top \right)$*, and the bi-level optimization problem (14) is equivalent to the classic convex optimization*

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} & \left( \begin{array}{l} \dfrac{1}{n} \sum\limits_{k=1}^n \left( \begin{array}{l} 1 - y_k \sum_{i=1}^{\widehat{P}} d_{ik} x_k^\top (v_i - w_i) \\ + \epsilon \left\| \sum_{i=1}^{\widehat{P}} d_{ik} (v_i - w_i) \right\|_1 \end{array} \right)_+ \\ + \beta \sum_{i=1}^{\widehat{P}} \left( \|v_i\|_2 + \|w_i\|_2 \right) \end{array} \right) \\ \text{s.t.} \quad & (2D_i - I_n) X v_i \geq \epsilon \|v_i\|_1, \quad \forall i \in [\widehat{P}], \\ & (2D_i - I_n) X w_i \geq \epsilon \|w_i\|_1, \quad \forall i \in [\widehat{P}], \end{aligned} \tag{15}$$

*where $d_{ik}$ denotes the $k^{th}$ diagonal element of $D_i$.*

**Proof sketch:** Observe that the regularizations in (14) are independent from $\Delta$ and the rest of the objective is piecewise linear. Using the fact that $\max_\Delta (\cdot)_+$ is equivalent to $(\max_\Delta \cdot)_+$, one can reform the inner maximization of (14) into an LP. The optimal $\Delta^\star_{\text{hinge}}$ is then obtained by solving the LP in closed form. Plugging $\Delta^\star_{\text{hinge}}$ back yields (15). □

The formal proof is provided in Appendix VIII-E. The problem (15) is a finite-dimensional convex program that provides an upper bound on (13). We can thus solve (15) to robustly train the neural network. The $\ell_1$ norm term in (15) explains the regularization effect of adversarial training.

## VI. Convex Squared Loss Adversarial Training

The squared loss $\ell(\widehat{y}, y) = \frac{1}{2}\|\widehat{y} - y\|_2^2$ is another commonly used loss function in machine learning. It is widely used for regression tasks, but can also be used for classification.

Consider the non-convex adversarial training problem of a one-hidden-layer ReLU neural network trained with the $\ell_2$-regularized squared loss:

$$\min_{(u_j,\alpha_j)_{j=1}^m} \left( \begin{array}{l} \max\limits_{\Delta: X+\Delta \in \mathcal{X}} \dfrac{1}{2} \left\| \sum\limits_{j=1}^m \left( (X+\Delta) u_j \right)_+ \alpha_j - y \right\|_2^2 \\ + \frac{\beta}{2} \sum_{j=1}^m \left( \|u_j\|_2^2 + \alpha_j^2 \right) \end{array} \right). \tag{16}$$

Applying Theorem 3 leads to the following formulation as an upper bound on (16):

$$\begin{aligned} \min_{(v_i,w_i)_{i=1}^{\widehat{P}}} & \left( \begin{array}{l} \max\limits_{\Delta: X+\Delta \in \mathcal{X}} \dfrac{1}{2} \left\| \sum\limits_{i=1}^{\widehat{P}} D_i (X+\Delta)(v_i - w_i) - y \right\|_2^2 \\ + \beta \sum_{i=1}^{\widehat{P}} \left( \|v_i\|_2 + \|w_i\|_2 \right) \end{array} \right) \\ \text{s.t.} \quad & (2D_i - I_n) X v_i \geq \epsilon \|v_i\|_1, \quad \forall i \in [\widehat{P}], \\ & (2D_i - I_n) X w_i \geq \epsilon \|w_i\|_1, \quad \forall i \in [\widehat{P}]. \end{aligned} \tag{17}$$

**Theorem 5.** *The optimization problem (17) is equivalent to the convex program:*

$$\begin{aligned} \min_{(v_i,w_i,b_i,c_i)_{i=1}^{\widehat{P}},a,z} \quad & a + \beta \sum_{i=1}^{\widehat{P}} (b_i + c_i) \\ \text{s.t.} \quad & (2D_i - I_n) X v_i \geq \epsilon \|v_i\|_1, \quad \forall i \in [\widehat{P}], \\ & (2D_i - I_n) X w_i \geq \epsilon \|w_i\|_1, \quad \forall i \in [\widehat{P}], \\ & \|v_i\|_2 \leq b_i, \quad \|w_i\|_2 \leq c_i, \quad \forall i \in [\widehat{P}] \\ & z_{n+1} \geq \left| 2a - \tfrac{1}{4} \right|, \quad \|z\|_2 \leq 2a + \tfrac{1}{4} \\ z_k \geq & \left| \sum_{i=1}^{\widehat{P}} D_{ik} x_k^T (v_i - w_i) - y_k \right| + \epsilon \left\| \sum_{i=1}^{\widehat{P}} D_{ik} (v_i - w_i) \right\|_1, \\ & \forall k \in [n]. \end{aligned} \tag{18}$$

**Proof sketch:** We rewrite (17) in the form of a robust second-order cone program (SOCP) and show that the robust SOCP is equivalent to the classic convex optimization (18) using the procedures outlined in [24]. □

The formal proof is provided in Appendix VIII-F. Problem (18) is a convex optimization that can train robust neural networks. However, directly using (18) for adversarial training can be intractable due to the large number of constraints that arise when we include all $D$ matrices associated with all $\Delta$ such that $X + \Delta \in \mathcal{X}$. To this end, we can use the approximation in Algorithm 2 and sample a subset of the diagonal matrices. The optimality gap again can be characterized with Theorem 2.

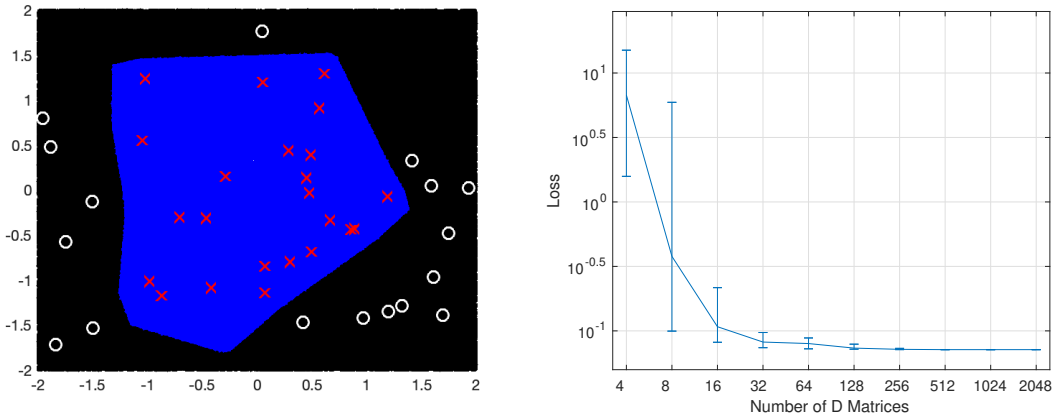


Fig. 1: The left figure is a randomized 2-dimensional dataset. The red crosses are positive training points and the white circles are negative points. The region classified as positive is in blue, whereas the negative region is in black. The right figure is the optimized training loss for each $P_s$.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

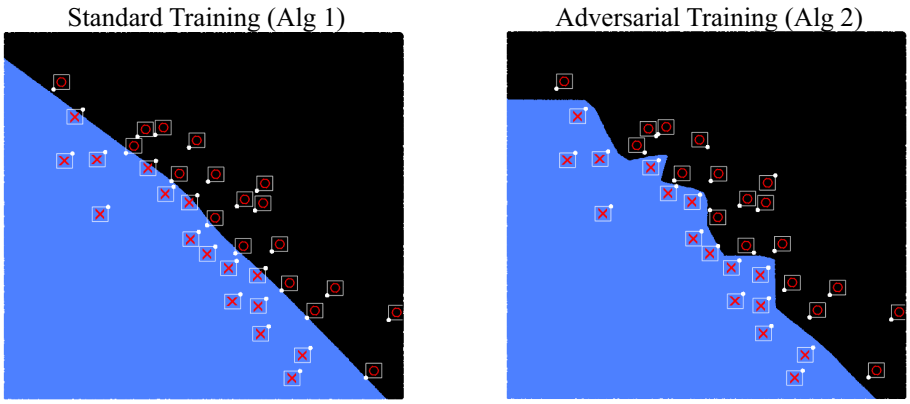


Fig. 2: Visualization of binary decision boundaries in 2-dimensional space. The red crosses $\times$ are positive training points while the red circles $\circ$ are negative points. The region classified as positive is in blue, whereas the negative region is in black. The white box around each training data is the $\ell_\infty$ perturbation bound. The white dot at a vertex of each box is the worst-case perturbation. Algorithm 2 fitted the perturbation boxes, while the standard training fitted the points.

## VII. Numerical Experiments

In this section, we focus on experimenting with the hinge loss. The experiment results with the squared loss convex adversarial training formulation (18) are provided in Appendix VIII-A.1. For all experiments, CVX [25] with the MOSEK [26] solver were used for solving the optimizations in Algorithm 1 and Algorithm 2 on a laptop computer.

### A. Approximation quality of Algorithm 1

We use numerical experiments to demonstrate the quality of the neural networks trained using the convex standard training algorithm (Algorithm 1). The experiment was performed on a randomly-generated dataset with $n = 40$ and $d = 2$. The upper bound on the number of ReLU activation patterns is $P \leq 4\left(\frac{e(39)}{2}\right)^2 = 11239$. We ran Algorithm 1 to train neural networks using the hinge loss with the number of $D_i$ matrices equal to $4, 8, 16, \ldots, 2048$, and with $\beta$ chosen as a commonly-used value $10^{-4}$. We repeated this experiment 15 times for each setting, and plotted the mean optimized loss in Figure 1. The error bars show the loss achieved in the best and the worst runs. When there are more than 128 matrices (much less than the theoretical bound on $P$), Algorithm 1 yields consistent and favorable results. Further increasing the number of $D_i$ matrices does not produce a significantly lower loss. This result supports the findings of Theorem 2. By Theorem 2, $P_s = 128$ corresponds to a confidence level of $\psi\xi = 0.318$.

### B. Convex adversarial training on 2-dimensional data

To analyze the decision boundaries obtained from convex adversarial training, we ran Algorithm 1 and Algorithm 2 on 34 random points in 2-dimensional space for binary classification. The algorithms were run with the parameters $\beta = 10^{-9}$, $P_s = 360$ and $\epsilon = 0.08$. A bias term was included by concatenating a column of ones to the data matrix $X$. The decision boundaries shown in Figure 2 confirm that Algorithm 2 fits the perturbation boxes as designed, coinciding with the theoretical prediction [11, Figure 3]. For illustration purposes, the regularization parameter $\beta$ is small to reduce the smoothing of $\ell_2$ regularization. Experiments with different

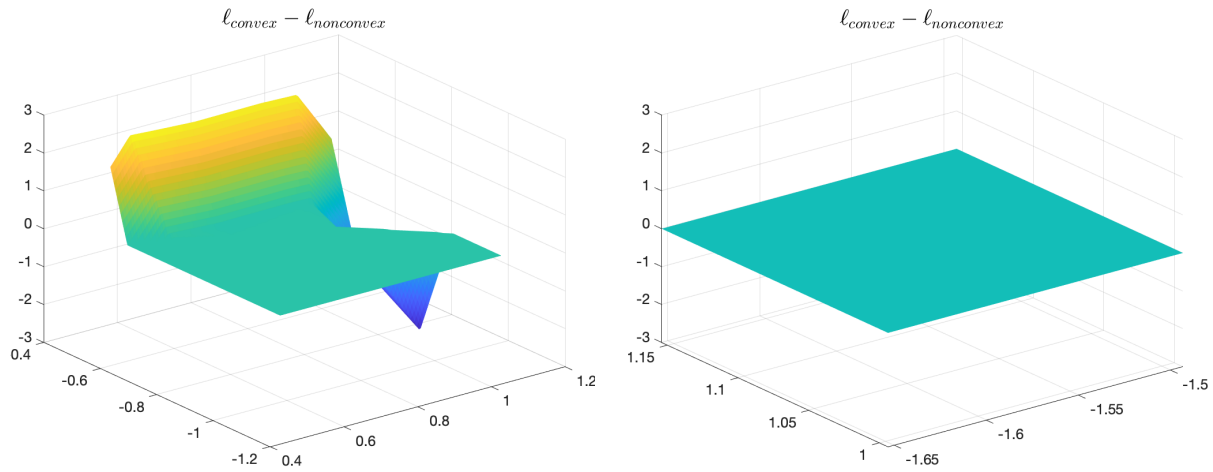


Fig. 3: $\ell_{\text{convex}} - \ell_{\text{nonconvex}}$ for $\|\delta\|_\infty \leq 0.3$ (left) and zoomed in to $\|\delta\|_\infty \leq 0.08$ (right).

TABLE I: Mean optimal objective and accuracy on clean and adversarial data (PGD and FGSM) over seven runs on the CIFAR-10 database. The numbers in the parentheses are the standard deviations over the seven runs.

| METHOD | CLEAN | FGSM ADV. | PGD ADV. | OBJECTIVE |
|---|---|---|---|---|
| GD-STD | 79.56 % (.4138%) | 47.09 % (.4290%) | 45.60 % (.4796%) | .3146 (.01101) |
| GD-FGSM | 75.30 % (3.104%) | 61.03 % (4.763%) | 60.99 % (4.769%) | .8370 ($6.681 \times 10^{-2}$) |
| GD-PGD | 76.56 % (.6038%) | 62.48 % (.2215%) | 62.44 % (.1988%) | .8220 ($3.933 \times 10^{-3}$) |
| ALGORITHM 1 | 81.01 % (.8090%) | .4857 % (.1842%) | .3571 % (.1239%) | $6.910 \times 10^{-3}$ ($3.020 \times 10^{-4}$) |
| ALGORITHM 2 | 78.36 % (.3250%) | 66.95 % (.4564%) | 66.81 % (0.4862%) | .6511 ($6.903 \times 10^{-3}$) |

choices of $\beta$ (presented in Appendix VIII-A.2) show that larger $\beta$ values yield similar behaviors, and that the decision boundaries by Algorithm 2 are more robust than GD-PGD boundaries.

### C. Convex adversarial training – the optimization landscape

This subsection visualizes that for a neural network trained with Algorithm 2, the convex landscape and the non-convex landscape overlap for an $\ell_\infty$-norm bounded perturbation $\delta$ with radius $\epsilon$ added upon a training point $x_k$. We use the same data and parameters as in Section VII-B to train a neural network. We then randomly pick one of the training points $x_k$, and plot the loss around $x_k$ for the convex objective (10a) and the non-convex objective (2). Specifically, we define

$$\ell_{\text{convex}} = \Big(1 - y_k \cdot \sum_{i=1}^{P} d_{ik}(x_k + \delta)^\top (v_i^\star - w_i^\star)\Big);$$

$$\ell_{\text{nonconvex}} = \Big(1 - y_k \cdot \sum_{j=1}^{m} \big((x_k + \delta)^\top u_j^\star\big)_+ \alpha_j^\star\Big),$$

where $d_{ik}$ is the $k^{\text{th}}$ entry of $D_i$, $y_k$ is the training label corresponding to $x_k$, and $v_i^\star$, $w_i^\star$ are the optimizers returned by Algorithm 2. Moreover, $u_j^\star$ and $\alpha_j^\star$ are the neural network weights recovered from $v_i^\star$ and $w_i^\star$ with (7).

We plot $\ell_{\text{convex}} - \ell_{\text{nonconvex}}$ for $\|\delta\|_\infty \leq 0.3$ and zoom in to $\|\delta\|_\infty \leq 0.08$ in Figure 3. When $\ell_{\text{convex}} - \ell_{\text{nonconvex}}$ is zero, the convex objective provides an exact certificate for the non-convex loss function. The right figure shows that the difference is zero for $\|\delta\|_\infty \leq 0.08$, and thereby verify that the convex objective (10a) provides an exact certification of the non-convex loss function (2) around the training points.

### D. Convex adversarial training on CIFAR-10

We then verified the real-world performance of the proposed convex training methods on a subset of the CIFAR-10 image classification dataset [27] for binary classification between the second class and the eighth class. The subset consists of 600 images downsampled to $d = 147$. The parameters were chosen as $\epsilon = 10$, $\beta = 10^{-4}$, and $P_s = 36$, so the widths of the recovered neural networks were at most 72. For back-propagation methods, the network width $m$ was set to 72.

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

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

and the PGD iterations (3) evaluates to

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

Algorithm 1 and Algorithm 2 are compared with traditional back-propagation methods GD-FGSM and GD-PGD. For GD-PGD, we used $\gamma = \epsilon/30$ and ran PGD for 40 steps.

Table I presents the CIFAR-10 experiment results. Algorithm 1 achieved a slightly higher clean accuracy compared with GD-std, and returned a much lower training cost. Such behavior supports the findings of Theorem 2. The convex

adversarial training algorithm (Algorithm 2) achieved better accuracies on clean data and adversarial data compared with GD-FGSM and GD-PGD. While Algorithm 2 solves the upper bound problem (15), it returned a lower training objective compared with GD-FGSM and GD-PGD, showing that the back-propagation methods failed to find the optimal network. Moreover, the back-propagation methods are highly sensitive to initializations and hyperparameter choices. In contrast, since Algorithm 1 and Algorithm 2 solve convex programs, they are much less sensitive and guarantee to converge to their global optima. Compared with Algorithm 1, Algorithm 2 retains the advantage in the absence of spurious local minima while vastly improving adversarial robustness.

## VIII. Conclusion

In this work, we proposed a novel "convex adversarial training" method that solves convex optimizations to train adversarially robust neural networks. Compared with traditional adversarial training methods, including GD-FGSM and GD-PGD, the favorable properties of convex optimization endow convex adversarial training with the following advantages:

- **Global convergence to an upper bound:** For the case of hinge loss and squared loss, convex adversarial training provably converges to an upper bound to the globally optimal cost, offering superior interpretability.
- **Guaranteed adversarial robustness on training data:** As shown in Theorem 4, the inner maximization over the robust loss function is solved exactly.
- **Hyperparameter-free:** In practice, Algorithm 2 can automatically determine its step size with line search, not requiring any preset parameters.
- **Immune to vanishing / exploding gradients:** The convex training method avoids this problem completely because it does not rely on back-propagation.

Overall, convex adversarial training makes it easier to train robust and interpretable neural networks, potentially facilitating their applications in the control of safety-critical systems. While this work explicitly focuses on one-hidden-layer fully-connected networks, the same robust optimization analysis extends to more sophisticated architectures, as deeper networks [28], vector-output networks [29], and certain ConvNets [30] also have their convex training representations.

## References


[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations*, 2014.

[2] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[4] W. Miller, R. Sutton, and P. Werbos, *Neural networks for control*. MIT Press, 1995.

[5] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *4th International Conference on Learning Representations*, 2016.

[6] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, pp. 39:1–39:40, 2016.

[7] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[8] B. G. Anderson, Z. Ma, J. Li, and S. Sojoudi, "Tightened convex relaxations for neural network robustness certification," in *59th IEEE Conference on Decision and Control (CDC)*, 2020.

[9] Z. Ma and S. Sojoudi, "Strengthened SDP verification of neural network robustness via non-convex cuts," *CoRR*, vol. abs/2010.08603, 2020. [Online]. Available: https://arxiv.org/abs/2010.08603

[10] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[12] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[13] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[14] A. Blum, T. Dick, N. Manoj, and H. Zhang, "Random smoothing might be unable to certify $\ell_\infty$ robustness for high-dimensional images," *J. Mach. Learn. Res.*, vol. 21, no. 211, pp. 1–21, 2020.

[15] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2018.

[16] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[17] J. Lacotte and M. Pilanci, "All local minima are global for two-layer relu neural networks: The hidden convex optimization landscape," *CoRR*, vol. abs/2006.05900, 2020.

[18] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[19] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *Journal of Machine Learning Research*, vol. 18, no. 19, pp. 1–53, 2017.

[20] Y. Bengio, N. Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *Advances in Neural Information Processing Systems*, 2006.

[21] M. Pilanci and T. Ergen, "Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[22] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *CoRR*, vol. abs/1511.03034, 2015.

[23] G. Calafiore and M. C. Campi, "Uncertain convex programs: randomized solutions and confidence levels," *Mathematical Programming*, vol. 102, no. 1, pp. 25–46, 2005.

[24] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[25] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014.

[26] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[28] T. Ergen and M. Pilanci, "Global optimality beyond two layers: Training deep relu networks via convex programs," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[29] A. Sahiner, T. Ergen, J. M. Pauly, and M. Pilanci, "Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms," in *International Conference on Learning Representations (ICLR)*, 2021.

[30] T. Ergen and M. Pilanci, "Implicit convex regularizers of cnn architectures: Convex optimization of two- and three-layer networks in polynomial time," in *International Conference on Learning Representations (ICLR)*, 2021.

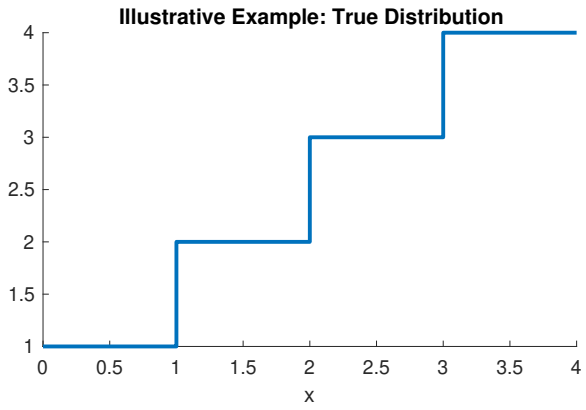


Fig. 4: True relationship between data ($x$) and target ($y$) used in the illustrative example in Section VIII-A.1. Training (with $n = 8$ points) and test (with $n = 100$ points) sets are uniformly sampled from the distribution.

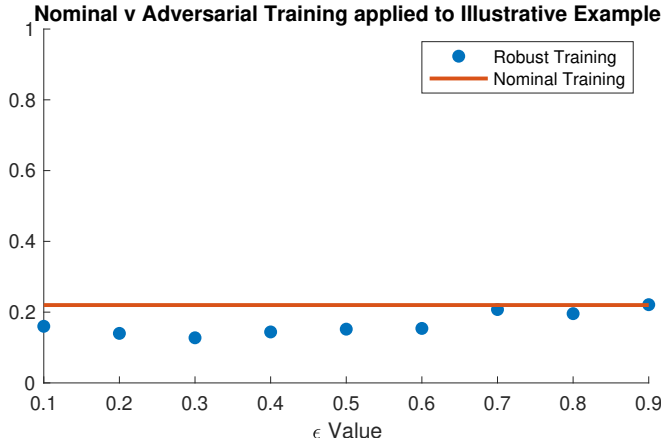


Fig. 5: This plot shows that the robust training approach (18) outperforms the standard approach for different $\epsilon \in \{0.1, ..., 0.9\}$ on the dataset studied in Section VIII-A.1.

# APPENDIX

## A. Additional experiments

*1) Experiments with squared loss adversarial training:* In this part of the appendix, the performance of the developed problem (18) was compared with the standard training problem (5) on a contrived 1-dimensional dataset. Figure 4 shows the true relationship between the data vector $X$ and the target output $y$. Throughout this experiment, training data were constructed by uniformly sampling 8 points from this distribution and test data were similarly constructed by uniformly sampling 100 points. A bias term was included by concatenating a column of ones to $X$.

The training and test procedure was repeated for 100 trials with standard training (Algorithm 1). For the adversarial training (Algorithm 2), we varied the perturbation radius $\epsilon = 0.1, \ldots, 0.9$. The training and test procedure was carried out for ten trials for each $\epsilon$. Figure 5 reports the average test mean square error (MSE) for each setup.

The adversarial training procedure outperforms standard training for all $\epsilon$ choices. We further observe that the average MSE is the lowest at $\epsilon \approx 0.3$. This behavior arises as the robust problem attempts to account for all points within the uncertainty interval around the sampled training points. When $\epsilon$ is too small, the robust problem approaches the standard training problem. Larger values of $\epsilon$ cause the uncertainty interval to overestimate the constant regions of the true distribution, increasing the MSE.

*2) Additional experiments on 2-D illustrative data:* The decision boundaries obtained from various methods with different regularization strengths are shown in Figure 6. The two standard training methods (Algorithm 1 and GD-std) learned decision boundaries that separated the training points but fail to separate the perturbation boxes. Note that Algorithm 1 learned slightly more sophisticated boundaries while GD-std learned almost-linear boundaries that were very close to one of the positive training points $\times$.

The convex adversarial training method Algorithm 2 learned boundaries that separated all perturbation boxes when $\beta$ is $10^{-3}$, $10^{-6}$, or $10^{-9}$. This behavior matches the theoretical illustration of adversarial training [11, Figure 3], and verifies that Algorithm 2 works as intended. When the regularization is too strong ($\beta = 10^{-2}$), the robust boundary becomes smoothed out and very similar to the standard training boundaries. The traditional adversarial training method GD-PGD learned boundaries that separated most perturbation boxes. However, the boundaries cut through the box at approximately $(1, -1)$ when $\beta$ is $10^{-3}$, $10^{-6}$, or $10^{-9}$. This behavior is likely caused by SGD back-propagation's the worse convergence due to non-convexity. When $\beta$ is too large, the GD-PGD boundaries also become smoothed out.

*3) Additional experiments on the CIFAR-10 dataset:* In this section, we repeat the experiments on the CIFAR-10 dataset with different numbers of sampled $D_i$ matrices. Compared with Table I, which used $\epsilon = 10$, $\beta = 10^{-4}$, and $P_s = 36$, these additional experiments keep the same $\epsilon$ and $\beta$ settings but reduce $P_s$ to 24 and 18. For fair comparisons, we set the neural network width $m$ equal to $2P_s$ for back-propagation implementations in all experiments. Each experiment was repeated seven times and the results are shown in Table II.

Table II shows that the effect of the neural network width on the prediction performance is not significant for all methods, but Algorithm 1 and Algorithm 2 are affected more: when $P_s$ is 36 or 24, Algorithm 2 outperforms GD-FGSM and GD-PGD, but when $P_s$ is 18, Algorithm 2 achieves worse FGSM and PGD accuracies. Our explanation is that the constraints in the convex training formulations become more restrictive when $P_s$ is small, worsening the suboptimalities of the solutions. Therefore, Algorithm 1 and Algorithm 2 are more suitable for neural networks that are not too narrow.

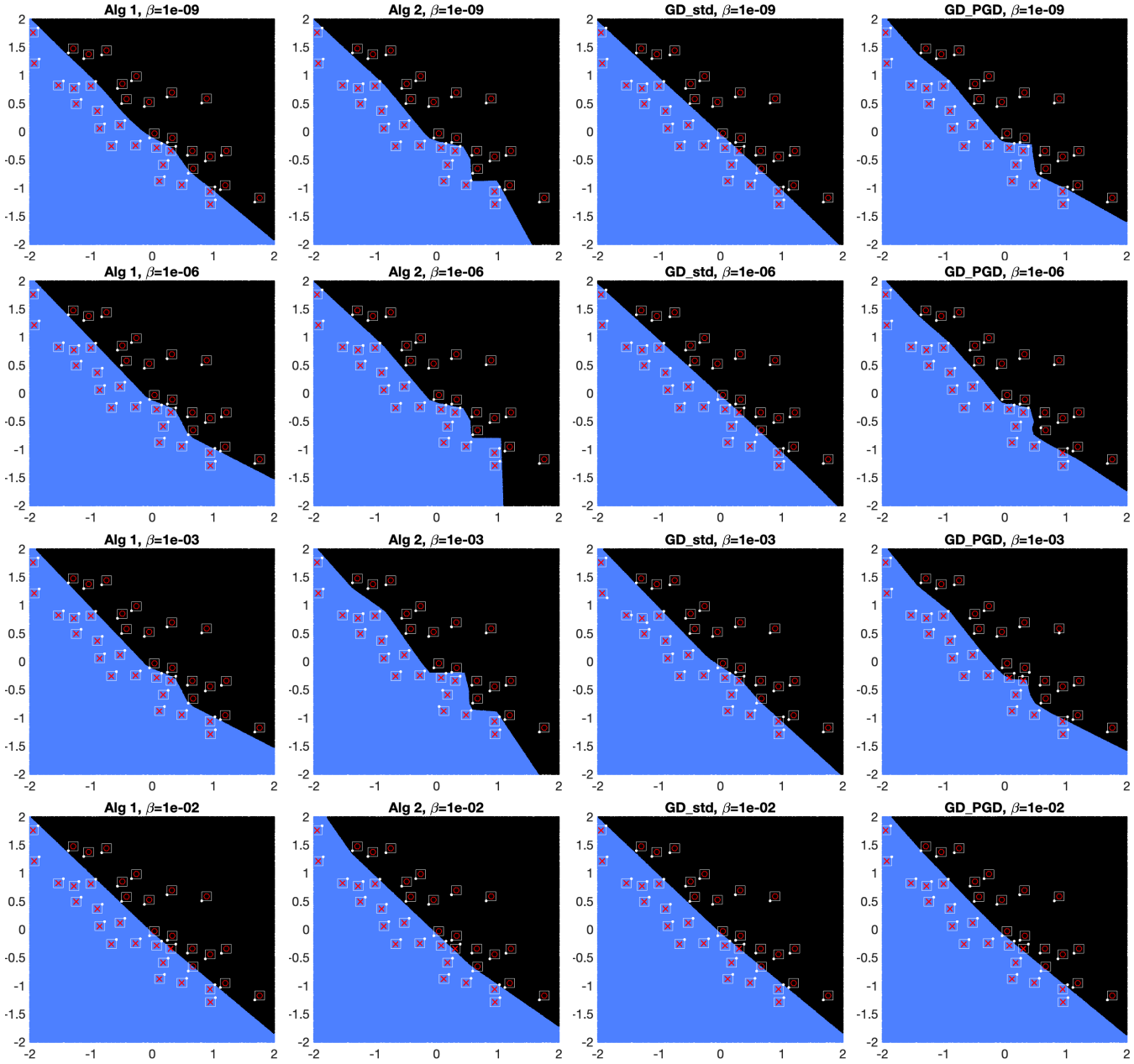


Fig. 6: Decision boundaries obtained from various methods with $\beta$ set to $10^{-9}$, $10^{-6}$, $10^{-3}$, and $10^{-2}$

### B. *Proof of Theorem 2*

We start by recasting the constraint of (6) as $\max_{\|u\|_2\leq 1} |v^\top (Xu)_+| \leq \beta$, and obtain

$$\max_{\|u\|_2\leq 1} \left|v^\top (Xu)_+\right| = \max_{\|u\|_2\leq 1} \left|v^\top \operatorname{diag}([Xu\geq 0])Xu\right| = \max_{i\in[P]} \left( \max_{\substack{\|u\|_2\leq 1 \\ (2D_i - I_n)Xu\geq 0}} \left|v^\top D_i Xu\right| \right),$$

where the last equality holds by the definition of the $D_i$ matrices: $D_1 \ldots, D_P$ are all distinct matrices that can be formed by $\operatorname{diag}([Xu\geq 0])$ for some $u\in\mathbb{R}^d$. The constraint $(2D_i - I_n)Xu \geq 0$ is equivalent to $D_i Xu \geq 0$ and $(I_n - D_i)Xu \leq 0$, which forces $D_i = \operatorname{diag}([Xu\geq 0])$ to hold.

Therefore, (6) can be recast as

$$\max_{v} -\ell^*(v) \qquad \text{s. t.} \quad \max_{\substack{\|u\|_2\leq 1 \\ (2D_i - I_n)Xu\geq 0}} \left|v^\top D_i Xu\right| \leq \beta, \ \forall i\in[P]. \tag{19}$$

To form a tractable convex program that provides an approximation to (19), one can independently sample a subset of the diagonal matrices. One possible sampling procedure is presented in Algorithm 1. The sampled matrices, denoted as

TABLE II: Optimal objective, CPU time, and test accuracy on clean and adversarial data with different neural network widths on the CIFAR-10 dataset.

$P_s = 24$ AND $m = 48$

| METHOD | CLEAN | FGSM ADV. | PGD ADV. | OBJECTIVE |
|---|---|---|---|---|
| GD-STD | 81.40 % | 54.72 % | 54.66 % | .1486 |
| GD-FGSM | 77.75 % | 64.53 % | 64.46 % | .7038 |
| GD-PGD | 76.49 % | 64.70 % | 64.64 % | .7363 |
| ALGORITHM 1 | 80.51 % | .2500 % | .1357 % | .007516 |
| ALGORITHM 2 | 78.54 % | 66.91 % | 66.75 % | .7123 |

$P_s = 18$ AND $m = 36$

| METHOD | CLEAN | FGSM ADV. | PGD ADV. | OBJECTIVE |
|---|---|---|---|---|
| GD-STD | 81.04 % | 54.86 % | 54.82 % | .1550 |
| GD-FGSM | 77.29 % | 64.69 % | 64.56 % | .7131 |
| GD-PGD | 76.44 % | 64.76 % | 64.74 % | .7365 |
| ALGORITHM 1 | 79.71 % | .3571 % | .2714 % | .008953 |
| ALGORITHM 2 | 78.71 % | 63.89 % | 63.67 % | .8049 |

$D_1, \ldots, D_{P_s}$, can be used to construct the relaxed problem:

$$d_{s1}^\star = \max_v -\ell^*(v) \qquad \text{s.t.} \max_{\substack{\|u\|_2 \le 1 \\ (2D-I_n)Xu \ge 0}} \left|v^\top D_i X u\right| \le \beta, \ \forall i \in [P_s]. \tag{20}$$

The optimization problem (20) is convex with respect to $v$. [21] has shown that (19) has the same optimal objective as its dual problem (5). By following precisely the same derivation, it can be shown that (20) has the same optimal objective as (8) and $p_{s1}^\star = d_{s1}^\star$. Moreover, if an additional diagonal matrix $D_{P_s+1}$ is independently randomly sampled to form (9), then we also have $p_{s2}^\star = d_{s2}^\star$, where

$$d_{s2}^\star = \max_v -\ell^*(v) \qquad \text{s.t.} \max_{\substack{\|u\|_2 \le 1 \\ (2D-I_n)Xu \ge 0}} \left|v^\top D_i X u\right| \le \beta, \ \forall i \in [P_s+1].$$

Thus, the level of suboptimality of (20) compared with (19) is the level of suboptimality of (8) compared with (5). Notice that by introducing a slack variable $w \in \mathbb{R}$, (19) can be represented as an instance of the uncertain convex program (UCP) with $n+1$ optimization variables, defined in [23]:

$$\max_{v,w:\ w \le -\ell^*(v)} w \qquad \text{s.t.} \max_{\substack{\|u\|_2 \le 1 \\ (2D_i-I_n)Xu \ge 0}} \left|v^\top D_i X u\right| \le \beta, \ \forall i \in [P].$$

The relaxed problem (20) can be regarded as a corresponding sampled convex program (SCP). Suppose $w^\star, v^\star$ is a solution to the sampled convex problem (20). It can be concluded from [23, Theorem 1] that if $P_s \ge \frac{n+1}{\psi\xi} - 1$, then $v^\star$ satisfies the original constraints of the UCP (19) with high probability. Specifically, with probability no smaller than $1-\xi$,

$$\mathbb{P}\Big\{ D \in \mathcal{D} \ : \max_{\substack{\|u\|_2 \le 1 \\ (2D-I_n)Xu \ge 0}} \left|v^{\star\top} D X u\right| > \beta \Big\} \le \psi.$$

where $\mathcal{D}$ denotes the set of all diagonal matrices that can be formed by $\mathrm{diag}([Xu \ge 0])$ for some $u \in \mathbb{R}^d$, which is the set formed by $D_1, \ldots, D_P$.

Since $D_{P_s+1}$ is randomly sampled from $\mathcal{D}$, we have

$$\mathbb{P}\Big\{ D \in \mathcal{D} \ : \max_{\substack{\|u\|_2 \le 1 \\ (2D-I_n)Xu \ge 0}} \left|v^{\star\top} D X u\right| > \beta \Big\} = \mathbb{P}\Big\{ \max_{\substack{\|u\|_2 \le 1 \\ (2D_{P_s+1}-I_n)Xu \ge 0}} \left|v^{\star\top} D_{P_s+1} X u\right| > \beta \Big\}$$

Thus, with probability no smaller than $1-\xi$,

$$\mathbb{P}\Big\{ \max_{\substack{\|u\|_2 \le 1 \\ (2D_{P_s+1}-I_n)Xu \ge 0}} \left|v^{\star\top} D_{P_s+1} X u\right| > \beta \Big\} \le \psi.$$

Moreover, $d_{s2}^\star < d_{s1}^\star$ if and only if $\left|v^{\star\top} D_{P_s+1} X u\right| > \beta$ with $d_{s2}^\star = d_{s1}^\star$ otherwise. The proof is completed by noting that $p_{s1}^\star = d_{s1}^\star$ and $p_{s2}^\star = d_{s2}^\star$. ■

### C. Proof of Theorem 3

Before proceeding with the proof, we first present the following result borrowed from [21].

**Lemma 6.** *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 (7), and it holds that*

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

Theorem 1 implies that the non-convex cost function (4) has the same objective value as the following finite-dimensional convex optimization problem:

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

where $D_1, \ldots, D_P$ are all of the matrices in the set of matrices $\mathcal{D}$, 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 (7).

Consider the following optimization problem:

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

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

**Lemma 7.** *It holds that $\widetilde{q}^\star = q^\star$, meaning that the optimization problem (23) has the same optimal objective as (22).*

The proof of Lemma 7 is given in Appendix VIII-G.

The robust minimax training problem (2) considers an uncertain data matrix $X + \Delta$. Different values of $X + \Delta$ within the perturbation set $\mathcal{U}$ can result in different $D$ matrices. Now, we define $\widehat{\mathcal{D}} = \bigcup_\Delta \mathcal{D}_\Delta$, where $\mathcal{D}_\Delta$ is the set of diagonal matrices for a particular $\Delta$ such that $X + \Delta \in \mathcal{U}$. By construction, we have $\mathcal{D}_\Delta \subseteq \widehat{\mathcal{D}}$ for every $\Delta$ such that $X + \Delta \in \mathcal{U}$. Thus, if we define $D_1, \ldots, D_{\widehat{P}}$ as all matrices in $\widehat{\mathcal{D}}$, then for every $\Delta$ with the property $X + \Delta \in \mathcal{U}$, the optimization problem

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

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 a convex objective given by

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

where $\mathcal{F}$ is defined as:

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

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

Moreover, consider the following problem:

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

where $\Delta^\star_{v,w}$ is the optimal point for $\max_{\Delta: X+\Delta\in\mathcal{U}} \ell\Big(\sum_{i=1}^{\widehat{P}} D_i(X+\Delta)(v_i-w_i), y\Big)$. 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:

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

Now, we analyze (25). Consider three 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:

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

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

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

In this case,

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

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

Therefore, (25) can be rewritten as

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

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

Hence, $g((v_i,w_i)_{i=1}^{\widehat{P}}) = f((v_i,w_i)_{i=1}^{\widehat{P}})$ for all $(v_i,w_i)_{i=1}^{\widehat{P}}$ belonging to the first and the third cases. $g((v_i,w_i)_{i=1}^{\widehat{P}}) < f((v_i,w_i)_{i=1}^{\widehat{P}})$ for all $(v_i,w_i)_{i=1}^{\widehat{P}}$ 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 $\Delta: X+\Delta\in\mathcal{U}$, 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 6, at those $\Delta$ where such constraints are violated, the convex problem (26) does not reflect the cost of the neural network. For these infeasible $\Delta$, the input-label pairs $(X+\Delta, y)$ can have a high cost in the neural network and potentially become the worst-case adversary. However, these $\Delta$ 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 7, it can be verified that (25) and (26) are lower bounds to (2) as long as $m\geq\widehat{m}^\star$:

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

To address the feasibility issue, we can apply robust optimization techniques ([24] Section 4.4.2) and replace the constraints in (26) with robust convex constraints, which will lead to (10). 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 (10) 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 (7), where $\widehat{m}^\star$ is the number of nonzero weights. In light of Lemma 6, 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 $X+\Delta\in\mathcal{U}$, all $X+\Delta\in\mathcal{U}$ satisfy the equality

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

Thus, since

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

giving rise:

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

Therefore, (10) is an upper bound to (2). ■

### *D. Proof of (11)*

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 $\min_{\Delta:X+\Delta\in\mathcal{U}}E_i(X+\Delta)v_i\geq 0$ element-wise (for each data point). Let $e_{ik}$ denote the $k^{\text{th}}$ diagonal element of $E_i$ and $\delta_{ik}^\top$ denote the $k^{\text{th}}$ element of $\Delta$ that appears in the $i^{\text{th}}$ constraint. We then have:

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

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

Note that as $\epsilon$ approaches 0, $\delta^{\star\star}_{ik}$ and $\Delta^\star_{\text{rob}}$ in Theorem 3 both approach 0, which means that the gap between the convex robust problem (15) and the non-convex adversarial training problem (13) diminishes. Plugging $\delta^{\star\star}_k$ into (27) yields that

$$\Big(e_{ik}x_k^\top v_i-\epsilon\|e_{ik}v_i\|_1\Big)=\Big(e_{ik}x_k^\top v_i-\epsilon\|v_i\|_1\Big)\geq 0.$$

Vertically concatenating $e_{ik}x_k^\top v_i-\epsilon\|v_i\|_1\geq 0$ for all $i\in[\widehat{P}]$ gives the vectorized representation $E_iXv_i-\epsilon\|v_i\|_1\geq 0$, which leads to (11). Since the constraints on $w$ are exactly the same, we also have that $\min_{\Delta:X+\Delta\in\mathcal{U}}E_i(X+\Delta)w_i\geq 0$ is equivalent to $E_iXw_i-\epsilon\|w_i\|_1\geq 0$ for every $i\in[\widehat{P}]$.

### *E. Proof of Theorem 4*

The regularization term is independent from $\Delta$. 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 (14) can be analyzed element-wise (cost of each data point).

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

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

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

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

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

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

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

Therefore, the overall loss function is:

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

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

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

which is the objective of (15). This completes the proof.

### *F. Proof of Theorem 5*

We first exploit the structure of (17) and reformulate it as the following robust second-order cone program (SOCP):

$$\begin{aligned} &\min_{(v_i, w_i, b_i, c_i)_{i=1}^{\widehat{P}}, a} \; a + \beta \sum_{i=1}^{\widehat{P}} (b_i + c_i) \\ &\text{s.t. } (2D_i - I_n) X v_i \geq \epsilon \|v_i\|_1, \;\; (2D_i - I_n) X w_i \geq \epsilon \|w_i\|_1, \;\; \|v_i\|_2 \leq b_i, \;\; \|w_i\|_2 \leq c_i, \;\; \forall i \in [\widehat{P}] \\ &\max_{\Delta : X + \Delta \in \mathcal{X}} \left\| \begin{bmatrix} \sum_{i=1}^{\widehat{P}} D_i (X + \Delta)(v_i - w_i) - y \\ 2a - \frac{1}{4} \end{bmatrix} \right\|_2 \leq 2a + \tfrac{1}{4}, \qquad \forall i \in [\widehat{P}]. \end{aligned} \tag{30}$$

Then, we need to establish the equivalence between (30) and (18). To this end, we consider the constraints of (30) and argue that these can be recast as the constraints given in (18). One can write:

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

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

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

■

*G. Proof of Lemma 7*

According to [21], recovering the neural network weights by plugging (7) in (22) leads to

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

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:

$$(\widetilde{u}_{j_{1i}}, \widetilde{\alpha}_{j_{1i}}) = \left(\frac{\widetilde{v}_i^\star}{\sqrt{\|\widetilde{v}_i^\star\|_2}}, \sqrt{\|\widetilde{v}_i^\star\|_2}\right), \quad (\widetilde{u}_{j_{2i}}, \widetilde{\alpha}_{j_{2i}}) = \left(\frac{\widetilde{w}_i^\star}{\sqrt{\|\widetilde{w}_i^\star\|_2}}, -\sqrt{\|\widetilde{w}_i^\star\|_2}\right), \quad \forall i \in [\widetilde{P}]. \tag{31}$$

Unlike in (7), zero weights are not discarded in (31). 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 (31). 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 6 to obtain:

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

Since $\widetilde{P} \geq P$, $m^\star \leq 2P$ and $\widetilde{m}^\star = 2\widetilde{P}$, we have $\widetilde{m}^\star \geq m^\star$. Thus, according to Section 2 and Theorem 6 of [21], we have:

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

The above inequality $q^\star \leq \widetilde{q}^\star$ 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 on $\widetilde{q}^\star$, we have $\widetilde{q}^\star = q^\star$, proving that as long as all matrices in $\mathcal{D}$ are included, the existence of redundant matrices does not change the optimal objective value. ■