2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

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 against adversaries becomes essential. "Adversarial training" is one of the most common methods for training robust networks. Current adversarial training algorithms solve highly non-convex bi-level optimization problems. These algorithms suffer from the lack of convergence guarantees and can exhibit unstable behaviors. A recent work has shown that the 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, addressing the aforementioned issues for training non-robust 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 the data size. We prove that a stochastic approximation procedure that scales linearly yields high-quality solutions. With the complexity roadblock removed, we derive convex optimization models that train robust neural networks. Our convex methods provably produce an upper bound on the global optimum of the adversarial training objective and can be applied to binary classification and regression. We demonstrate in experiments that the proposed method achieves a superior robustness compared with the existing methods.**

I. Introduction

Neural networks, 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. For example, deep reinforcement learning has been used to control highly non-linear and complex systems [5], [6]. An adversarial attack on the underlying neural network may cause the control system to completely fail [7]. Thus, it is crucial to analyze the adversarial robustness of neural networks, especially when they are applied to safety-critical systems.

While there have been studies on robustness certifications [8]–[10], researchers have also been working extensively on training classifiers whose predictions are robust to adversarial inputs [3], [11]–[13]. "Adversarial training" is one of the most effective methods to train robust networks, compared with other methods such as obfuscated gradients [14]. Adversarial training replaces the standard loss function with a robust loss function and solves a bi-level min-max optimization problem. More recently, [15] and [16] proposed "randomized smoothing", complementing adversarial training by achieving certified robustness at test time. Prior works have applied convex relaxation techniques to adversarial training. These works use weak duality to upper-bound the inner maximum of the adversarial training formulation and develop upper-bound loss functions that can be optimized with back-propagation [17], [18]. While these works rely on convex relaxations, the resulting 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 network is wide enough [19], [20], 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 globally optimal. To overcome the issue of arriving at spurious local minima when training neural networks, existing works have considered convexifying the neural network training problem [21], [22]. More recently, [23] 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 [23] grows exponentially in the training data matrix rank, leading to an exponential overall complexity. While [23] 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 the 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

*This work was supported by grants from ONR, NSF and C3.ai Digital Transformation Institute.

[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

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.

solving the proposed robust convex programs 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.

The novelties of this paper include a convex robust neural network training analysis (Sections IV-VI) and a probabilistic bound on the suboptimality of a relaxation method that was previously known as a heuristic (Section III). Due to space restrictions, some of the proofs are moved to the technical report [24], 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 ℓ_p-norm within $\mathbb{R}^n$ and $\odot$ denote the Hadamard product. For $P \in \mathbb{N}_+$, we define $[P]$ as the set $\{a \in \mathbb{N}_+ | a \le P\}$, where $\mathbb{N}_+$ is the positive integer set. For $q \in \mathbb{R}^n$, 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 $\operatorname{diag}(q)$ 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. A classifier is considered adversarially robust if it assigns the same label to all inputs within an ℓ_∞ bound with radius ϵ [3]. The perturbation set can be defined formally as

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

One standard method for training robust classifiers minimizes the "robust cost", defined as the maximum loss within the perturbation set. 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 min-max problem

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

(see [12] 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 [12]. 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

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

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 ϵ from x. FGSM can be regarded as running PGD for a single step with a large step size.

While GD-FGSM and GD-PGD have demonstrated their capabilities of training robust networks in various settings [3], [11], [12], they suffer from the following 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 can be unclear whether a satisfactory robustness has been achieved or the training was unsuccessful.
- **Sensitivity to hyperparameters:** The hyperparameters of GD-PGD include the number of epochs, batch size, and step size of SGD (for outer minimization), and the step size and the number of steps of PGD (for inner maximization). The value of each parameter affects the the performance, but is challenging to design. SGD back-propagation 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 gradients at shallower layers depend on deeper layers, thus susceptible to vanishing or exploding gradients.

Moreover, iteratively solving the bi-level optimization (2) requires an algorithm with an inefficient 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 $\ell(\widehat{y}, y)$:

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

where $\beta > 0$ is a regularization parameter. Consider a set of diagonal matrices $\{\operatorname{diag}([Xu \ge 0]) \mid u \in \mathbb{R}^d\}$, and denote the distinct elements of this set as $D_1, \dots, D_P$. 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

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.

rows of X [23]. Intuitively, the D_i matrices represent the ReLU activation patterns associated with X.

Consider the convex optimization problem

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

and its dual formulation

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

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 [23, 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*

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

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 proved in Theorem 2.

III. Practical Convex Training

Unfortunately, the worst-case computational complexity of solving (5) is $\mathcal{O}\big(d^3 r^3 (\frac{n}{r})^{3r}\big)$ using standard interior-point solvers [23], 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, [23, Remark 3.3] introduced Algorithm 1. Algorithm 1 approximately solves (5) by independently sampling a subset of the D_i matrices. However, [23] 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 $D_i \leftarrow \text{diag}([Xa_i \geq 0])$ where $a_i \sim \mathcal{N}(0, I_d)$ i.i.d.
   for all $i$, generate $P_s$ distinct diagonal matrices.
2: Solve
   $$p_{s1}^\star = \min_{(v_i,w_i)_{i=1}^{P_s}} \begin{pmatrix} \ell\Big(\sum_{i=1}^{P_s} D_i X(v_i - w_i), y\Big) \\ +\beta \sum_{i=1}^{P_s} \big(\|v_i\|_2 + \|w_i\|_2\big)\end{pmatrix} \tag{8}$$
   $$\begin{aligned}\text{s.t.}\quad &(2D_i - I_n)Xv_i \geq 0, \quad \forall i \in [P_s],\\ &(2D_i - I_n)Xw_i \geq 0, \quad \forall i \in [P_s];\end{aligned}$$
3: 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 (8) using (7).
```

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

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

It holds that $p_{s2}^\star \leq p_{s1}^\star$. Furthermore, if $P_s \geq \frac{n+1}{\psi\xi} - 1$, where ψ and ξ 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 analysis presented in [25], which introduces the confidence level constants ψ and ξ. $\square$

The formal proof is presented in [24, Appendix 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.

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.

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 Δ 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 $\text{diag}([(X+\Delta)u \geq 0])$ 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 ϵ 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

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

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

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

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. ☐

The formal proof is provided in [24, Appendix 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.

Algorithm 2 Practical convex adversarial training

```
1:  for $i = 1$ to $P_a$ do
2:      $a_i \sim \mathcal{N}(0, I_d)$ i.i.d.
3:      $D_{i1} \leftarrow \text{diag}([Xa_i \geq 0])$
4:      for $j = 2$ to $S$ do
5:          $R_{ij} \leftarrow [r_1, \ldots, r_d]$, where $r_h \sim \mathcal{N}(\mathbf{0}, I_n), \forall h$
6:          $D_{ij} \leftarrow \text{diag}([\overline{X}_{ij} a_i \geq 0])$, where $\overline{X}_{ij} \leftarrow X + \epsilon \cdot \text{sgn}(R_{ij})$
7:          Discard repeated $D_{ij}$ matrices
8:          break if $P_s$ distinct $D_{ij}$ matrices have been sampled
9:      end for
10: end for
11: Solve
        $\min_{(v_i,w_i)_{i=1}^{\widehat{P}}} \left( \max_{\Delta: X+\Delta\in\mathcal{U}} \ell\left( \sum_{i=1}^{P_s} D_i(X+\Delta)(v_i - w_i), y \right) + \beta \sum_{i=1}^{P_s} \left( \|v_i\|_2 + \|w_i\|_2 \right) \right)$
        s.t. $\min_{\Delta: X+\Delta\in\mathcal{U}} (2D_i - I_n)(X+\Delta)v_i \geq 0, \ \forall i \in [P_s]$
             $\min_{\Delta: X+\Delta\in\mathcal{U}} (2D_i - I_n)(X+\Delta)w_i \geq 0, \ \forall i \in [P_s]$        (12)
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).
```

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 ℓ_∞ 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 ℓ_∞ set, (10b) and (10c) become linear programming (LP) subproblems. Solving the LPs in closed forms yields (11). The detailed derivation is provided in [24, Appendix 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 Δ, 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

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.

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 ℓ_2 regularized hinge loss:

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

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

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

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* $\Delta^\star_{hinge} = -\epsilon\cdot\text{sgn}\left(\sum_{i=1}^{\widehat{P}} D_i y (v_i - w_i)^\top\right)$, *and the bi-level optimization problem (14) is equivalent to the classic convex optimization*

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

where d_{ik} *denotes the* kth *diagonal element of* D_i.

Proof sketch: Observe that the regularizations in (14) are independent from Δ 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 [24, Appendix 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 ℓ_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 ℓ_2-regularized squared loss:

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

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

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

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

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

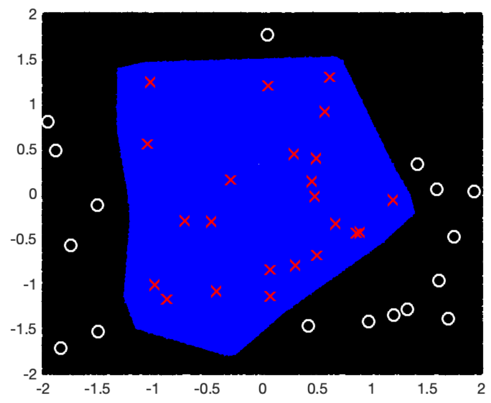
Proof sketch: We rewrite (17) in the form of a robust second-order cone program (SOCP) and show that the robust SOCP is equivalent to the classic convex optimization (18) using the procedures outlined in [26]. □

The formal proof is provided in [24, Appendix 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 Δ 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.

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 [24, Appendix A.1]. For all experiments, CVX [27] with the

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.



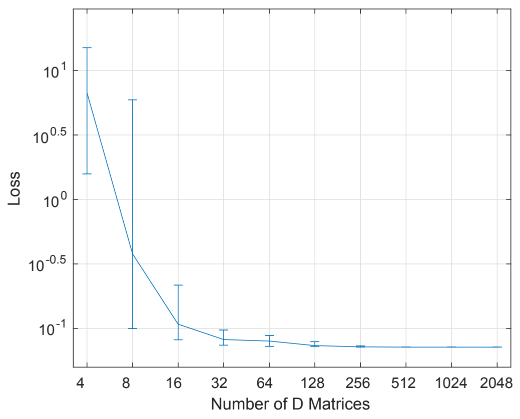
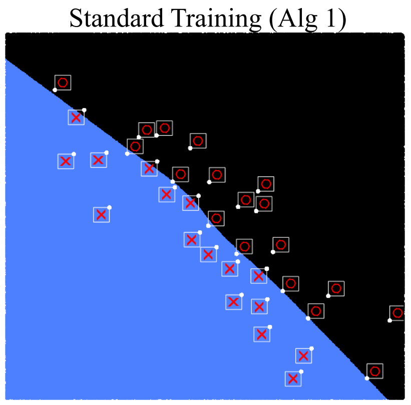


Fig. 1: The left figure is a randomized 2-dimensional dataset. The red crosses are positive training points and the white circles are negative points. The region classified as positive is in blue, whereas the negative region is in black. The right figure is the optimized training loss for each P_s. When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



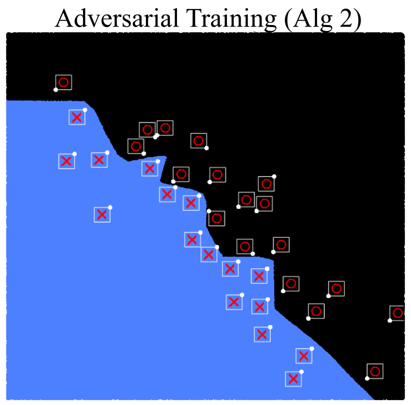


Fig. 2: Visualization of binary decision boundaries in 2-dimensional space. The red crosses $\times$ are positive training points while the red circles $\circ$ are negative points. The region classified as positive is in blue, whereas the negative region is in black. The white box around each training data is the ℓ_∞ 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.

MOSEK [28] solver was 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 β 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 [12, Figure 3]. For illustration purposes, the regularization parameter β is small to reduce the smoothing of ℓ_2 regularization. Experiments with different choices of β [24, Appendix A.2] show that larger β values yield similar behaviors, and that the decision boundaries by Algorithm 2 are more robust than GD-PGD boundaries.

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.

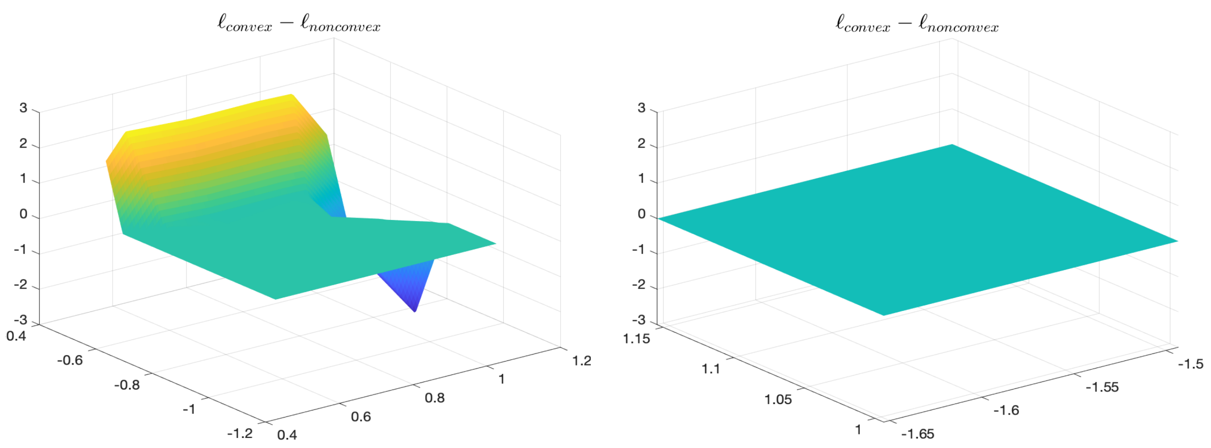


Fig. 3: $\ell_{\text{convex}} - \ell_{\text{nonconvex}}$ for $\|\delta\|_\infty \leq 0.3$ (left) and zoomed in to $\|\delta\|_\infty \leq 0.08$ (right).

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 (6.681×10^{-2})
GD-PGD	76.56 % (.6038%)	62.48 % (.2215%)	62.44 % (.1988%)	.8220 (3.933×10^{-3})
ALGORITHM 1	81.01 % (.8090%)	.4857 % (.1842%)	.3571 % (.1239%)	6.910×10^{-3} (3.020×10^{-4})
ALGORITHM 2	78.36 % (.3250%)	66.95 % (.4564%)	66.81 % (0.4862%)	.6511 (6.903×10^{-3})

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 ℓ_∞-norm bounded perturbation δ with radius ϵ 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

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

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

where d_{ik} is the kth 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 [29] 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

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

and the PGD iterations (3) evaluates to

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

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 accuracy 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 an optimal network. Moreover, the back-propagation methods are highly sensitive

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.

to initializations and hyperparameter choices. In contrast, since Algorithm 1 and Algorithm 2 solve convex programs, they are much less sensitive and are guaranteed 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

This paper proposes a novel "convex adversarial training" method that solves convex programs to train robust neural networks. Compared with traditional adversarial training methods, the favorable properties of convex optimization endow convex adversarial training with the following advantages:

- **Global convergence to an upper bound:** For the hinge loss and the squared loss, convex adversarial training provably converges to an upper bound on 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 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, since deeper networks [30], vector-output networks [31], and certain ConvNets [32] also have similar 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, "A sequential framework towards an exact SDP verification of neural networks," in *International Conference on Data Science and Advanced Analytics*, 2021.

[10] B. G. Anderson and S. Sojoudi, "Data-driven assessment of deep neural networks with random input uncertainty," *arXiv preprint arXiv:2010.01171*, 2020.

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

[12] 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.

[13] Y. Bai, B. G. Anderson, and S. Sojoudi, "Avoiding the accuracy-robustness trade-off of classifiers via local adaptive smoothing," 2022. [Online]. Available: https://github.com/Bai-YT/Public-Papers/blob/main/Adaptive_Smoothing_Preprint.pdf

[14] 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.

[15] 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.

[16] B. Anderson and S. Sojoudi, "Certified robustness via locally biased randomized smoothing," *Conference on Learning for Dynamics and Control*, 2022.

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

[18] 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.

[19] Y. Wang, J. Lacotte, and M. Pilanci, "The hidden convex optimization landscape of regularized two-layer reLU networks: an exact characterization of optimal solutions," in *International Conference on Learning Representations*, 2022.

[20] 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.

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

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

[23] 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.

[24] Y. Bai, T. Gautam, Y. Gai, and S. Sojoudi, "Practical convex formulations of one-hidden-layer neural network adversarial training," Technical Report, 2021. [Online]. Available: https://people.eecs.berkeley.edu/~sojoudi/Convex_NN.pdf

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

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

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

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

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

[30] 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.

[31] 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.

[32] 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.

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on December 21,2022 at 19:49:55 UTC from IEEE Xplore. Restrictions apply.