ECE 285: Sum of Squares Optimization 1–11, 2023

Training robust neural networks using Lipschitz constant estimation techniques

Gaurav Singh G8SINGH@UCSD.EDU

Sreevatsank Kadaveru SKADAVERU@UCSD.EDU

Varun Pawar VPAWAR@UCSD.EDU

1. Introduction

In recent times, neural networks have demonstrated tremendous dexterity in a multitude of tasks that require a human-like perception and comprehension to accomplish. These tasks have proved to be very difficult to solve using conventional hand-crafted rule-based techniques. Unlike rule-based techniques, the internal operations of neural networks are unclear and opaque to interpretation. This opaque nature of neural networks poses a significant impediment to their real-world adoption this problem is often referred to as the **Black Box** problem. It is very difficult to provide hard guarantees on their functionality and failure modes.

Compounding on the black box problem, recent studies Szegedy et al. (2013) have shown that neural networks are vulnerable to adversarial attacks. An adversarial attack is a maliciously crafted input by an attacker that is designed to can trick the neural network into misclassifying the input. Fig 1 is an example of an adversarial attack in which an elephant is misclassified as a sea snake with high confidence.

In the literature, attempts have been made to address this problem from a theoretical and practical viewpoints. The first is an attempt to provide a theoretical guarantee on the performance of a neural network, such as by providing a lower/upper bound on the robustness of a network against adversarial attacks shown in Fig 1. The Second is an attempt to train a neural network in a robust manner that guarantees the reliability of the network. We detail further on this in Section 2.

In this work, we study the above two points of view and also the recent attempt to combine the techniques.

2. Related Works

Estimating the Lipschitz constant is extensively studied in recent years. Szegedy et al. (2013) stated and later Combettes and Pesquet (2019) corroborated that the upper and lower bound on the Lipschitz constant can be estimated as a matrix norm of the network weights. Authors in Szegedy et al. (2013) estimated the Lipschitz constant of multiple-layer neural networks as a product of the Lipschitz constant of individual layers. The estimate is scalable and can be calculated efficiently, but the constant is a weak bound on the equation. Combettes and Pesquet (2019) proposed a similar algorithm that scales well with a number of hidden units per layer but performs poorly on an increasing number of layers. Later, authors in Scaman and Virmaux (2018) provided an efficient method to

© 2023 G. Singh, S. Kadaveru & V. Pawar.

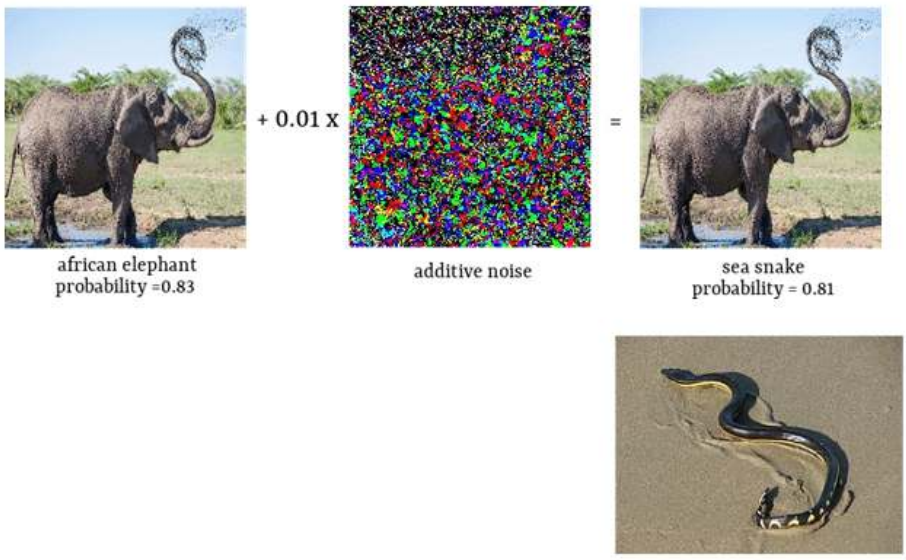


Figure 1: An example of an adversarial attack on neural network

solve the problem using a greedy algorithm to under-approximate the Lipschitz constant. Fazlyab et al. (2019b) estimates least bound Lipschitz constant using efficient SDP formulation for a deep neural network with ReLU activation function.

Parallelly, there have been advancements in robust training of neural networks which implements relaxed optimization objectives for better estimation over a perturbed input space. Most notably, methods like Wong and Kolter (2017) and Madry et al. (2017a) solve the training problem using convex relaxation on the base neural network optimization problem that ensures l_∞ norm ϵ-bound guarantees on the neural network.

In Pauli et al. (2022) the author attempts to integrate the Lipschitz constant estimation technique to train a neural network in a robust manner.

3. Robust Neural Network

Robustness is a notion that refers to the resilience of a Neural Network against adversarial attacks. We formalise the notion of robustness in order to be able to mathematically quantify the robustness of a neural network and to be able to devise strategies to train a robust neural network

3.1. Definitions of robustness

There are many ways to define the notion of robustness, below are three common notions:

- Bounded Input robustness: A network f_θ is said to be ϵ robust if the output of the network remains constant for all inputs within an input norm ball of radius ϵ.

$$y = f_\theta(x),\ \forall\ ||x - x_0|| < \epsilon$$

- Probabilistically bounded robustness: A network f_θ is said to be ϵ robust if the misclassification probability of the network is less than ϵ for random perturbation of input data δ_x drawn

from a distribution $\mathcal{P}_{\delta_x}$

$$P(\hat{y} \neq y |\ \hat{y} = f_\theta(x_0 + \delta_x)) < \epsilon,\ \forall\ \delta_x \sim \mathcal{P}_{\delta_x}$$

- Lipschitz bounded robustness: the robustness of the network is defined with respect to the global Lipschitz constant of the network.

$$||f_\theta(x_1) - f_\theta(x_2)|| \leq L||x_1 - x_2||,\ \forall\ x, y \in \mathcal{R}^n$$

3.2. Adversarial Attack Models

The adversarial attacks on neural network can broadly be categorised into **black box** attacks and **white box** attacks, based on the access level of the attacker. An attack is called a "black box" attack if the attacker has no knowledge of the weights of the neural network being attacked, where as if the attacker has access to the weights then those attacks are classified as "white box" attacks. In this study our focus is primarily on white box attacks.

White box attacks of neural networks use the back propagation gradients to craft their attack vector. The attacker uses these gradients to introduce perturbations to the input. We use an l_∞ norm bounded perturbation model as it encapsulates all other norms and is considered a natural notion for adversarial perturbation Madry et al. (2017b). Some example attacks are detailed below:

- Fast Gradient Sign Method (FGSM) attack:

$$\tilde{x} = x + \epsilon * sgn(\nabla_x L(\theta, x, y))$$

- t-step Projected Gradient Descent (PGD) attack

$$x^t = \prod_{x+\mathcal{S}} (x^{t-1} + \alpha * sgn(\nabla_x L(\theta, x^{t-1}, y)))$$

Note that the FGSM attack is a single step method, where as PGD attack is an iterative multistep algorithm. The multi-step attack is more potent as one would expect and its potency increases with the number of iterations t Madry et al. (2017b).

4. Current methods in Robust Neural Network Training

Research in adversarial attack of Robust Neural network is studied through the lens of robust optimization. Typically, a saddle point(min-max) formulation is used to understand the robustness against adversarial attacks in a structured manner.

4.1. Formulating Robust Neural Network

Given: A set of data points $\mathcal{D} = \bigcup_{i \in \{1,2,\ldots N\}} \{(x_i, y_i)\}$ where $x_i \in \mathbb{R}^n$ is an input and $y_i \in \mathbb{R}$ is the corresponding output.

Objective: For a given datapoint (x_i, y_i), with a set of perturbations defined as $\mathcal{S} \subset \mathbb{R}^n$ and loss function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^{|\theta|} \to \mathbb{R}$

$$\min_\theta\ \mathbb{E}_{(x,y)\in\mathcal{D}}[\max_{\delta \in \mathcal{S}(x,y)}\ \mathcal{L}(x + \delta, y; \theta)]$$

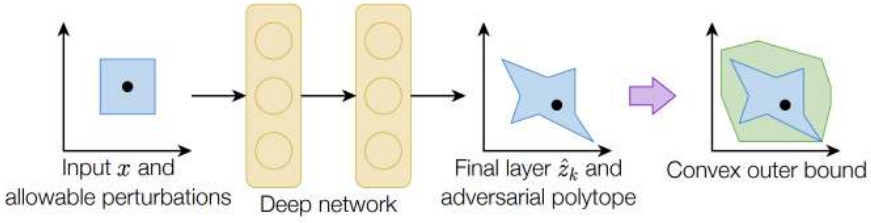


Figure 2: Illustration of (non-convex) adversarial polytope and it's convex transfomration.

Typically, we define perturbations using l_∞ norms. Given a data point $x \in \mathbb{R}^n$, we define perturbations in l_∞ bound such that $\delta \in \mathbb{R}^n$, given a $\epsilon \in \mathbb{R}_+$ and $\|\delta\|_\infty \leq \epsilon$.

l_∞-bounded Objective: Given a positive ϵ,

$$\min_\theta \mathbb{E}_{(x,y)\in\mathcal{D}}\Big[\max_{\|\hat{x}-x\|_\infty \leq \epsilon} \mathcal{L}(\hat{x}, y; \theta)\Big]$$

As you can observe, robust training objective is a min-max problem which is highly concave in nature(this is a saddle point formulation). Saddle point formulations are intractable and don't have any analytical solutions. So, there are different techniques that are used to make the problem tractable. We will be discussing two of these methods in this project:

- Convex outer-relaxation(LP-train)
- PGD-based adversary attack(PGD-train)

4.2. Convex outer-relaxation(LP-train)

Wong and Kolter (2018) developed this method to learn deep ReLU-based classifiers that are provably robust against l_∞-norm bounded adversarial perturbation. The method applies convex outer approximation to the set of ReLU activations in the neural network, see figure (2). The convex relaxation converts non-linear neural network to a affine map with affine constraint. Let, l and u be lower and upper bound of pre-activation of given activation. Then we can linearize ReLU as,

$$z = \max(0, \hat{z})$$

where $\hat{z} = W * Z_{prev} + b$ is the preactivation value of the given neuron. It is linearized as the following,

$$z \geq 0,\ z \geq \hat{z},\ -u\hat{z} + (u-l)z \leq -ul \tag{1}$$

Robust guarantees are provided by exploiting affine constraints from ReLU activation, we can define a linear program that maximizes non-target labels. Given a sample input x with known label y^*, we can find a point in $\mathcal{Z}_\epsilon$ such that it minimizes the label and maximizes some alternative target label y^{targ}. We define an optimization objective,

$$\min_{\hat{z}} (\hat{z})_{y^*} - (\hat{z})_{y^{targ}}$$
$$subject\ to\ \hat{z} \in \mathcal{Z}_\epsilon$$

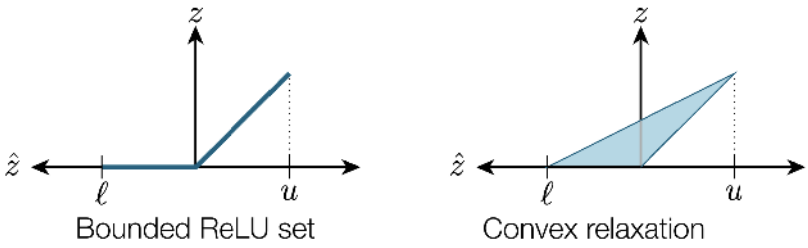


Figure 3: Illustration of the linear ReLU relaxation given a bounded set $[l, u]$.

Here, $\mathcal{Z}_\epsilon$ is a set of convex polytope after ReLU linearization and is defined as,

$$\mathcal{Z}_\epsilon - \{f(x+\Delta) : \|\Delta\|_\infty \leq \epsilon\} \tag{2}$$

THe upper and lower bound of the activations are variables in the above primal formulation. Hence it scales with the size of neural network and hence is intractable. Dual formulation of the above is solvable and is defined as,

$$\max_{\alpha} \mathcal{J}_\epsilon(x, g_\theta(c, \alpha))$$
$$subject\ to\ \alpha_{i,j} \in [0,1],\ \forall i,\ j$$

Solving the above equation gives us the cost $\mathcal{J}_\epsilon$, which is parametric in ϵ. Since the objective is a linear program, $\mathcal{J}_\epsilon$ provides a tight lower bound on the robustness objective, by strong duality. Therefore, we can rewrite robust neural network training objective as,

$$\min_{theta} \sum_{i=1}^{N} \mathcal{L}(-\mathcal{J}_\epsilon(x_i, g_\theta(e_{y_i}, 1^T - I)), y_i) \tag{3}$$

From equation (3), we observe that convex relaxation converts a robust training objective to a minization problem. This could be solved using back-propogation techniques such as gradient descent.

4.3. PGD-based adversary attack(PGD-train)

Structured adversarial attacks have been used for identifying vulnerabilities of neural network and conversely aiding the neural network to train against them. In this domain, Projected Gradient Descent(PGD) is one of the most popular white-box adversary attack methods. Madry et al. (2017b), formulated a robust training approach where PGD based perturbations are exploited for robust training(PGD-train). The authors claim that a network trained against PGD attacks yields robustness to all first-order attacks.
PGD-train algorithm is formulated as below,

$$x_{t+1} = \prod_{\|\hat{x}-x\| \leq \epsilon} (x_t + \Delta_x \mathcal{L}(\hat{x}, y; \theta))$$

here $\prod_{\mathcal{S}}$ is a projection function on a perturbation set $\mathcal{S} = \{x, \hat{x} \in \mathbb{R}^n | \ \|x - \hat{x}\| \leq \epsilon, \ \epsilon \in \mathbb{R}_+\}$

$$\min_{\theta} \max_{\|\hat{x}-x\| \leq \epsilon} \mathcal{L}(\hat{x}, y; \theta)$$

therefore,

$$x^* = \arg \max_{\|\hat{x}-x\| \leq \epsilon} \mathcal{L}(\hat{x}, y; \theta) = \lim_{t \to \infty} x_t$$

PGD-train provides a tractable solution to a robust training objective. The algorithm itself is itself in the inner projected gradient descent and thus is significantly slower than naive neural network training.

5. Lipschitz estimation using SDP formulation

A $f : \mathbb{R}^m \to \mathbb{R}^n$ is globally Lipschitz continuous on $\mathcal{X} \subseteq \mathbb{R}^n$ if there exists smallest non-negative constant $L \in \mathbb{R}_+$ such that,

$$||f(x) - f(y)||_2 \leq L||x - y||_2 \tag{4}$$

It is called as Lipschitz constant. Lipschitz constant gives a ratio of variation in the output space compared to the input space, thus measuring the sensitivity of a function. This idea is extended to deep-neural network and thus Lipschitz constant provides robustness certification for a classifier.
Till date, there isn't any analytical algorithm to measure the Lipschitz constant of a general deep neural network. But there has been extensive research done in this domain to estimate the tight bounds on the Lipschitz constant. In the following section we will discuss lip-SDP. This method proves to provide the tighest bound on the Lipschitz constant. We will also compare the method against other Lispchitz constatn estimation methods including naive upper and lower bound, seqLip and CPLip.

5.1. SDP formulation of deep Neural Network

We exploit the slope restriction on activation functions in Neural Network to derive better bounds on the Lipschitz constant. The bounds are then estimated via SDP constraints, thus earning it the name of LipSDP. Slope restriction on activation function φ evolves as follows:

$$\alpha \leq \frac{\varphi(y) - \varphi(x)}{y - x} \leq \beta \tag{5}$$

$$(\frac{\varphi(y) - \varphi(x)}{y - x} - \alpha)(\frac{\varphi(y) - \varphi(x)}{y - x} - \beta) \leq 0 \tag{6}$$

This evolves to:

$$\begin{bmatrix} x - y \\ \varphi(x) - \varphi(y) \end{bmatrix}^T \begin{bmatrix} -2\alpha\beta I_n & (\alpha + \beta) I_n \\ (\alpha + \beta) I_n & -2 I_n \end{bmatrix} \begin{bmatrix} x - y \\ \varphi(x) - \varphi(y) \end{bmatrix} \geq 0 \tag{7}$$

Given the slope restricted function $\varphi : \mathbb{R} \to \mathbb{R}$ on $[\alpha, \beta]$, we can define:

$$\mathcal{T}_n = \{T \in \mathbb{S}^n | T = \sum_{i=1}^{n} \lambda_{ii} e_i e_i^T \lambda_{ii} \geq 0\} \tag{8}$$

So, for any $T \in \mathcal{T}_n$, borrowing from equation (5), we get:

$$\begin{bmatrix} x-y \\ \varphi(x)-\varphi(y) \end{bmatrix}^T \begin{bmatrix} -2\alpha\beta T & (\alpha+\beta)T \\ (\alpha+\beta)T & -2T \end{bmatrix} \begin{bmatrix} x-y \\ \varphi(x)-\varphi(y) \end{bmatrix} \geq 0 \tag{9}$$

Our main goal is to find the right set of parameters, T, to get a tight estimation of the bounds of the Lipschitz constant.
Let us consider a single-layered neural network described by $f(x) = W^1\phi(W^0x+b^0)+b^1$. Where $\phi(x) = [\varphi(x_1)...\varphi(x_n)]$.
Let us consider the following inequality holds for some $T \in \mathcal{T}_n$ and ρ:

$$M(\rho, T) := \begin{bmatrix} -2\alpha\beta {W^0}^T T W^0 - \rho I_{n_0} & (\alpha+\beta){W^0}^T T \\ (\alpha+\beta)TW^0 & -2T + {W^1}^T W^1 \end{bmatrix} \leq 0 \tag{10}$$

Then we can say that $||f(x) - f(y)||_2 \leq \sqrt{\rho}||x-y||_2$ holds true.
Our final problem then translates to minimizing ρ such that:

$$M(\rho, T) \leq 0, and, T \in \mathcal{T}_n \tag{11}$$

The above problem can be viewed as the SDP problem. And the same can be expanded to a multi-layered neural Network.
Prove: Equation (11) gives a feasible solution that is also Lipschitz's continuous.
Proof for equation (11): Define $x^1 = \phi(W^0x + b^0)$, $y^1 = \phi(W^0y + b^0)$ and $\sqrt{\rho} = L_2$.
Multiply $M(\rho, T)$ on left and right sides by $[(x-y)^T(x^1-y^1)^T]$ and $[(x-y)^T(x^1-y^1)^T]^T$ respectively we get:

$$\begin{bmatrix} x-y \\ x^1-y^1 \end{bmatrix}^T \begin{bmatrix} -2\alpha\beta {W^0}^T T W^0 & (\alpha+\beta){W^0}^T T \\ (\alpha+\beta)TW^0 & -2T \end{bmatrix} \begin{bmatrix} x-y \\ x^1-y^1 \end{bmatrix} \leq \begin{bmatrix} (x-y)^T & (x^1-y^1)^T \end{bmatrix} \begin{bmatrix} L_2^2 I_{n_0} & 0 \\ 0 & -{W^1}^T W^1 \end{bmatrix} \begin{bmatrix} x-y \\ x^1-y^1 \end{bmatrix}$$

Looking at the left-hand term, we can simplify it as:

$$\begin{bmatrix} (W^0x+b^0)-(W^0y+b^0) \\ x^1-y^1 \end{bmatrix}^T \begin{bmatrix} -2\alpha\beta T & (\alpha+\beta)T \\ (\alpha+\beta)T & -2T \end{bmatrix} \begin{bmatrix} (W^0x+b^0)-(W^0y+b^0) \\ x^1-y^1 \end{bmatrix} \tag{12}$$

From the above equations (7) and (10), we can determine

$$\begin{bmatrix} x-y \\ x^1-y^1 \end{bmatrix}^T \begin{bmatrix} L_2^2 I_{n_0} & 0 \\ 0 & -{W^1}^T W^1 \end{bmatrix} \begin{bmatrix} x-y \\ x^1-y^1 \end{bmatrix} \geq 0 \tag{13}$$

Or in other words

$$(x^1-y^1)^T {W^1}^T W^1 (x^1-y^1) \leq L_2^2 (x-y)^T(x-y) \tag{14}$$

Hence proved!

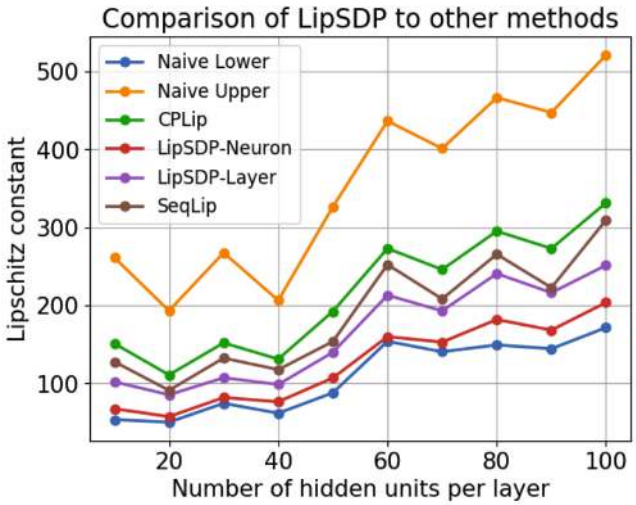


Figure 4: Comparision of LipSDP with other Lipschitz estimation methods. The network is 5-layered, trained on MNIST dataset with Adam Optimizer.

5.2. LipSDP-Neuron and LipSDP-Layer

From equation (11), the SDP objective is formulated in terms of T, where $T = \sum_{i=1}^{n} \lambda_{ii} e_i e_i^T$. In a neural network, the value n is the number of hidden neurons. Parameterization at every neuron would yield us the LipSDP-Neuron model. Therefore, with the increase in the dimensionality(number of layers) of the network, the number of variables will explode. To tackle this issue, authors introduce LipSDP-Neuron and LipSDP-Layer.

1. **LipSDP-Neuron** has $n + 1$ decision variables. For this case, we have $T = diag(\lambda_1, ...\lambda_l)$
2. **LipSDP-layer** considers only one contraint per layer i.e., $T = blkdiag(\lambda_1 I_{n_1}, ...\lambda_l I_{n_l})$. Where n_i is the number of hidden neurons in i^{th} layer.

5.3. Experimental Results

In the following experiment, Lischiptz's constant esitmate was compared between Naive Method, CPLip, SeqLip and Lip-SDP(both LipSDP-Neuron and LipSDP-Layer). From figure (4), it is observed that LipSDP-Neuron and LipSDP-Layer provides the best lower bound on the Lipschitz's constant compared to the other methods.

6. Robust Training using Lipschitz constant

In this section, we discuss a innovative technique to train a robust neural network using the Lipschitz constant. The motivation for this technique stems from the idea that by guiding the training process of a neural network using the Lipschitz constant at every step of the training process we can be confident of obtaining a robust neural network. This idea is explore in Pauli et al. (2022).In Pauli et al. (2022), the author describes two ways of using Lipschitz constant to guide the training process.

One, the Lipschitz bound estimate obtained in Fazlyab et al. (2019a) is used as a regulariser during the training process. We refer to this technique as Lipschitz regularisation (or Lip-reg) method.

The optimisation objective for Lip-reg method can be formulated as below:

$$\min_{W,L^2} \mathcal{L}(W) + \mu L^2 + 1_{Ml}(L^2, W) \tag{15}$$

Where μL^2 is the Lipschitz regulariser term, 1_{Ml} is an indicator function defined as,

$$1_{Ml}(L^2, \tilde{W}) = \begin{cases} 0, M(\rho, T) \leq 0 \\ \infty, M(\rho, T) > 0 \end{cases}$$

$M(\rho, T)$ is the same as defined in Eq 10.

We observe that in Eq 15, L is a function of the neural network weights W, and L itself is obtained by solving the SDP optimisation in Eq 11. Therefore the expression is a coupled multi-objective optimisation problem. We introduce a slack variable $\tilde{W}$ and reformulate Eq 15 in order to decouple the objective function, as shown below

$$\min_{W,\tilde{W},L^2} \mathcal{L}(W) + \mu L^2 + 1_{Ml}(L^2, \tilde{W}), s.t., W = \tilde{W} \tag{16}$$

This reformulation is essential in being able to obtain a solution using the Alternating Direction Method of Multipliers (ADMM) method. The ADMM formulation is detailed in Section 6.1

Two, A Lipschitz upper bound is imposed as a constraint during the training process. We refer to this technique as Imposed Lipschitz (or Lip-imp) method.

The optimisation objective for Lip-imp method is formulated as below:

$$\min_{W,\tilde{W}} \mathcal{L}(W) + 1_{Ml}(L^2_{des}, \tilde{W}), s.t., W = \tilde{W}$$

Here, L_{des} is the target Lipschitz constant for the network.

Similarly to the first method, we use ADMM to solve the above optimisation problem, the only difference being we set $L = L_{des}$ in the ADMM step in Eq 18.

6.1. ADMM formulation

The ADMM methods uses an augmented Lagrangian dual formulation as below

$$\begin{aligned} \mathcal{L}_p(W, \tilde{W}, L^2, Y) = \mathcal{L}(W) + \mu L^2 + 1_{Ml}(L^2, \tilde{W}) \\ + \langle Y, W - \tilde{W} \rangle + \frac{\rho}{2} ||W - \tilde{W}||^2 \end{aligned}$$

Here, Y is the Lagrange multiplier and ρ is the penalisation factor for the $W = \tilde{W}$ constraint.

The update steps of ADMM are as follows

$$W_{k+1} = arg \min_{W} \mathcal{L}_p(W, \tilde{W}_k, L^2_k, Y_k) \tag{17}$$

$$(L^2_{k+1}, \tilde{W}_{k+1}) = arg \min_{L^2,\tilde{W}} \mathcal{L}_p(W_{k+1}, \tilde{W}, L^2, Y_k) \tag{18}$$

$$Y_{k+1} = Y_k + \rho(W_{k+1} - \tilde{W}_{K+1}) \tag{19}$$

Here, Eq 17 is a non-convex multi-step optimisation problem and it is solved using the familiar stochastic gradient descent (SGD), and Eq 18 is an SDP optimisation problem which is solved using a standard SDP solvers like MOSEK ApS (2019).

TABLE I
CEL, ACCURACY, LIPSCHITZ BOUNDS, TRAINING TIMES FOR 2D AND MNIST EXPERIMENT

	CEL	Accuracy	L	L_{triv}	$\bar{t}_{\text{tr}}$ (loss/Lip step)
Nom-NN	0.15	96.0%	339.1	710.5	19,0s
L2-NN	0.42	80.5%	56.7	109.5	28.4s
Lip-NN	0.37	88.7%	49.4	134.2	164.9s (41.3s/0.63s)
Nom-NN	0.10	96.5%	140.6	879.1	202.8s
L2-NN	0.27	90.4%	15.3	23.2	225.5s
Lip-NN	0.21	96.3%	13.2	24.0	4632s (854s/521s)

6.2. Results

In Table 1, We take a look at the accuracy, Lipschitz bound and time complexity. The Lip-reg method is used to train a robust neural network, referred to a "Lip-NN" in the table. The results are compared against a non-robustly trained neural network, referred to as "Nom-NN", and a l_2 regularised neural network, referred to as "$L2$-NN", baselines.

We observe that the cost we pay in terms of time complexity is substantially higher for Lip-NN as compared to other methods, however, Lip-NN has a better Lipschitz bound. $L2$-NN is an interesting case, it is able to obtain a similarly small Lipschitz bound as Lip-NN but at a significantly lower time complexity, however, it is important to note that the accuracy of Lip-NN is significantly higher than L2-NN. The payoff for the additional time complexity of Lip-NN is in the form of better accuracy.

7. Conclusion

Robust learning is a crucial topic in machine learning. Identifying and defending against adversarial attacks is vital for many neural network based systems. We study in detail Pauli et al. (2022), Fazlyab et al. (2019a), Wong and Kolter (2018), Madry et al. (2017b), Szegedy et al. (2013)

In this regards, we defined and discussed the robust learning problem in this project. We talked about the problem of robust learning and described a few ways to solve them. We also discussed Lipschitz's constant as a robustness certification. Even with the current progress in the topic, the solutions for robust training are still limited, mostly available for deep-ReLU based classifiers. Also, the robustness certifications are limited. Even for a sophisticated methods like Lipschitz's constant estimates are weak bounded. On top of that problem itself is computationally expensive eg. Lip-SDP scales with the size of neural network. Future works include (i). Lipschitz's estimate for non-ReLU based network. Currently they are only limited to classifiers. For regression methods, slope-restricted activation relaxation may not work. (ii). Efficient robust training method. LP-train is fairly limited to ReLU activations. PGD-train is more generalized but is computationally very slow. (iii). Combining Lipschitz's estimate and Robust training for non-ReLU classifiers/regression model.

References

MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. URL http://docs.mosek.com/9.0/toolbox/index.html.

Patrick L. Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators, 2019. URL https://arxiv.org/abs/1903.01014.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019a.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks, 2019b. URL https://arxiv.org/abs/1906.04893.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017a. URL https://arxiv.org/abs/1706.06083.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017b.

Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgower. Training robust neural networks using lipschitz bounds. *IEEE Control Systems Letters*, 6:121–126, 2022. doi: 10.1109/lcsys.2021.3050444. URL https://doi.org/10.1109%2Flcsys.2021.3050444.

Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation, 2018. URL https://arxiv.org/abs/1805.10965.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope, 2017. URL https://arxiv.org/abs/1711.00851.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International conference on machine learning*, pages 5286–5295. PMLR, 2018.