Towards Certificated Model Robustness Against Weight Perturbations

Tsui-Wei Weng*,[1] Pu Zhao*,[2] Sijia Liu,[3] Pin-Yu Chen,[3] Xue Lin,[2] Luca Daniel[1]
[1]Massachusetts Institute of Technology, Cambridge, MA 02139
[2]Northeastern University, Boston, MA 02115
[3]MIT-IBM Watson AI Lab, IBM Research, Yorktown Heights, NY 10598
twweng@mit.edu, zhao.pu@husky.neu.edu, sijia.liu@ibm.com,
pin-yu.chen@ibm.com, xue.lin@northeastern.edu, dluca@mit.edu

Abstract

This work studies the sensitivity of neural networks to weight perturbations, firstly corresponding to a newly developed threat model that perturbs the neural network parameters. We propose an efficient approach to compute a certified robustness bound of weight perturbations, within which neural networks will not make erroneous outputs as desired by the adversary. In addition, we identify a useful connection between our developed certification method and the problem of weight quantization, a popular model compression technique in deep neural networks (DNNs) and a 'must-try' step in the design of DNN inference engines on resource constrained computing platforms, such as mobiles, FPGA, and ASIC. Specifically, we study the problem of weight quantization – weight perturbations in the non-adversarial setting – through the lens of certificated robustness, and we demonstrate significant improvements on the generalization ability of quantized networks through our robustness-aware quantization scheme.

Introduction

Although deep neural networks (DNNs) have achieved human-level performance in many learning tasks, intricate adversarial examples have been shown to exist in DNNs (Szegedy et al. 2014; Moosavi-Dezfooli, Fawzi, and Frossard 2016; Chen et al. 2018; Zhao et al. 2019a). An ever-increasing amount of research effort has been devoted to implementing adversarial attacks in various applications (Athalye, Carlini, and Wagner 2018; Carlini and Wagner 2017; Papernot et al. 2016a; Song et al. 2018; Carlini and Wagner 2018), developing defense methods ranging from heuristic methods to provable defenses (Papernot et al. 2016b; Liu et al. 2018; Madry et al. 2018; Kolter and Wong 2018; Liu et al. 2019), as well as efficient verification of neural networks against adversarial examples (Hein and Andriushchenko 2017; Weng et al. 2018b; 2018a; Gehr et al. 2018; Boopathy et al. 2019) and random noises (Weng et al. 2019) in image classification as well as in natural language processing (Ko et al. 2019) and reinforcement learning (Wang, Weng, and Daniel 2019). Different from above work on studying the problem of robustness against *input perturbations*, this work aims to evaluate the sensitivity of DNNs to *weight perturbations*.

*Equal contribution in alphabetical order

Weight perturbations of DNNs are of realistic significance. First, a new threat model of weight perturbations was proposed by (Liu et al. 2017; Zhao et al. 2019b), which showed that the so-called *fault sneaking/injection attack* can enforce DNN to misclassify some natural input images into target labels by slightly modifying weights at a single layer, while maintaining the classification of unspecified input images intact. This implies that the outputs of DNNs are also sensitive to weight perturbations. Moreover, there also exist weight perturbations in the non-adversarial setting. For example, *weight quantization* (Zhou et al. 2017; Leng et al. 2018; Ren et al. 2019), a major DNN model compression technique commonly utilized by industry for DNN acceleration/implementation, induces weight perturbations by replacing full floating-point precision weights with fixed-point lower precision weights. It is even well supported in GPUs and mobile devices, e.g., PyTorch (Paszke et al. 2017) in NVIDIA GPUs and TensorFlow Lite (Abadi et al. 2016) for mobile devices. However, such direct mapping from full precision weights of DNNs into quantized weights could result in significant generalization error (Sheng et al. 2018).

Different from the robustness issue caused by input perturbations, weight perturbations focus on DNN models for natural (unperturbed) examples rather than adversarial examples. If training and testing samples stem from the same distribution, evaluating the model robustness against weight perturbations in the training dataset (namely, sensitivity of training accuracy to weight perturbations) is able to provide informative guidelines on the generalization ability of the weight-perturbed network (e.g., weight-quantized network). Thus, both *fault injection attack* and *weight quantization* motivate us to study the problem of model robustness against weight perturbations.

Contributions. First, we formulate the certificate problem of model robustness against weight perturbations. The solution to this problem provides the *certified weight perturbation region* such that DNNs will maintain the accuracy if weight perturbations are within that region.

Second, we find provable non-trivial lower bounds on the exact certified weight perturbation region in two scenarios: a) single-layer perturbation and b) multi-layer perturbation. We empirically show that the certified lower bound provides a reasonable assessment on the practical model robustness against fault injection (Liu et al. 2017; Zhao et al. 2019b).

Third, we propose a new design of weight quantization by leveraging the statistics obtained from the certified weight perturbation region. This leads to a robust-aware quantization scheme, for which we show it can be efficiently achieved via alternating direction method of multipliers (ADMM) (Boyd et al. 2011). The resulting quantized network yields a significant improvement on its generalization ability compared to other quantization methods which neglect the effect of weight perturbations on the training procedure.

Related works. A line of work relevant to ours is formal verification of neural networks (Liu et al. 2019), which provides robustness guarantee that any input perturbation within a neighborhood of the natural example cannot fool the classifier's top-1 prediction. In (Katz et al. 2017; Ehlers 2017; Bunel et al. 2018; Dutta et al. 2017), satisfiability modulo theory (SMT) or mixed-integer programming (MIP) based methods provided exact robustness certificate with respect to the input perturbation strength. However, these approaches suffer from the scalability issue due to high computational complexity. Moreover, the work (Raghunathan, Steinhardt, and Liang 2018; Dvijotham et al. 2018; Wong and Kolter 2017; Weng et al. 2018a; 2018b; Wong et al. 2018) relaxed the exact verification problem by over-approximating the output space of a network given a neighborhood near the natural input. Such a relaxation leads to fast computation in the verification process but only proves a lower bound of exact robustness guarantees. Our paper is extended from the second line of work on certification of networks but with the main difference on the threat model: weight perturbations rather than input perturbations. Besides rigorously certifying network robustness, the work (Cheney, Schrimpf, and Kreiman 2017) empirically showed the robustness of convolutional neural networks (CNNs) against weight perturbations.

Different from existing literature, our work on certified robustness extended from input to weight perturbations is non-trivial, since the latter could be coupled at multiple layers. Most importantly, we provide a use case, design of weight quantization scheme, showing that robustness against weight perturbations matters even in the non-adversarial setting. It is worth mentioning that different from existing weight quantization work (Leng et al. 2018; Park, Ahn, and Yoo 2017; Zhou et al. 2017; Lin, Talathi, and Annapureddy 2016; Wu et al. 2016; Rastegari et al. 2016; Hubara et al. 2016), our work provides a solution through the leans of formal verification of model robustness and has the potential to be integrated with well-developed weight quantization frameworks (Paszke et al. 2017; Abadi et al. 2016).

Problem Setup

Notations. Throughout the paper, unless specified otherwise we use the following notations. Let $(\mathbf{x}, c)$ denote a pair of example $\mathbf{x}$ and class label c. For a K-layer neural network, let n_k, $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$, $\mathbf{b}^{(k)} \in \mathbb{R}^{n_k}$ denote the number of neurons, the weight matrix and the bias vector at layer k, respectively. We use the superscript (k) to indicate the variable associated with layer k. We also define $\mathbf{W} := \{\mathbf{W}^{(k)}\}_{k=1}^K$ and $\mathbf{b} := \{\mathbf{b}^{(k)}\}_{k=1}^K$ to denote the vector/matrix/set containing all variables indexed by k. And we use $[K]$ to denote the integer set $\{1, 2, \ldots, K\}$. Let $f(\mathbf{x}; \mathbf{W}, \mathbf{b}) \in \mathbb{R}^{n_K}$ be a neural network function with respect to the input $\mathbf{x}$ for n_K output classes. Here we refer f as the logit layer. The softmax layer can be safely discarded in our analysis due to its monotonicity. We use $f_j(\mathbf{x}; \mathbf{W}, \mathbf{b})$ (or simply f_j) to denote the j-th class output of the neural network. Let $\tilde{\Phi}^{(k)}(\mathbf{x}) \in \mathbb{R}^{n_k}$ and $\Phi^{(k)}(\mathbf{x}) \in \mathbb{R}^{n_k}$ denote the pre-activation and post-activation values of the k-th layer, respectively. And let $\sigma(\cdot)$ denote a non-linear element-wise activation function.

Neural network model. We focus on the K-layer fully-connected (FC) feedforward neural network with ReLU activation functions but the results can also be generalized to convolutional neural networks and general activations. The input-output relationship of the network is given by

$$\begin{aligned} &\Phi^{(k)}(\mathbf{x}) = \sigma\left(\tilde{\Phi}^{(k)}(\mathbf{x})\right), \\ &\tilde{\Phi}^{(k)}(\mathbf{x}) = \mathbf{W}^{(k)}\Phi^{(k-1)}(\mathbf{x}) + \mathbf{b}^{(k)}, \quad k \in [K], \end{aligned} \tag{1}$$

where $\Phi^{(0)}(\mathbf{x}) := \mathbf{x}$, and $f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \Phi^{(K)}(\mathbf{x})$. In the classification setting, the predicted class c is the class that has the largest output value: $\arg\max_j f_j$.

Weight perturbations. The problem of our interest is to provide a robustness certificate for a neural network when its weight parameters are perturbed. We define ℓ_∞-norm based weight perturbations as

$$\begin{aligned} \mathcal{B}(\mathbf{W}, \epsilon) = \Big\{ \hat{\mathbf{W}} \mid \hat{\mathbf{W}} = \{\hat{\mathbf{W}}^{(k)}\}, \\ \|\hat{\mathbf{W}}^{(k)} - \mathbf{W}^{(k)}\|_\infty \le \epsilon, \forall k \in [K] \Big\}, \end{aligned} \tag{2}$$

where ϵ is the perturbation radius, and $\hat{\mathbf{W}}^{(k)}$ denotes the perturbed weights against the original weights $\mathbf{W}^{(k)}$ at each layer. Given ϵ, verifying the neural network robustness (at input $\mathbf{x}$ with the true label c) against weight perturbation can be cast as the following optimization problem

$$\begin{array}{ll} \underset{\hat{\mathbf{W}}}{\text{minimize}} & f_c(\hat{\mathbf{W}}) - \max_{j \neq c} f_j(\hat{\mathbf{W}}) \\ \text{subject to} & \hat{\mathbf{W}} \in \mathcal{B}(\mathbf{W}, \epsilon), \end{array} \tag{3}$$

where we use the simplified notation $f_j(\mathbf{W})$ to highlight the dependency of classification on model weights by omitting $\mathbf{x}$ and $\mathbf{b}$. If the optimal value of problem (3) is *positive*, then the robustness of the neural network is certified under ϵ-tolerant weight perturbation at input $\mathbf{x}$.

Goal of robustness certification under weight perturbation: Finding the *largest* ϵ such that problem (3) has the *positive* optimal value.

We remark that problem (3) maintains the similar formulation of verifying neural network's robustness against *input perturbation*, e.g., recent works (Wong and Kolter 2017; Weng et al. 2018a; 2018b; Wong et al. 2018). However, none of them investigated the direction of certifying robustness against weight perturbation. Different from the input perturbation that generates an adversarial example, the effect of weight perturbation on network robustness is measured under the original input. At this sense, the certified perturbation region in terms of ϵ in (3) offers the new perspective on how sensitive the prediction accuracy of a well-trained network could be against weight perturbation. Moreover, since weights can be perturbed at multiple layers, the problem of weight perturbation suffers from a more complicated layer-wise coupling issue than that of certifying input perturbation.

Towards Certified Sensitivity of Weight Perturbations

In this section, we formally describe the idea of certified lower bound when the weights of neural networks are perturbed. We start from a simple 2-layer MLP example and provide general results in Theorem 3.1. Based on Theorem 3.1, we illustrate how to further generalize our results to multi-layer perturbation setting.

Certified lower bound: Single-layer weight perturbation

When the weight perturbation only occurs at a single layer (e.g. the N-th layer, $N \leq K$), we have the constraint:

$$\|\hat{\mathbf{W}}^{(N)} - \mathbf{W}^{(N)}\|_\infty \leq \epsilon,\ \hat{\mathbf{W}}^{(k)} = \mathbf{W}^{(k)},\ \text{if } k \neq N, k \in [K]. \tag{4}$$

Ideally, we would like to solve problem (3) *exactly* to get the maximum possible (exact) tolerance ϵ on the weight perturbation such that the top-1 prediction of a neural network classifier will not change. However, it has been shown that there does not exist a polynomial time algorithm to compute the *exact robustness* of neural networks (Katz et al. 2017). Hence, our goal is to find a non-trivial ϵ efficiently and this problem can be formulated as follows.

Let $f_c^L(\hat{\mathbf{W}})$ and $f_c^U(\hat{\mathbf{W}})$ be two linear functions of $\hat{\mathbf{W}}$ such that $f_c^L(\hat{\mathbf{W}}) \leq f_c(\hat{\mathbf{W}}) \leq f_c^U(\hat{\mathbf{W}})$ for all $\hat{\mathbf{W}}$, and let

$$\gamma_c^L = \min_{\hat{\mathbf{W}} \in \mathcal{B}(\mathbf{W},\epsilon)} f_c^L(\hat{\mathbf{W}}),\ \gamma_c^U = \max_{\hat{\mathbf{W}} \in \mathcal{B}(\mathbf{W},\epsilon)} f_c^U(\hat{\mathbf{W}}). \tag{5}$$

We can compute ϵ by solving the following problem:

$$\begin{array}{ll} \underset{\epsilon}{\text{maximize}} & \epsilon \\ \text{subject to} & \gamma_c^L - \gamma_j^U > 0, \forall j \neq c. \end{array} \tag{6}$$

Note that the constraint set $\gamma_c^L - \max_{j \neq c} \gamma_j^U > 0$ (namely, $\gamma_c^L - \gamma_j^U > 0, \forall j \neq c$) is more restricted than $f_c(\hat{\mathbf{W}}) - \max_{j \neq c} f_j(\hat{\mathbf{W}}) > 0$. Thus, the solution to problem (6) provides a certified lower bound on the maximum ϵ to ensure the positive objective value of problem (3). In fact, in the following, we will show that γ_c^L and γ_c^U can be computed analytically, and hence we are able to find the solution of (6) efficiently through bi-section on ϵ. Note that in the work (Weng et al. 2018a), the authors proposed an efficient algorithm *Fast-Lin* to compute a certified lower bound for neural networks with *input perturbation*, whereas in this work, we focus on *weight-perturbation* on the neural networks and show that it is also possible to derive certified lower bound for this problem setting. Furthermore, we show in Sec 4 and 5 that our technique are beneficial to developing a new weight quantization scheme with better generalization.

Neural network f is bounded by two linear functions $f_c^L(\hat{\mathbf{W}})$ and $f_c^U(\hat{\mathbf{W}})$. The core idea to deriving the linear bounds $f_c^L(\hat{\mathbf{W}})$, $f_c^U(\hat{\mathbf{W}})$ of a K-layer feed-forward neural network f is to apply linear upper and lower bound on each neuron's activation and consider the signs of associated weights. We start with a 2-layer network ($K = 2$) and then extend it to the general case. Suppose that the first layer weights are perturbed and we have (4) with $K = 1$. The j-th output of the network (with respect to $\hat{\mathbf{W}}^{(1)}$) is then given by

$$f_j(\hat{\mathbf{W}}^{(1)}) = \sum_{r \in [n_1]} W_{j,r}^{(2)} \sigma\left(\hat{\mathbf{W}}_{r,:}^{(1)}\mathbf{x} + b_r^{(1)}\right) + b_j^{(2)}, \tag{7}$$

where $W_{j,r}$ denotes the (j,r)-th entry of $\mathbf{W}$, and $\hat{\mathbf{W}}_{r,:}$ denotes the r-th row of $\hat{\mathbf{W}}$. Since $\mathbf{W}_{r,:}^{(1)} - \epsilon \leq \hat{\mathbf{W}}_{r,:}^{(1)} \leq \mathbf{W}_{r,:}^{(1)} + \epsilon$, the pre-activation $y_r^{(1)} := \hat{\mathbf{W}}_{r,:}^{(1)}\mathbf{x} + b_r^{(1)}$ at the 1-st layer is bounded by some constants $l_r^{(1)}$, and $u_r^{(1)}$, which are determined by the signs of $\mathbf{x}$ and the bounds of $\hat{\mathbf{W}}_{r,:}^{(1)}$.

Given $y_r^{(1)} \in [l_r^{(1)}, u_r^{(1)}]$, the non-linear activation function $\sigma(y_r^{(1)})$ has explicit linear bounds (Zhang et al. 2018) with slope and bias parameters $\{\alpha_{L,r}^{(1)}, \alpha_{U,r}^{(1)}\}$ and $\{\beta_{L,r}^{(1)}, \beta_{U,r}^{(1)}\}$ as follows:

$$\alpha_{L,r}^{(1)}(y_r^{(1)} + \beta_{L,r}^{(1)}) \leq \sigma(y_r^{(1)}) \leq \alpha_{U,r}^{(1)}(y_r^{(1)} + \beta_{U,r}^{(1)}). \tag{8}$$

If $l_r^{(1)} < 0 < u_r^{(1)}$, then $\alpha_{L,r}^{(1)} = \alpha_{U,r}^{(1)} = \frac{u_r^{(1)}}{u_r^{(1)} - l_r^{(1)}}$, $\beta_{L,r}^{(1)} = 0$, and $\beta_{U,r}^{(1)} = -l_r^{(1)}$; if $l_r^{(1)} \leq u_r^{(1)} \leq 0$, then all parameters are zeros; if $0 \leq l_r^{(1)} \leq u_r^{(1)}$, then $\alpha_{L,r}^{(1)} = \alpha_{U,r}^{(1)} = 1$ and $\beta_{L,r}^{(1)} = \beta_{U,r}^{(1)} = 0$. The equations (7) and (8) of the 2-layer network example imply two general rules:

1. The pre-activation bounds at the N-th layer are known *a priori* (since no weight prior to the N-th layer is perturbed), and thus we only need to perform bound propagation for $k > N$ layers;
2. The final layer bounds $f_c^L(\hat{\mathbf{W}})$ and $f_c^U(\hat{\mathbf{W}})$ can be computed via bound propagation. The idea is to compute the pre-activation bounds layer by layer (which is so-called bound propagation) analytically via Theorem 0.1. In Theorem 0.1, we show the analytic output bounds of neural networks when there exists single-layer ℓ_p-norm weight perturbation with $p \geq 1$.

Theorem 0.1 *Suppose that the N-th layer weights are perturbed in a K-layer neural network. Let $f : \mathbb{R}^{n_N \times n_{N-1}} \to \mathbb{R}^{n_K}$ denote the mapping from perturbed weights $\hat{\mathbf{W}}^{(N)}$ at the single layer N to predicted outputs at the final layer K. Then there exist two explicit functions $f_j^L : \mathbb{R}^{n_N \times n_{N-1}} \to \mathbb{R}$ and $f_j^U : \mathbb{R}^{n_N \times n_{N-1}} \to \mathbb{R}$ for class $\forall j \in [n_K]$, such that the following inequality holds*

$$f_j^L(\hat{\mathbf{W}}^{(N)}) \leq f_j(\hat{\mathbf{W}}^{(N)}) \leq f_j^U(\hat{\mathbf{W}}^{(N)}), \tag{9}$$

where $\|\hat{\mathbf{W}}_{s,:}^{(N)} - \mathbf{W}_{s,:}^{(N)}\|_p \leq \epsilon$ and $\hat{\mathbf{W}}^{(k)} = \mathbf{W}^{(k)}$ for $\forall k \neq N$, and $p \geq 1$. The closed forms of lower and upper bounds *in* (9) *are given by*

$$f_j^U(\hat{\mathbf{W}}^{(N)}) = \mathbf{\Lambda}_{j,:}^{(N-1)}\hat{\mathbf{W}}^{(N)}\Phi^{(N-1)}(\mathbf{x}) + \sum_{k=N}^{K}\mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}), \quad (10)$$

$$f_j^L(\hat{\mathbf{W}}^{(N)}) = \mathbf{\Omega}_{j,:}^{(N-1)}\hat{\mathbf{W}}\Phi^{(N-1)}(\mathbf{x}) + \sum_{k=N}^{K}\mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}), \quad (11)$$

where

$$\mathbf{\Lambda}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = K+1, \\ \mathbf{\Lambda}_{j,:}^{(k)} \odot \lambda_{j,:}^{(k-1)} & \text{if } k = N, \\ (\mathbf{\Lambda}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \lambda_{j,:}^{(k-1)} & \text{otherwise.} \end{cases}$$

$$\mathbf{\Omega}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = K+1, \\ \mathbf{\Omega}_{j,:}^{(k)} \odot \omega_{j,:}^{(k-1)} & \text{if } k = N, \\ (\mathbf{\Omega}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \omega_{j,:}^{(k-1)} & \text{otherwise.} \end{cases}$$

Here the matrices $\lambda^{(k)}, \omega^{(k)}, \mathbf{\Delta}^{(k)}, \mathbf{\Theta}^{(k)}$ *are functions of the linear bounding parameters* $\{\alpha_{L,i}^{(k)}, \alpha_{U,i}^{(k)}\}$ *and* $\{\beta_{L,i}^{(k)}, \beta_{U,i}^{(k)}\}$ *on each neuron* i *at layer* k*:*

$$\lambda_{j,i}^{(k)} = \begin{cases} \alpha_{U,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0, \\ \alpha_{L,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0, \\ 1 & \text{if } k = N-1. \end{cases}$$

$$\omega_{j,i}^{(k)} = \begin{cases} \alpha_{L,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0, \\ \alpha_{U,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0, \\ 1 & \text{if } k = N-1. \end{cases}$$

$$\mathbf{\Delta}_{i,j}^{(k)} = \begin{cases} \beta_{U,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0, \\ \beta_{L,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0, \\ 0 & \text{if } k = N-1. \end{cases}$$

$$\mathbf{\Theta}_{i,j}^{(k)} = \begin{cases} \beta_{L,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0, \\ \beta_{U,i}^{(k)} & \text{if } k \neq N-1,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0, \\ 0 & \text{if } k = N-1. \end{cases}$$

where $\odot$ *is the Hadamard product and* $\mathbf{e}_j \in \mathbb{R}^{n_K}$ *is a* j*-th basis vector.*

Proof. The proof on the input perturbation (Weng et al. 2018a; Zhang et al. 2018; Boopathy et al. 2019) can be adapted to weight perturbation in our case, where we have $f_j(\hat{\mathbf{W}}^{(N)}) \leq f_j^{U,K-1}(\hat{\mathbf{W}}^{(N)}) \leq f_j^{U,K-2}(\hat{\mathbf{W}}^{(N)}) \leq \ldots \leq f_j^{U,N+1}(\hat{\mathbf{W}}^{(N)}) = \tilde{\mathbf{W}}_{j,:}^{(N+1)}\sigma(\hat{\mathbf{W}}^{(N)}\Phi^{(N-1)} + \mathbf{b}^{(N)})$. The derivation of $\mathbf{\Lambda}_{j,:}^{(k-1)}, \mathbf{\Omega}_{j,:}^{(k-1)}$ from layers $N+1$ to $K-1$ is the same except for the N-th layer. For N-th layer, since the perturbation is now on $\hat{\mathbf{W}}^{(N)}$ rather than $\mathbf{x}$, we let $\mathbf{\Lambda}_{j,:}^{(N-1)} = \mathbf{\Lambda}_{j,:}^{(N)} \odot \lambda_{j,:}^{(N-1)}$. By using the same trick to decompose $\sigma(y)$ by the inequality (8) with associated sign of the equivalent matrix $\tilde{\mathbf{W}}_{j,:}^{(N+1)}$, we get the final upper bound (10). The lower bound $f_j^L(\hat{\mathbf{W}}^{(N)})$ can be derived similarly. □

Global output bounds γ_j^U and γ_j^L. Based on (10) and (11), we can further derive the global output bounds γ_j^U and γ_j^L, which are constants, determined by (5). Since f_j^L and f_j^U are two linear functions and $\hat{\mathbf{W}} \in \mathcal{B}(\mathbf{W}, \epsilon)$ is a convex norm constraint, the optimal value of problem (5) can be obtained by Holder's inequalities:

$$\gamma_j^U = \epsilon\|\mathbf{\Lambda}_{j,:}^{(N-1)}\|_1 \cdot \|\Phi^{(N-1)}(\mathbf{x})\|_q + \mathbf{\Lambda}_{j,:}^{(N-1)}\mathbf{W}^{(N)}\Phi^{(N-1)}(\mathbf{x}) + \sum_{k=N}^{K}\mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}), \quad (12)$$

$$\gamma_j^L = -\epsilon\|\mathbf{\Omega}_{j,:}^{(N-1)}\|_1 \cdot \|\Phi^{(N-1)}(\mathbf{x})\|_q + \mathbf{\Omega}_{j,:}^{(N-1)}\mathbf{W}^{(N)}\Phi^{(N-1)}(\mathbf{x}) + \sum_{k=N}^{K}\mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}), \quad (13)$$

where $1/q = 1 - 1/p$. As γ_j^U and γ_c^L are monotonically increasing and decreasing with respect to ϵ, respectively, we can solve problem (6) by the bisection search over ϵ, which renders the certified lower bound on network's robustness against weight perturbation.

As a final remark, our work is different from adversarial robustness against input perturbation in that f is a function of perturbed weight matrix $\hat{\mathbf{W}}^{(N)}$ bounded by two linear functions $f_j^U(\hat{\mathbf{W}}^{(N)}), f_j^L(\hat{\mathbf{W}}^{(N)})$ as opposed to a function of perturbed input $\mathbf{x}$. Interestingly, once the neuron's activation are linearly bounded, we can directly apply the similar idea in (Weng et al. 2018a; Zhang et al. 2018; Boopathy et al. 2019) to compute the bounds layer-by-layer with Theorem 0.1. Our results can also be directly extended to convolutional layers following (Boopathy et al. 2019).

Certified lower bound: Multi-layer perturbation

When there exists multi-layer weights perturbations, deriving the certified lower bound becomes much more involved as the weight perturbations will be coupled across layers. However, computing layer-wise bound for each neuron is still possible – we can integrate previous single-layer results with interval bound propagation as demonstrated below. Without loss of generality, assume that the weight matrices at the S-th layer and the N-th layer are both perturbed with $S < N < K$. Equations (12) and (13) can be directly used to compute the perturbation bounds at each neuron of layer S by viewing the $N-1$-th layer as the output layer. For $N-1 \leq i \leq K$, we have $\hat{l}_r^{(i)} \leq \Phi_r^{(i)} \leq \hat{u}_r^{(i)}$, where $\hat{u}_r^{(i)} = \sigma(u_r^{(i)})$, $\hat{l}_r^{(i)} = \sigma(l_r^{(i)})$. Let $k = N-1$, $p = \infty$ and $q = 1$, we then have:

- if $i = k$,

$$u_j^{(i+1)} = |\mathbf{W}_{j,:}^{(i+1)}|\frac{\hat{u}^{(i)} - \hat{l}^{(i)}}{2} + \mathbf{W}_{j,:}^{(i+1)}\frac{\hat{u}^{(i)} + \hat{l}^{(i)}}{2} + \mathbf{b}_j^{(k+1)} + \epsilon\|\frac{\hat{u}^{(i)} - \hat{l}^{(i)}}{2}\|_q + \epsilon\|\frac{\hat{u}^{(i)} + \hat{l}^{(i)}}{2}\|_q$$

$$l_j^{(i+1)} = -|\mathbf{W}_{j,:}^{(i+1)}|\frac{\hat{u}^{(i)} - \hat{l}^{(i)}}{2} + \mathbf{W}_{j,:}^{(i+1)}\frac{\hat{u}^{(i)} + \hat{l}^{(i)}}{2} + \mathbf{b}_j^{(k+1)} - \epsilon\|\frac{\hat{u}^{(i)} - \hat{l}^{(i)}}{2}\|_q + \epsilon\|\frac{\hat{u}^{(i)} + \hat{l}^{(i)}}{2}\|_q$$

- if $i > k$,
  $u_j^{(i+1)} = |\mathbf{W}_{j,:}^{(i+1)}| \frac{\hat{u}^{(i)} - \hat{l}^{(i)}}{2} + \mathbf{W}_{j,:}^{(i+1)} \frac{\hat{u}^{(i)} + \hat{l}^{(i)}}{2} + \mathbf{b}_j^{(i+1)}$
  $l_j^{(i+1)} = -|\mathbf{W}_{j,:}^{(i+1)}| \frac{\hat{u}^{(i)} - \hat{l}^{(i)}}{2} + \mathbf{W}_{j,:}^{(i+1)} \frac{\hat{u}^{(i)} + \hat{l}^{(i)}}{2} + \mathbf{b}_j^{(i+1)}$

and the network output $f(\hat{\mathbf{W}}^{(S)}, \hat{\mathbf{W}}^{(N)})$ is bounded by $l_j^{(K)} \le f_j(\hat{\mathbf{W}}^{(S)}, \hat{\mathbf{W}}^{(N)}) \le u_j^{(K)}$. For simplicity, we present the above analysis with same ϵ and without bias perturbation, but our analysis can be extended to the case where ϵ_i associated to the i-th layer and when biases $\mathbf{b}$ are perturbed.

Weight Quantization with Perturbation Certificate

In this section, we revisit the problem of weight quantization, commonly used for network compression, through the lens of certificated robustness against weight perturbation. We propose a unified quantization framework with perturbation certificate by leveraging alternating direction method of multipliers (ADMM).

Weight quantization. Given a finite number of bits, the goal is to discretize a model's weights but to preserve its accuracy. At d_k-bit quantization of layer k, we use 1 bit to represent zero value, and the remaining $d_k - 1$ bits to represent at most $2^{d_k - 1}$ different values with distance $q^{(k)}$ (here we follow the convention from (Zhou et al. 2017; Leng et al. 2018)).The quantization of continuous weights $\mathbf{W}^{(k)} = \mathbf{C}^{(k)}$ is given by

$$\hat{\mathbf{W}}^{(k)} \in \{ -2^{d_k-2} q^{(k)}, \ldots, -2q^{(k)}, -q^{(k)}, 0, \\ q^{(k)}, 2q^{(k)}, \ldots, 2^{d_k-2} q^{(k)} \}, \ \forall k \in [K] \quad (14)$$

Here we use the notation of perturbed weights $\hat{\mathbf{W}}^{(k)}$ to represent the weights after quantization. We remark that the set of quantized values in (14) can easily be encoded using binary bits and implemented in hardware. For example, by storing $q^{(k)}$, we can express $\{-2q^{(k)}, -q^{(k)}, 0, q^{(k)}, 2q^{(k)}\}$ as $\{-2, -1, 0, 1, 2\}$.

Given the training loss f and the number of bits $\{d_k\}$, the problem of weight quantization determines $\{q^{(k)}, \hat{\mathbf{W}}^{(k)}\}$ by solving the optimization problem

$$\underset{\{q^{(k)}, \hat{\mathbf{W}}^{(k)}\}}{\text{minimize}} \ f(\{\hat{\mathbf{W}}^{(k)}\}), \ \text{subject to (14)}. \quad (15)$$

Note that a quantized network may suffer from the generalization issue. However, any weight perturbation within the certified region found by Theorem 0.1 will not cause misclassification. Thus, from the perspective of weight perturbation, one can integrate the certified region with problem (15) to reduce the generalization error caused by quantization. We call the resulting quantization problem *certified weight quantization*, which is given by

$$\begin{array}{ll} \underset{\{q^{(k)}, \hat{\mathbf{W}}^{(k)}\}}{\text{minimize}} & f(\{\hat{\mathbf{W}}^{(k)}\}) \\ \text{subject to} & (14), \ \|\hat{\mathbf{W}}^{(k)} - \mathbf{C}^{(k)}\|_\infty \le \epsilon_c^{(k)}, \ \forall k, \end{array} \quad (16)$$

where $\epsilon_c^{(k)}$ is a threshold chosen by the radius of the certified perturbation region with respect to (w.r.t.) continuous weights $\mathbf{C}^{(k)}$ at layer k. Given a training sample $\mathbf{x}$, the solution to problem (6) provides $\epsilon_c^{(k)}(\mathbf{x})$. The sample-independent threshold $\epsilon_c^{(k)}$ can then be set as the average or other percentiles of $\{\epsilon_c^{(k)}(\mathbf{x})\}$ over multiple samples. Our experiments will show that the empirical improvement on the generalization error of quantized networks is significant by incorporating the certification constraints on weight perturbation.

We finally remark that at the first glance, one may think that problem (16) would yield worse training loss compared to problem (15), since the former has additional constraints. However, due to non-convexity, it is difficult to solve problem (15) and (16) at the *global* optimality. The story is then different when comparing a *local* optimal solution to problem (16) with a *local* optimal solution to problem (15). Suppose that training and testing samples obey the same underlying data distribution, then the proposed robustness certificate can drive the designed sub-optimal quantizer toward better generalization capability. Indeed, our experiments show that the introduction of certificate constraints reduces both training and generalization errors; see Figure 2.

Certified weight quantization via ADMM. Upon defining $\hat{\mathbf{W}}^k = q^{(k)} \hat{\mathbf{V}}^k$, problem (16) can be rewritten as

$$\begin{array}{ll} \underset{\{q^{(k)}, \hat{\mathbf{V}}^{(k)}\}}{\text{minimize}} & f(\{q^{(k)} \hat{\mathbf{V}}^{(k)}\}) \\ \text{subject to} & \hat{\mathbf{V}}^{(k)} \in \mathcal{D}^{(k)}, \ \forall k, \\ & \|q^{(k)} \hat{\mathbf{V}}^k - \mathbf{C}^{(k)}\|_\infty \le \epsilon_c^{(k)}, \ \forall k \end{array} \quad (17)$$

where $\mathcal{D}^{(k)} := \{-2^{d_k-2}, \ldots, -2, -1, 0, 1, 2, \ldots, 2^{d_k-2}\}$. Note that solving problem (17) is challenging since certificate constraints involve bilinear terms $\{q^{(k)} \hat{\mathbf{V}}^k\}$. We propose an ADMM-based problem formulation and optimization algorithm to obtain a solution.

We begin by introducing an auxiliary variable $\hat{\mathbf{G}}^k$ together with $\hat{\mathbf{G}}^k = \hat{\mathbf{V}}^k$ and reformulate (17) by lending itself to the application of ADMM,

$$\begin{array}{ll} \underset{\{q^{(k)}, \hat{\mathbf{V}}^{(k)}, \hat{\mathbf{G}}^{(k)}\}}{\text{minimize}} & f(\{q^{(k)} \hat{\mathbf{G}}^{(k)}\}) + \mathcal{I}_1(\{\hat{\mathbf{V}}^{(k)}\}) \\ & + \mathcal{I}_2(\{q^{(k)} \hat{\mathbf{G}}^{(k)}\}) \\ \text{subject to} & \hat{\mathbf{G}}^{(k)} = \hat{\mathbf{V}}^{(k)}, \ \forall k, \end{array} \quad (18)$$

where $\mathcal{I}_1$ and $\mathcal{I}_2$ are indicator functions encoding the constraints a) $\hat{\mathbf{V}}^{(k)} \in \mathcal{D}^{(k)}$ for all k, and b) $\|q^{(k)} \hat{\mathbf{G}}^{(k)} - \mathbf{C}^{(k)}\|_\infty \le \epsilon_c^{(k)}$ for all k. The key difference from the ADMM formulation in (Leng et al. 2018) is that we impose the auxiliary variable w.r.t. $\hat{\mathbf{V}}^{(k)}$ rather than $\hat{\mathbf{W}}^{(k)}$ so that the variable $q^{(k)}$ is explicitly included in our formulation.

We then introduce the augmented Lagrangian function of (18) that will be alternatively minimized in ADMM,

$$\begin{aligned} & \mathcal{L}(\{q^{(k)}\}, \{\hat{\mathbf{V}}^{(k)}\}, \{\hat{\mathbf{G}}^k\}, \{\hat{\mathbf{U}}^{(k)}\}) \\ & = f(\{q^{(k)} \hat{\mathbf{G}}^{(k)}\}) + \mathcal{I}_1(\{\hat{\mathbf{V}}^{(k)}\}) + \mathcal{I}_2(\{q^{(k)} \hat{\mathbf{G}}^{(k)}\}) \\ & + \sum_{k=1}^{K} (\hat{\mathbf{U}}^{(k)})^T (\hat{\mathbf{G}}^{(k)} - \hat{\mathbf{V}}^{(k)}) + \frac{\rho}{2} \sum_{k=1}^{K} \|\hat{\mathbf{G}}^{(k)} - \hat{\mathbf{V}}^{(k)}\|_F^2, \end{aligned} \quad (19)$$

where $\rho > 0$ is a regularization parameter, $\hat{\mathbf{U}}^{(k)}$ are Lagrangian multipliers w.r.t. equality constraints of (18), and $\| \cdot \|_F$ denotes the Frobenius norm.

For ease of notation, let $\hat{\mathbf{G}}$, $\mathbf{q}$, $\hat{\mathbf{V}}$ and $\mathbf{U}$ denote the set of variables $\{\hat{\mathbf{G}}^{(k)}\}$, $\{q^{(k)}\}$, $\{\hat{\mathbf{V}}^{(k)}\}$ and $\{\hat{\mathbf{U}}^{(k)}\}$. The main steps of ADMM are alternatively minimizing (19) over *two blocks* of variables, $\hat{\mathbf{G}}$ and $\{\mathbf{q}, \hat{\mathbf{V}}\}$. That is, ADMM at the t-th iteration is given by

$$\hat{\mathbf{G}}(t+1) = \arg\min_{\hat{\mathbf{G}}} \mathcal{L}(\mathbf{q}(t), \hat{\mathbf{V}}(t), \hat{\mathbf{G}}, \mathbf{U}(t)), \quad (20)$$

$$\{\mathbf{q}(t+1), \hat{\mathbf{V}}(t+1)\} = \arg\min_{\mathbf{q},\hat{\mathbf{V}}} \mathcal{L}(\mathbf{q}, \hat{\mathbf{V}}, \hat{\mathbf{G}}(t+1), \mathbf{U}(t)), \quad (21)$$

where $\mathbf{U}(t+1) = \mathbf{U}(t) + \rho(\hat{\mathbf{G}}(t+1) - \hat{\mathbf{V}}(t+1))$, and we initialize ADMM with specified $\mathbf{q}(0)$,$\hat{\mathbf{V}}(0)$ and $\mathbf{U}(0)$. We note that problem (21) is decomposable w.r.t. $\mathbf{q}$ and $\hat{\mathbf{V}}$; see equivalent ADMM steps in Proposition 1.

Proposition 1 *The ADMM subproblems* (20) *and* (21) *can be equivalently transformed into a)* $\hat{\mathbf{G}}$*-minimization step, b)* $\mathbf{q}$*-minimization step and c)* $\hat{\mathbf{V}}$*-minimization step. That is,*

$\underline{\hat{\mathbf{G}}\textit{-minimization step:}}$ $\hat{\mathbf{G}}(t+1)$ *in* (20) *is given by the solution of the problem*

$$\begin{array}{ll} \underset{\hat{\mathbf{G}}}{\textit{minimize}} & f(\hat{\mathbf{G}}; \mathbf{q}(t)) + \frac{\rho}{2}\|\hat{\mathbf{G}} - \mathbf{A}\|_F^2 \\ \textit{subject to} & (1/q^{(k)})\check{\mathbf{D}}^{(k)} \leq \hat{\mathbf{G}}^{(k)} \leq (1/q^{(k)})\hat{\mathbf{D}}^{(k)}, \ \forall k, \end{array} \quad (22)$$

where $f(\hat{\mathbf{G}}; \mathbf{q}(t))$ *represents* f *w.r.t. variables* $\hat{\mathbf{G}}$ *under given values* $\mathbf{q}(t)$, $\check{\mathbf{D}}^{(k)} := \mathbf{C}^{(k)} - \epsilon_c^{(k)}\mathbf{I}$, *and* $\hat{\mathbf{D}}^{(k)} := \epsilon_c^{(k)}\mathbf{I} + \mathbf{C}^{(k)}$.

$\underline{\mathbf{q}\textit{-minimization step:}}$ $\mathbf{q}(t+1)$ *in* (21) *is given by the solution of the problem*

$$\begin{array}{ll} \underset{\mathbf{q}}{\textit{minimize}} & f(\mathbf{q}; \hat{\mathbf{G}}(t+1)) \\ \textit{subject to} & \max\{\hat{a}_1^{(k)}, \hat{a}_2^{(k)}\} \leq q^{(k)} \leq \min\{\check{a}_1^{(k)}, \check{a}_2^{(k)}\}, \ \forall k. \end{array} \quad (23)$$

Let $I_{k,+}$ *and* $I_{k,-}$ *denote the index set of positive and negative elements in* $\hat{\mathbf{G}}^{(k)}(t+1)$, *respectively. And let* $\mathit{1}[\mathbf{X}]_I$ *denote the sub-matrix of* $\mathbf{X}$ *selected by the index set* I, *and* $\cdot/\cdot$ *and* $\max\{\cdot\}$ *denote the element-wise division and maximum operation, respectively, Then* $\hat{a}_1^{(k)}$, $\hat{a}_2^{(k)}$, $\check{a}_1^{(k)}$, *and* $\check{a}_2^{(k)}$ *in* (23) *are given by*

$$\begin{aligned} \hat{a}_1^{(k)} &:= \max\{[\hat{\mathbf{D}}^{(k)}]_{I_{k,+}} / [\hat{\mathbf{G}}^{(k)}(t+1)]_{I_{k,+}}\}, \\ \hat{a}_2^{(k)} &:= \max\{[\hat{\mathbf{D}}^{(k)}]_{I_{k,-}} / [\hat{\mathbf{G}}^{(k)}(t+1)]_{I_{k,-}}\}, \\ \check{a}_1^{(k)} &:= \min\{[\hat{\mathbf{D}}^{(k)}]_{I_{k,+}} / [\hat{\mathbf{G}}^{(k)}(t+1)]_{I_{k,+}}\}, \\ \check{a}_2^{(k)} &:= \min\{[\check{\mathbf{D}}^{(k)}]_{I_{k,-}} / [\hat{\mathbf{G}}^{(k)}(t+1)]_{I_{k,-}}\}. \end{aligned}$$

$\underline{\hat{\mathbf{V}}\textit{-minimization step:}}$ $\hat{\mathbf{V}}(t+1)$ *in* (21) *is given by*

$$\hat{V}_{ij}^{(k)}(t+1) = \begin{cases} \lfloor B_{ij} \rceil & B_{ij} \in [-2^{d_k-2}, 2^{d_k-2}] \\ -2^{d_k-2} & B_{ij} < -2^{d_k-2} \\ 2^{d_k-2} & B_{ij} > 2^{d_k-2}, \end{cases} \quad (24)$$

where $\hat{V}_{ij}^{(k)}(t+1)$ *denotes the* (i,j)*-th element of* $\hat{\mathbf{V}}^{(k)}(t+1)$, $\lfloor x \rceil$ *represents the nearest integer to* x, *and* $\mathbf{B} := \hat{\mathbf{G}}(t+1) - (1/\rho)\mathbf{U}(t)$.

Proof: See Appendix[1]. □

[1] The appendix and code are available at https://github.com/lilyweng/Quantization.

Comparison with (Leng et al. 2018). We note that (Leng et al. 2018) also uses ADMM to solve a similar problem. The key difference of the two ADMM-based solutions is described as follows. 1) The introduced auxiliary variables in ADMM are different. In (Leng et al. 2018), the auxiliary variables are copies of network weights, ignoring the effect of $q^{(k)}$. However, in (18), the auxiliary variables $\hat{\mathbf{G}}^k$ are weights separated from the quantization intervals q^k. 2) The operator splitting is different. We perform a two-block splitting over the variables $\hat{G}^k$ and $\{q^k, \hat{\mathbf{V}}^k\}$, and show that the resulting second-block subproblem can be further decomposed into easier problems (23) and (24). In contrast, a sub-optimal iterative optimization method was internally used in (Leng et al. 2018) to solve a more involved splitting problem.

In Proposition 1, each subproblem can be efficiently handled by standard optimization solvers. In (22)-(23), since the constraint sets are simple box constraints, projected gradient descent methods can be used to find solutions $\hat{\mathbf{G}}(t+1)$ and $\mathbf{q}(t+1)$. In (24), we have the closed-form expression of the solution $\hat{\mathbf{V}}(t+1)$. We also note that ADMM is convergent and reaches a sub-linear convergence rate for nonconvex and nonsmooth stochastic optimization (Huang and Chen 2018).

Experiments

We demonstrate the effectiveness of our approach under two applications: a) weight quantization described in Sec. and b) model robustness against fault sneaking attack (Zhao et al. 2019b). To align with our theoretical results, we perform experiments under multilayer perceptron (MLP) models of various numbers of layers[1]. The performance is evaluated under 4 datasets, MNIST, MNIST-fashion, SVHN, and CIFAR-10.

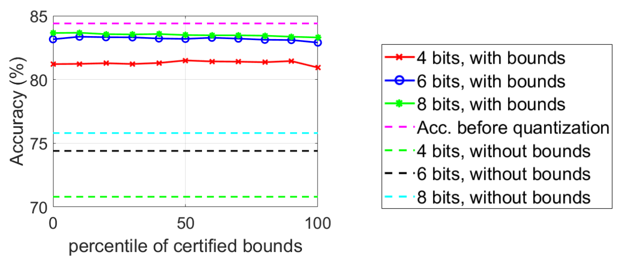

Figure 1: Testing accuracy of quantized 8-layer MLP (4, 6, and 8 bits per layer) on SVHN versus the certification constraint parameter chosen by different percentiles of certified weight perturbation lower bounds over 100 training images.

Robustness-aware weight quantization. We consider MLP models of 2, 4, 6, 8 and 10 layers, each of which is quantized using 4, 6, and 8 bits. Figure 1 presents the testing accuracy of the quantized 8-layer MLP on SVHN versus the choice of the certification constraint parameter $\epsilon_c^{(k)}$ of problem (16). Here we set $\epsilon_c^{(k)}$ as a percentile of certified robustness bounds (6) over 100 training images. For comparison, we also present the performance of weight quantization by solving problem (15) without imposing certification constraints. We see that the use of certification bounds sig-

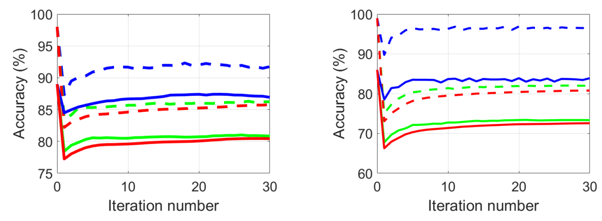

Figure 2: Training/testing accuracy of quantization with/without certification constraints. left) MNIST-Fashion. middle) SVHN. Dashed lines denote training accuracy and solid lines represent test accuracy. Blue lines show the proposed ADMM method with bounds, and green lines show the proposed ADMM method without bounds. Red lines show the LB method without bounds.

nificantly boosts the testing accuracy of quantized networks (around 10% improvement) and approaches the testing accuracy without quantization. Moreover, we observe that the improved performance is insensitive to the choice of $\epsilon_c^{(k)}$ except a slight degradation when $\epsilon_c^{(k)}$ is greater than the 90 percentile of certification bounds. This is not surprising since the performance of weight quantization without imposing certification constraints is worst, corresponding to the extreme case $\epsilon_c^{(k)} \to \infty$. In the following experiments, unless specified otherwise, we choose $\epsilon_c^{(k)}$ as 50 percentile of certified robustness bounds. Additional results on CIFAR-10 and other model architectures are provided in Table A1 of Appendix. We note that the test accuracy of quantization becomes worse as $\epsilon_c^{(k)}$ becomes much larger, e.g. the use of 2,5,10 times of 100-th percentile certification bounds yields worse test accuracy 82.7%,78.6%,76.1% for 6 layer MLP with 6 bits quantization, whereas the use of 0, 100th percentile gives 83.4%, 83%. More results are shown in the appendix.

In Figure 2, we compare our proposed method with the ADMM-based low-bits (LB) quantization method (Leng et al. 2018) in terms of training/testing accuracy versus ADMM iterations under datasets MNIST-fashion and SVHN. Recall that the LB method was designed to solve problem (15) without introducing certification constraints. All the methods are initialized from the same pre-trained model of continuous weights. It can be seen from training accuracy that our method converges to a better local optimal solution than the LB method even in the absence of certification constraints. When information on certified robustness bounds is considered, both training and testing accuracy are significantly improved, consistent with results in Figure 1 and Table A1.

Model robustness against fault sneaking attack (Zhao et al. 2019b). It was shown in (Zhao et al. 2019b) that slightly perturbing model weights at a single layer is capable of misclassfying a specific set of natural images toward target labels but keeping classification of unspecified input images intact. The corresponding threat model is called *fault sneaking attack (FSA)*. It is worth mentioning that such an attack is commonly performed at deep layers of a network due to its stealthyness requirement. We refer readers to Appendix for

Table 1: Certified perturbation bounds and ℓ_∞-norm of weight perturbations caused by FSA. Here FAS perturbs the last 4 layers of a 10-layer MLP under 4 datasets.

dataset	layer index	7	8	9	10
MNIST	original Acc (%)	97.8	97.8	97.8	97.8
	Acc after attack (%)	94	93.9	93.6	94.3
	certified bound	2×10^{-6}	2.6×10^{-5}	0.0003	0.0083
	attack perturbation	0.042	0.048	0.052	0.073
MNIST-Fashion	original Acc (%)	88.6	88.6	88.6	88.6
	Acc after attack (%)	84.6	84.3	84.0	84.3
	certified bound	2×10^{-6}	2.4×10^{-5}	0.00033	0.0117
	attack perturbation	0.025	0.0306	0.0351	0.11
SVHN	original Acc (%)	82.6	82.6	82.6	82.6
	Acc after attack (%)	80.5	80.1	80.3	80.6
	certified bound	4×10^{-6}	5.6×10^{-5}	0.0008	0.035
	attack perturbation	0.023	0.027	0.036	0.102
CIFAR-10	original Acc (%)	56.7	56.7	56.7	56.7
	Acc after attack (%)	50.2	51.1	51.5	51.2
	certified bound	4×10^{-6}	5×10^{-5}	0.0007	0.027
	attack perturbation	0.036	0.04	0.056	0.15

the detailed setting of attack generation and our experiment.

Table 1 shows the original accuracy (Acc), Acc after FSA, certified lower bound on weight perturbation, and the ℓ_∞ norm of weight perturbations caused by FSA. We see that the certified lower bound decreases as the layer index decreases since the linear bound approximation becomes looser when it needs to propagate over more layers prior to the output layer. Moreover, the certified lower bound shares the same pattern of FSA against the layer index. The indicates the intrinsic robustness of networks against FSA: A shallower layer is more vulnerable to FSA, which is also verified by Figure A1. Additional results on ImageNet are shown in the appendix.

Conclusion

In this paper, we take the first step to study the sensitivity of neural networks to weight perturbations and propose an efficient algorithm for computing a certified robustness bound. Our study on weight perturbation could be useful in both non-adversarial (weight quantization with certified robustness) and adversarial environments (robustness indicator against weight perturbation).

Acknowledgments

This work is partly supported by National Science Foundation CNS-1929300.

References

Abadi, M.; Agarwal, A.; Barham, P.; and et. al. 2016. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *2018 ICML* arXiv preprint arXiv:1802.00420.

Boopathy, A.; Weng, T.-W.; Chen, P.-Y.; Liu, S.; and Daniel, L. 2019. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *AAAI*.

Boyd, S.; Parikh, N.; Chu, E.; et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*.

Bunel, R. R.; Turkaslan, I.; Torr, P.; Kohli, P.; and Mudigonda, P. K. 2018. A unified view of piecewise linear neural network verification. In *NIPS*, 4790–4799.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 39–57. IEEE.

Carlini, N., and Wagner, D. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, 1–7.

Chen, P.-Y.; Sharma, Y.; Zhang, H.; Yi, J.; and Hsieh, C.-J. 2018. Ead: elastic-net attacks to deep neural networks via adversarial examples. *AAAI*.

Cheney, N.; Schrimpf, M.; and Kreiman, G. 2017. On the robustness of convolutional neural networks to internal architecture and weight perturbations. *arXiv preprint arXiv:1703.08245*.

Dutta, S.; Jha, S.; Sanakaranarayanan, S.; and Tiwari, A. 2017. Output range analysis for deep neural networks. *arXiv preprint arXiv:1709.09130*.

Dvijotham, K.; Stanforth, R.; Gowal, S.; et al. 2018. A dual approach to scalable verification of deep networks. *UAI*.

Ehlers, R. 2017. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 269–286. Springer.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE SP*.

Hein, M., and Andriushchenko, M. 2017. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NIPS*.

Huang, F., and Chen, S. 2018. Mini-batch stochastic admms for nonconvex nonsmooth optimization. *arXiv preprint arXiv:1802.03284*.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In *NeurIPS*.

Katz, G.; Barrett, C.; Dill, D. L.; et al. 2017. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 97–117. Springer.

Ko, C.-Y.; Lyu, Z.; Weng, T.-W.; Daniel, L.; Wong, N.; and Lin, D. 2019. Popqorn: Quantifying robustness of recurrent neural networks. *arXiv preprint arXiv:1905.07387*.

Kolter, J. Z., and Wong, E. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. *2018 ICML* arXiv preprint arXiv:1711.00851.

Leng, C.; Dou, Z.; Li, H.; et al. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *AAAI*.

Lin, D.; Talathi, S.; and Annapureddy, S. 2016. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, 2849–2858.

Liu, Y.; Wei, L.; Luo, B.; and Xu, Q. 2017. Fault injection attack on deep neural network. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 131–138.

Liu, X.; Cheng, M.; Zhang, H.; and Hsieh, C.-J. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 369–385.

Liu, C.; Arnon, T.; Lazarus, C.; Barrett, C.; and Kochenderfer, M. J. 2019. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. *2018 ICLR* arXiv preprint arXiv:1706.06083.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2574–2582.

Papernot, N.; McDaniel, P.; Swami, A.; and Harang, R. 2016a. Crafting adversarial input sequences for recurrent neural networks. In *IEEE Military Communications Conference (MILCOM)*, 49–54.

Papernot, N.; McDaniel, P.; Wu, X.; et al. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, 582–597.

Park, E.; Ahn, J.; and Yoo, S. 2017. Weighted-entropy-based quantization for deep neural networks. In *CVPR*, 7197–7205.

Paszke, A.; Gross, S.; Chintala, S.; and Chanan, G. 2017. Pytorch.

Raghunathan, A.; Steinhardt, J.; and Liang, P. 2018. Certified defenses against adversarial examples. *ICLR*.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 525–542. Springer.

Ren, A.; Zhang, T.; Ye, S.; et al. 2019. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *ASPLOS*, 925–938. ACM.

Sheng, T.; Feng, C.; Zhuo, S.; et al. 2018. A quantization-friendly separable convolution for mobilenets. In *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications*, 14–18. IEEE.

Song, D.; Eykholt, K.; Evtimov, I.; et al. 2018. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; et al. 2014. Intriguing properties of neural networks. *2014 ICLR*.

Wang, Y.-S.; Weng, T.-W.; and Daniel, L. 2019. Verification of neural network control policy under persistent adversarial perturbation. *arXiv preprint arXiv:1908.06353*.

Weng, T.-W.; Zhang, H.; Chen, H.; et al. 2018a. Towards fast computation of certified robustness for relu networks. *ICML*.

Weng, T.-W.; Zhang, H.; Chen, P.-Y.; et al. 2018b. Evaluating the robustness of neural networks: An extreme value theory approach. *ICLR*.

Weng, L.; Chen, P.-Y.; Nguyen, L.; Squillante, M.; Boopathy, A.; Oseledets, I.; and Daniel, L. 2019. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In *ICML 2019*.

Wong, E., and Kolter, J. Z. 2017. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*.

Wong, E.; Schmidt, F.; Metzen, J. H.; and Kolter, J. Z. 2018. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*.

Wu, J.; Leng, C.; Wang, Y.; et al. 2016. Quantized convolutional neural networks for mobile devices. In *CVPR*, 4820–4828.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. In *NIPS*.

Zhao, P.; Liu, S.; Chen, P.-Y.; and et. al. 2019a. On the design of black-box adversarial examples by leveraging gradient-free optimization and operator splitting method. In *ICCV 2019*.

Zhao, P.; Wang, S.; Gongye, C.; et al. 2019b. Fault sneaking attack: a stealthy framework for misleading deep neural networks. In *Proceedings of the 56th Annual Design Automation Conference*.

Zhou, A.; Yao, A.; Guo, Y.; et al. 2017. Incremental network quantization: Towards lossless cnns with low-precision weights. *2017 ICLR*.