

IMPROVED GRADIENT BASED ADVERSARIAL ATTACKS FOR QUANTIZED NETWORKS

– SUPPLEMENTARY MATERIAL –

Anonymous authors

Paper under double-blind review

Here, we first provide the pseudocodes, proof of the proposition and the derivation of Hessian. Later we give additional experiments, analysis and the details of our experimental setting.

A PSEUDOCODE

We provide pseudocode for PGD++ with NJS in Algorithm 1 and PGD++ with HNS in Algorithm 2.

Algorithm 1 PGD++ with NJS with L_∞ , T iterations, radius ϵ , step size η , network $f_{\mathbf{w}^*}$, input \mathbf{x}^0 , label k , one-hot $\mathbf{y} \in \{0, 1\}^d$, gradient threshold ρ .

Require: $T, \epsilon, \eta, \rho, \mathbf{x}^0, \mathbf{y}, k$
Ensure: $\|\mathbf{x}^{T+1} - \mathbf{x}^0\|_\infty \leq \epsilon$

- 1: $\beta_1 = (Md) / (\sum_{i=1}^M \sum_{j=1}^d \mu_j(\mathbf{J}_i))$ $\triangleright \beta_1$ computed using Network Jacobian.
- 2: $\mathbf{x}^1 = P_\infty^\epsilon(\mathbf{x}^0 + \text{Uniform}(-1, 1))$ \triangleright Random Initialization with Projection
- 3: **for** $t \leftarrow 1, \dots, T$ **do**
- 4: $\beta_2 = 1.0$
- 5: $\mathbf{p}' = \text{softmax}(\beta_1(f_{\mathbf{w}^*}(\mathbf{x}^t)))$
- 6: **if** $1 - p'_k \leq \rho$ **then** $\triangleright \rho = 0.01$
- 7: $\beta_2 = -\log(\rho/(d-1)(1-\rho))/\gamma$ $\triangleright \gamma$ computed using Proposition 1
- 8: $\ell = -\mathbf{y}^T \log(\text{softmax}(\beta_2 \beta_1(f_{\mathbf{w}^*}(\mathbf{x}^t))))$
- 9: $\mathbf{x}^{t+1} = P_\infty^\epsilon(\mathbf{x}^t + \eta \text{sign}(\nabla_{\mathbf{x}} \ell(\mathbf{x}^t)))$ \triangleright Update Step with Projection

Algorithm 2 PGD++ with HNS with L_∞ , T iterations, radius ϵ , step size η , network $f_{\mathbf{w}^*}$, input \mathbf{x}^0 , label k , one-hot $\mathbf{y} \in \{0, 1\}^d$, gradient threshold ρ .

Require: $T, \epsilon, \eta, \mathbf{x}^0, \mathbf{y}, k$
Ensure: $\|\mathbf{x}^{T+1} - \mathbf{x}^0\|_\infty \leq \epsilon$

- 1: $\mathbf{x}^1 = P_\infty^\epsilon(\mathbf{x}^0 + \text{Uniform}(-1, 1))$ \triangleright Random Initialization with Projection
- 2: $\beta^* = \text{argmax}_{\beta > 0} \|\partial^2 \ell(\beta) / \partial (\mathbf{x}^0)^2\|_F$ \triangleright Grid Search
- 3: **for** $t \leftarrow 1, \dots, T$ **do**
- 4: $\ell = -\mathbf{y}^T \log(\text{softmax}(\beta^*(f_{\mathbf{w}^*}(\mathbf{x}^t))))$
- 5: $\mathbf{x}^{t+1} = P_\infty^\epsilon(\mathbf{x}^t + \eta \text{sign}(\nabla_{\mathbf{x}} \ell(\mathbf{x}^t)))$ \triangleright Update Step with Projection

B DERIVATIONS

B.1 DERIVING β GIVEN A LOWERBOUND ON $1 - p_k(\beta)$

Proposition 1. Let $\mathbf{a}^K \in \mathbb{R}^d$ with $d > 1$ and $a_1^K \geq a_2^K \geq \dots \geq a_d^K$ and $a_1^K - a_d^K = \gamma$. For a given $0 < \rho < (d-1)/d$, there exists a $\beta > 0$ such that $1 - \text{softmax}(\beta \mathbf{a}_1^K) > \rho$, then $\beta < -\log(\rho/(d-1)(1-\rho))/\gamma$.

Proof. Assuming $a_1^K - a_d^K = \gamma$, we derive a condition on β such that $1 - \text{softmax}(\beta a_1^K) > \rho$.

$$\begin{aligned} 1 - \text{softmax}(\beta a_1^K) &> \rho, \\ \text{softmax}(\beta a_1^K) &< 1 - \rho, \\ \exp(\beta a_1^K) / \sum_{\lambda=1}^d \exp(\beta a_\lambda^K) &< 1 - \rho, \\ 1 / (1 + \sum_{\lambda=2}^d \exp(\beta(a_\lambda^K - a_1^K))) &< 1 - \rho. \end{aligned} \tag{1}$$

Since, $a_1^K - a_\lambda^K \leq \gamma$ for all $\lambda > 1$,

$$1 / (1 + \sum_{\lambda=2}^d \exp(\beta(a_\lambda^K - a_1^K))) \leq 1 / (1 + \sum_{\lambda=2}^d \exp(-\beta\gamma)). \tag{2}$$

Therefore, to ensure $1 / (1 + \sum_{\lambda=2}^d \exp(\beta(a_\lambda^K - a_1^K))) < 1 - \rho$, we consider,

$$\begin{aligned} 1 / (1 + \sum_{\lambda=2}^d \exp(-\beta\gamma)) &< 1 - \rho, \quad a_1^K - a_\lambda^K \leq \gamma \text{ for all } \lambda > 1, \\ 1 / (1 + (d-1) \exp(-\beta\gamma)) &< 1 - \rho, \\ \exp(-\beta\gamma) &> \rho / (d-1)(1-\rho), \\ -\beta\gamma &> \log(\rho / (d-1)(1-\rho)), \quad \text{exp is monotone}, \\ \beta &< -\log(\rho / (d-1)(1-\rho)) / \gamma. \end{aligned} \tag{3}$$

Therefore for any $\beta < -\log(\rho / (d-1)(1-\rho)) / \gamma$, the above inequality $1 - \text{softmax}(\beta a_1^K) > \rho$ is satisfied. \square

B.2 DERIVATION OF HESSIAN

We now derive the Hessian of the input mentioned in Eq. (8) of the paper. The input gradients can be written as:

$$\frac{\partial \ell(\beta)}{\partial \mathbf{x}^0} = \frac{\partial \ell(\beta)}{\partial \mathbf{p}(\beta)} \frac{\partial \mathbf{p}(\beta)}{\partial \bar{\mathbf{a}}^K(\beta)} \beta \mathbf{J} = \psi(\beta) \beta \mathbf{J}. \tag{4}$$

Now by product rule of differentiation, input hessian can be written as:

$$\begin{aligned} \frac{\partial^2 \ell(\beta)}{\partial (\mathbf{x}^0)^2} &= \beta \left[\psi(\beta) \frac{\partial \mathbf{J}}{\partial \mathbf{x}^0} + \left(\frac{\partial \psi(\beta)}{\partial \mathbf{x}^0} \right)^T \mathbf{J} \right], \\ &= \beta \left[\psi(\beta) \frac{\partial \mathbf{J}}{\partial \mathbf{x}^0} + \left(\frac{\partial \mathbf{p}(\beta)}{\partial \mathbf{x}^0} \right)^T \mathbf{J} \right], \quad \psi(\beta) = -(\mathbf{y} - \mathbf{p}(\beta))^T, \\ &= \beta \left[\psi(\beta) \frac{\partial \mathbf{J}}{\partial \mathbf{x}^0} + \beta \left(\frac{\partial \mathbf{p}(\beta)}{\partial \bar{\mathbf{a}}^K} \mathbf{J} \right)^T \mathbf{J} \right]. \end{aligned} \tag{5}$$

C ADDITIONAL EXPERIMENTS

In this section we first provide more experimental details and then some ablation studies.

C.1 EXPERIMENTAL DETAILS

Method	ResNet-18				VGG-16			
	APGD	Square Attack	PGD++ NJS	HNS	APGD	Square Attack	PGD++ NJS	HNS
REF	0.00	0.55	0.00	0.00	0.79	2.25	0.00	0.00
BNN-WQ	0.00	0.41	0.00	0.00	8.23	1.98	0.00	0.00
BNN-WAQ	6.32	21.45	0.03	0.04	0.38	16.67	0.01	0.02

Table 2: Adversarial accuracy for REF, BNN-WQ and BNN-WAQ trained on CIFAR-10 using ResNet-18. Both our NJS and HNS variants consistently outperform Auto-PGD (APGD) (Croce & Hein (2020)) performed using Difference of Logits Ratio (DLR) loss and a gradient free attack namely, Square Attack (Andriushchenko et al. (2020)) under L_∞ bound (8/255).

We first mention the hyperparameters used to perform FGSM and PGD attack for all the experiments in the paper in Table 1. To make a fair comparison, we keep the attack parameters same for our proposed variants of FGSM++ and PGD++ attacks. For PGD++ with HNS variant, we maximize Frobenius norm of Hessian with respect to the input as specified in Eq. (8) of the paper by grid search for the optimum β . We would like to point out that since only $\psi(\beta)$ and $\mathbf{p}(\beta)$ terms are dependent on β , we do not need to do forward and backward pass of the network multiple times during the grid search. This significantly reduces the computational overhead during the grid search. We can simply use the same network outputs \mathbf{a}^K and network jacobian \mathbf{J} (as computed without using β) for the grid search, while computing the other terms at each iteration of grid search. We apply grid search to find the optimum beta between 100 equally spaced intervals of β starting from β_1 to β_2 . Here, β_1 and β_2 are computed based on Proposition 1 in the paper where $\rho = 1e - 72$ and $\rho = 1 - (1/d) - (1e - 2)$ respectively, where d is number of classes and $\gamma = a_1^K - a_2^K$ so that $1 - \text{softmax}(\beta a_1^K) < \rho$. Also, note that we estimate the optimum β for each test sample only at the start of the first iteration of an iterative attack and then use the same β for the next iterations.

Dataset	Attack	ϵ	η	T
CIFAR-10	FGSM	8	8	1
	PGD (L_∞)	8	2	20
	PGD (L_2)	120	15	20
CIFAR-100	FGSM	4	4	1
	PGD (L_∞)	4	1	10
	PGD (L_2)	60	15	10

Table 1: Attack parameters (ϵ & η in pixels).

Computational Overhead of NJS and HNS. Our Jacobian calculation takes just a single backward pass through the network and thus adds a negligible overhead. Our NJS approach for scaling estimates β as inverse of mean JSV using 100 random test samples, which is similar to 100 backward passes. For HNS, in Eq. (8) Jacobian \mathbf{J} can be computed in single backward pass. Moreover, for piecewise linear networks (eg, relu activations), $\partial \mathbf{J} / \partial \mathbf{x}^0 = 0$ almost everywhere (Yao et al. (2018)). Thus PGD++ with NJS and HNS is almost as efficient as PGD.

C.2 COMPARISONS AGAINST AUTO-PGD ATTACK AND GRADIENT FREE ATTACK

We also compared our proposed PGD++ variants against recently proposed Auto-PGD (APGD) with Difference of Logits Ratio (DLR) loss (Croce & Hein (2020)) and gradient free Square Attack (Andriushchenko et al. (2020)) on different networks trained using ResNet-18 and VGG-16 on CIFAR-10 dataset and the results are reported in Table 2. The attack parameters for this experiment are the same as reported in the paper. It can be clearly seen that our proposed variants perform much better than both APGD with DLR loss and Square Attack, consistently achieving 0% adversarial accuracy. Infact, much computationally expensive Square attack is unable to achieve 0% adversarial accuracy in any of the cases under the enforced L_∞ bound.

C.3 OTHER EXPERIMENTS

We provide adversarial accuracy comparisons for different attack methods on CIFAR-100 using ResNet-18, VGG-16, ResNet-50 and DenseNet-121 in Table 3. Again similar to the results in the paper, our proposed PGD++ and FGSM++ outperform original form of PGD and FGSM consistently in all the experiments on floating point networks. We also provide adversarial accuracy comparison of our proposed variants against stronger attacks namely DeepFool (Moosavi-Dezfooli et al. (2016)) and

Network	Adversarial Accuracy (%)									
	FGSM	FGSM++		PGD (L_∞)	PGD++ (L_∞)			PGD (L_2)	PGD++ (L_2)	
		NJS	HNS		NJS	HNS	NJS		HNS	
ResNet-18	9.06	9.23	2.70	0.14	0.14	0.00	5.38	0.17	0.15	
VGG-16	16.28	17.24	9.19	1.53	0.95	0.25	4.87	1.50	1.38	
ResNet-50	12.95	12.95	11.94	0.12	0.00	0.00	31.01	4.43	4.14	
DenseNet-121	11.41	11.41	10.74	0.00	0.00	0.00	6.10	3.09	2.76	

Table 3: Adversarial accuracy on the test set of CIFAR-100 for REF (floating point networks). Both our NJS and HNS variants consistently outperform original FGSM and PGD (L_∞/L_2 bounded) attacks.

Network	PGD	Deep Fool	BBA	PGD++	
				NJS	HNS
ResNet-18	8.57	18.92	0.81	0.03	0.04
VGG-16	78.01	12.12	0.10	0.01	0.02

Table 4: Adversarial accuracy on the test set of CIFAR-10 for BNN-WAQ. Here, we compare our proposed variants against much stronger attacks namely DeepFool (Moosavi-Dezfooli et al. (2016)) and BBA (Brendel et al. (2019)). Both our variants outperform stronger attacks. Note, DeepFool and BBA are much slower in practise requiring 100-1000 iterations. BBA specifically requires even an adversarial start point that needs to be computed using another adversarial attack.

BBA (Brendel et al. (2019)) on BNN-WAQ trained on CIFAR-10 dataset in Table 4. In this experiment, our proposed variants again outperform even the stronger attacks which take 100-1000 iterations with adversarial start point (instead of random initial perturbation). It should be noted that although BBA performs much better than DeepFool and PGD, it still has inferior success rate than ours considering the fact that it takes multiple hours to run BBA whereas our proposed variants are almost as efficient as PGD attack.

Step Size Tuning for PGD attack. We would like to point out that step size η and temperature scale β have different effects in the attacks performed. Notice, PGD and FGSM attack under L_∞ bound only use the sign of input gradients in each gradient ascent step. Thus, if the input gradients are completely saturated (which is the case for BNNs), original forms of PGD or FGSM will not work irrespective of the step size used. To illustrate this, we performed extensive step size tuning for original form of PGD attack on different ResNet-18 models trained on CIFAR-10 dataset and the adversarial accuracies are reported in Fig. 1. It can be observed clearly that although tuning the step size lowers adversarial accuracy a bit in some cases but still cannot reach zero for BNNs unlike our proposed variants.

Adversarial training using PGD++. We also investigate the potential application of PGD++ for adversarial training to improve the robustness of neural networks. PGD++ attack is most effective when applied to a network with poor signal propagation. However, adversarial training is performed from random initialization (Glorot & Bengio (2010)) exhibiting good signal propagation. Thus, PGD and PGD++ perform similarly for adversarial training. We infer these conclusions from our experiments on adversarial training using PGD++.

CLEVER Scores. Recently CLEVER Scores (Weng et al. (2018)) have been proposed as an empirical estimate to measure robustness lower bounds for deep networks. It has been later shown that gradient masking issues cause CLEVER to overestimate the robustness bounds (Goodfellow (2018)). Here we try to improve the CLEVER scores using different ways of choosing β in temperature scaling. For this experiment, we use CLEVER implementation of Adversarial Training Toolbox¹ (Nicolae et al. (2018)). We set number

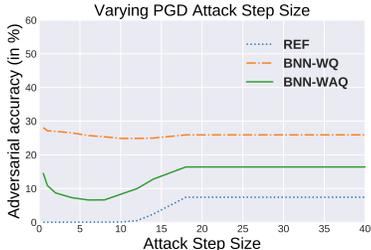


Figure 1: Adversarial accuracy using PGD attack under L_∞ bound (8/255) with varying step size (η) on ResNet-18 trained on CIFAR-10. Notice, PGD attack is unable to reach zero adversarial accuracy for BNNs with any step size.

¹<https://github.com/Trusted-AI/adversarial-robustness-toolbox>

	Original	Heuristic	NJS	HNS
BNN-WQ	0.8585	0.8845	0.4139	0.3450
BNN-WAQ	0.7239	3.1578	0.3120	0.2774

Table 5: CLEVER Scores (Weng et al. (2018)) for BNN-WQ and BNN-WAQ trained on CIFAR-10 using ResNet-18. We compare CLEVER Scores returned for L_1 norm perturbation using different ways of temperature scaling applied. Here, Original refers to original network without temperature scaling and Heuristic denotes temperature scale with small $\beta = 0.01$.

Methods	PGD++ (NJS) - Varying ρ					
	$1e-05$	$1e-04$	$1e-03$	$1e-02$	$1e-01$	$2e-01$
REF	0.00	0.00	0.00	0.00	0.00	0.00
BNN-WQ	0.00	0.00	0.00	0.00	0.00	0.00
BNN-WAQ	0.15	0.08	0.04	0.03	0.04	0.02

Table 6: Adversarial accuracy on the test set for binary neural networks using L_∞ bounded PGD++ attack using NJS with varying ρ . For different values of ρ , our approach is quite stable.

of batches to 50, batch size to 10, radius to 5, and chose L_1 norm as hyperparameters (based on the Weng et al. (2018)). We compare our variants namely NJS and HNS against heuristic choice of small $\beta = 0.01$ and original CLEVER Scores for BNN-WQ and BNN-WAQ (trained on CIFAR-10 using ResNet-18) in Table 5. It can be clearly seen that our proposed variants improve the robustness bounds computed using CLEVER whereas a heuristic choice of $\beta = 0.01$ performs even worse.

C.4 STABILITY OF PGD++ WITH NJS WITH VARIATIONS IN ρ

We perform ablation studies with varying ρ for PGD++ with NJS in Table 6 for CIFAR-10 dataset using ResNet-18 architecture. It clearly illustrates that our NJS variant is quite robust to the choice of ρ as we are able to achieve near perfect success rate with PGD++ with different values of ρ . As long as value of ρ is large enough to avoid one-hot encoding on softmax outputs (in turn avoid $\|\psi(\beta)\|$ to be zero) of correctly classified sample, our approach with NJS variant is quite stable.

C.5 SIGNAL PROPAGATION AND INPUT GRADIENT ANALYSIS USING NJS AND HNS

We first provide an example illustration in Fig. 2 to better understand how the input gradient norm *i.e.*, $\|\partial\ell(\beta)/\partial\mathbf{x}^0\|_2$, and norm of sign of input gradient, *i.e.*, $\|\text{sign}(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$ is influenced by β . It clearly shows that both the plots have a concave behavior where an optimal β can maximize the input gradient. Also, it can be quite evidently seen in Fig. 2 (b) that within an optimal range of β , gradient vanishing issue can be avoided. If $\beta \rightarrow 0$ or $\beta \rightarrow \infty$, it changes all the values in input gradient matrix to zero and return $\|\text{sign}(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2 = 0$.

We also provide the signal propagation properties as well as analysis on input gradient norm before and after using the β estimated based on NJS and HNS in Table 7. For binarized networks as well floating point networks tested on CIFAR-10 dataset using ResNet-18 architecture, our HNS and NJS variants result in larger values for $\|\psi\|_2$, $\|\partial\ell(\beta)/\partial\mathbf{x}^0\|_2$ and $\|\text{sign}(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$. This reflects the efficacy of our method in overcoming the gradient vanishing issue. It can be also noted that our variants also improves the signal propagation of the networks by bringing the mean JSV values closer to 1.

C.6 ABLATION FOR ρ VS. PGD++ ACCURACY

In this subsection, we provide the analysis on the effect of bounding the gradients of the network output of ground truth class k , *i.e.* $\partial\ell(\beta)/\partial\bar{a}_k^K$. Here, we compute β using Proposition 1 for all correctly classified images such that $1 - \text{softmax}(\beta a_k^K) > \rho$ with different values of ρ and report the PGD++ adversarial accuracy in Table 8. It can be observed that there is an optimum value of ρ at which PGD++ success rate is maximized, especially on the adversarially trained models. This can also be seen in connection with the non-linearity of the network where at an optimum value of β ,

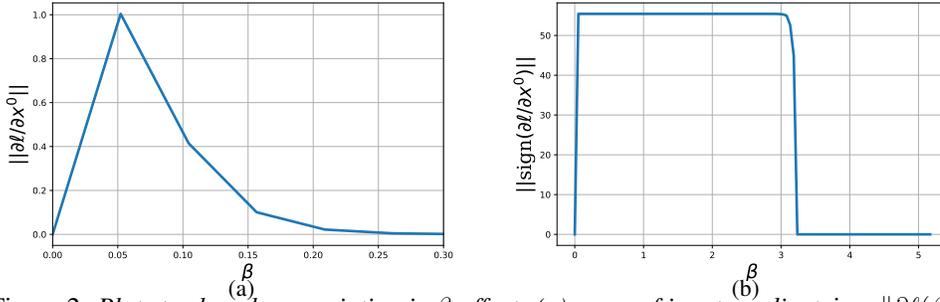


Figure 2: Plots to show how variation in β affects (a) norm of input gradient, i.e., $\|\partial\ell(\beta)/\partial\mathbf{x}^0\|_2$, (b) norm of sign of input gradient, i.e., $\|\text{sign}(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$ on a random correctly classified image. Notice that, both input gradient and signed input gradient norm behave similarly, showing a concave behaviour. This plot is computed for BNN-WQ network on CIFAR-10, ResNet-18. (b) clearly illustrates how optimum β can avoid vanishing gradient issue since $\|\text{sign}(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$ will only be zero if input gradient matrix has only zeros.

Methods		REF	Adv. Train	BNN-WQ	BNN-WAQ
JSV (Mean)	Orig.	8.09e+00	5.15e-01	3.53e+01	1.11e+00
	NJS	9.51e-01	5.70e-01	9.95e-01	2.24e-01
	HNS	2.38e+00	6.11e+00	1.19e+01	4.65e+00
JSV (Std.)	Orig.	6.27e+00	4.10e-01	3.53e+01	1.97e+00
	NJS	7.58e-01	6.34e-01	9.71e-01	6.73e-01
	HNS	4.41e+00	5.34e+02	2.13e+02	1.24e+02
$\ \psi\ _2$	Orig.	9.08e-03	2.33e-01	6.20e-03	9.46e-03
	NJS	4.66e-01	2.35e-01	5.37e-01	1.20e-01
	HNS	1.48e-01	2.57e-01	2.07e-01	2.44e-01
$\ \partial\ell/\partial\mathbf{x}^0\ _2$	Orig.	2.42e-01	8.52e-02	2.27e-01	6.33e-02
	NJS	9.52e-01	1.10e-01	8.91e-01	1.24e-01
	HNS	7.49e-01	8.18e-01	3.70e-01	2.70e-01
$\ \text{sign}(\frac{\partial\ell}{\partial\mathbf{x}^0})\ _2$	Orig.	5.55e+01	5.54e+01	4.39e+01	5.55e+01
	NJS	5.55e+01	5.54e+01	5.55e+01	5.55e+01
	HNS	5.55e+01	5.54e+01	5.55e+01	5.55e+01

Table 7: Mean and standard deviation of Jacobian Singular Values (JSV), mean $\|\psi\|_2$, mean $\|\partial\ell/\partial\mathbf{x}^0\|_2$ and mean $\|\text{sign}(\partial\ell/\partial\mathbf{x}^0)\|_2$ for different methods on CIFAR-10 with ResNet-18 computed with 500 correctly classified samples. Note here for NJS and HNS, JSV is computed for scaled jacobian i.e. $\beta\mathbf{J}$. Also note that, values of $\|\psi\|_2$, $\|\partial\ell(\beta)/\partial\mathbf{x}^0\|_2$ and $\|\text{sign}(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$ are larger for our NJS and HNS variant (for most of the networks) as compared with network with no β , which clearly indicates better gradients for performing gradient based attacks.

even for robust (locally linear) (Moosavi-Dezfooli et al. (2019); Qin et al. (2019)) networks such as adversarially trained models, non-linearity can be maximized and better success rate for gradient based attacks can be achieved. Our HNS variant essentially tries to achieve the same objective while trying to estimate β for each example.

REFERENCES

- Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.
- Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems*, pp. 12861–12871, 2019.

Methods	PGD++ with Varying ρ					
	$1e-15$	$1e-09$	$1e-05$	$1e-01$	$2e-01$	$5e-01$
REF	0.00	0.00	0.00	0.00	0.00	0.00
BNN-WQ	9.61	0.04	0.00	0.00	0.00	0.00
REF*	48.18	47.66	48.00	53.09	54.58	57.57
BNN-WQ*	40.66	40.01	40.04	45.09	46.57	49.72

Table 8: Adversarial accuracy on the test set for adversarially trained networks and binary neural networks using L_∞ bounded PGD++ attack with varying ρ as lower bound on the gradient of network output for ground truth class k . Here * denotes the adversarially trained models obtained where adversarial samples are generated using L_∞ bounded PGD attack with $T = 7$ iterations, $\eta = 2$ and $\epsilon = 8$. Note, here PGD++ attack refers to PGD attack where $\partial\ell(\beta)/\partial\bar{a}_k^K$ is bounded by ρ for each sample, where k is ground truth class.

- Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *ICML*, 2020.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.
- Ian Goodfellow. Gradient masking causes clever to overestimate adversarial perturbation size. *arXiv preprint arXiv:1804.07870*, 2018.
- Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.
- Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9078–9086, 2019.
- Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Amrith Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.
- Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems*, pp. 13824–13833, 2019.
- Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*, 2018.
- Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, pp. 4949–4959, 2018.