Improved Gradient based Adversarial Attacks for Quantized Networks

Kartik Gupta[1,2] and Thalaiyasingam Ajanthan[1]

[1] Australian National University
[2] DATA61, CSIRO

Abstract. Neural network quantization has become increasingly popular due to efficient memory consumption and faster computation resulting from bitwise operations on the quantized networks. Even though they exhibit excellent generalization capabilities, their robustness properties are not well-understood. In this work, we systematically study the robustness of quantized networks against gradient based adversarial attacks and demonstrate that these quantized models suffer from gradient vanishing issues and show a fake sense of security. By attributing gradient vanishing to poor forward-backward signal propagation in the trained network, we introduce a simple temperature scaling approach to mitigate this issue while preserving the decision boundary. Despite being a simple modification to existing gradient based adversarial attacks, experiments on CIFAR-10/100 datasets with VGG-16 and ResNet-18 networks demonstrate that our temperature scaled attacks obtain near-perfect success rate on quantized networks while outperforming original attacks on adversarially trained models as well as floating-point networks.

1 Introduction

Neural Network (NN) quantization has become increasingly popular due to reduced memory and time complexity enabling real-time applications and inference on resource-limited devices. Such quantized networks often exhibit excellent generalization capabilities despite having low capacity due to reduced precision for parameters and activations. However, their robustness properties are not well-understood. In particular, while parameter quantized networks are claimed to have better robustness against gradient based adversarial attacks [10], activation only quantized methods are shown to be vulnerable [18].

In this work, we consider the extreme case of Binary Neural Networks (BNNs) and systematically study the robustness properties of parameter quantized models, as well as both parameter and activation quantized models against gradient based adversarial attacks. Our analysis reveals that these quantized models suffer from gradient masking issues [3] (especially vanishing gradients) and in turn show fake robustness. We attribute this vanishing gradients issue to poor forward-backward signal propagation caused by trained binary weights, and our idea is to improve signal propagation of the network without affecting the prediction of the classifier.

There is a body of work on improving signal propagation in a neural network (*e.g.*, [11,23]), however, we are facing a unique challenge of *improving signal propagation while preserving the decision boundary*, since our ultimate objective is to generate adversarial attacks. To this end, we first discuss the conditions to ensure informative gradients and resort to a temperature scaling approach [13] (which scales the logits before applying softmax cross-entropy) to show that, even with a single positive scalar the vanishing gradients issue in BNNs can be alleviated achieving *near perfect success rate* in all tested cases.

Specifically, we introduce two techniques to choose the temperature scale: 1) based on the singular values of the input-output Jacobian, 2) by maximizing the norm of the Hessian of the loss with respect to the input. The justification for the first case is that if the singular values of input-output Jacobian are concentrated around 1 (defined as dynamical isometry [23]) then the network is said to have good signal propagation and we intend to make the mean of singular values to be 1. On the other hand, the intuition for maximizing the Hessian norm is that if the Hessian norm is large, then the gradient of the loss with respect to the input is sensitive to an infinitesimal change in the input. This is a sufficient condition for the network to have good signal propagation as well as informative gradients under the assumption that the network does not have any randomized or non-differentiable components.

We evaluated our improved gradient based adversarial attacks on CIFAR-10/100 datasets with VGG-16 and ResNet-18 networks quantized using multiple recent techniques [1,2,4,16]. In all tested quantized models, both versions of our temperature scaled attacks obtained near perfect success rate outperforming gradient based attacks (FGSM [12], PGD [19]) in their original form. Furthermore, this temperature scaling improved gradient based attacks even on adversarially trained models (both high-precision and quantized) as well as floating-point networks, showing the significance of signal propagation for adversarial attacks.

2 Preliminaries

We first provide some background on the neural network quantization and adversarial attacks. We then empirically show how Binary Neural Networks (BNNs) tend to show fake sense of robustness and introduce modifications to existing gradient based adversarial attacks to overcome gradient masking issues in BNNs.

2.1 Neural Network Quantization

Neural Network (NN) quantization is defined as training networks with parameters constrained to a minimal, discrete set of quantization levels. This primarily relies on the hypothesis that since NNs are usually overparametrized, it is possible to obtain a quantized network with performance comparable to the floating-point network. Given a dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$, NN quantization can be written as:

$$\min_{\mathbf{w}\in\mathcal{Q}^m} L(\mathbf{w};\mathcal{D}) := \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w};(\mathbf{x}_i,\mathbf{y}_i)) \ . \tag{1}$$

Here, $\ell(\cdot)$ denotes the input-output mapping composed with a standard loss function (*e.g.*, cross-entropy loss), $\mathbf{w}$ is the m dimensional parameter vector, and $\mathcal{Q}$ is a predefined discrete set representing quantization levels (*e.g.*, $\mathcal{Q} = \{-1, 1\}$ in the binary case).

Most of the NN quantization approaches [1,2,4,16] convert the above problem into an unconstrained problem by introducing auxiliary variables ($\tilde{\mathbf{w}}$) and optimize via (stochastic) gradient descent. Specifically, the objective and the update step can be written as:

$$\min_{\tilde{\mathbf{w}} \in \mathbb{R}^m} L(P(\tilde{\mathbf{w}}); \mathcal{D}) \;, \qquad \tilde{\mathbf{w}}^{k+1} = \tilde{\mathbf{w}}^k - \eta \, \nabla L(P(\tilde{\mathbf{w}}); \mathcal{D})|_{\tilde{\mathbf{w}} = \tilde{\mathbf{w}}^k} \;. \tag{2}$$

Here, $P : \mathbb{R}^m \to \mathcal{Q}^m$ is a projection and $\eta > 0$ is the learning rate. To this end, the algorithms differ in the choice of quantization set (*e.g.*, keep it discrete [7], relax it to the convex hull [4] or convert the problem in to a lifted probability space [1]), the projection used and how differentiation through projection is performed. In the case when the constraint set is relaxed, a gradually increasing annealing hyperparameter is used to enforce a quantized solution [1,2,4]. We refer the interested reader to respective papers for more detail.

Note that, here we have described weight quantization, however, activation quantization can be analogously formulated. In this paper, we consider BNNs where the learnable parameters are constrained to the binary set $\{-1, 1\}$ and activations are not quantized unless otherwise specified.

2.2 Adversarial Attacks

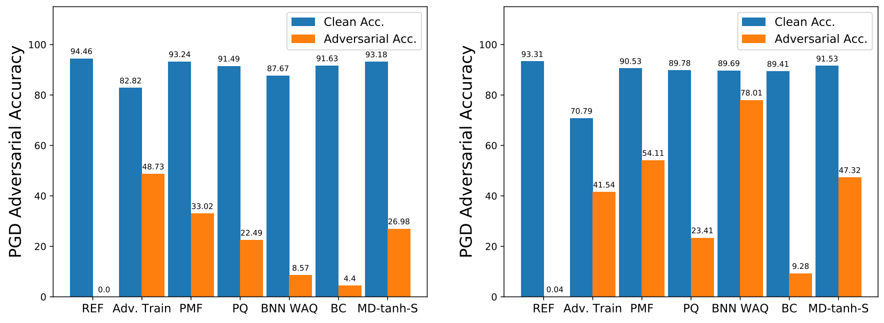
Adversarial examples consist of small bounded perturbations to the images that fool trained deep learning models by altering their prediction. Even though such perturbations are imperceptible to human vision, they are sufficient to alter the model's prediction with high confidence. Existing adversarial attacks can be categorized into white-box, and black-box attacks. The difference between them lies in the knowledge of the adversaries. White-box attacks allow the adversaries to have access to the target model's architecture and parameters. In the black-box threat model, the adversaries can only resort to the query access to generate adversarial samples. Since white-box gradient based attacks are more powerful and widely used, we now briefly summarize them below.

In fact, gradient based attacks can be compactly written as Projected Gradient Descent (PGD) on the negative of the loss function [19]. Formally, let $\mathbf{x}^0 \in \mathbb{R}^N$ be the input image, then at iteration t, the PGD update can be written as:

$$\mathbf{x}^{t+1} = P\left(\mathbf{x}^t + \eta \, \mathbf{g}_{\mathbf{x}}^t\right) \;, \tag{3}$$

where $P : \mathbb{R}^N \to \mathcal{X}$ is a projection, $\mathcal{X} \subset \mathbb{R}^N$ is the constraint set that bounds the perturbations, $\eta > 0$ is the step size, and $\mathbf{g}_{\mathbf{x}}^t$ is a form of gradient of the loss with respect to the input $\mathbf{x}$ evaluated at $\mathbf{x}^t$. With this general form, the popular gradient based adversarial attacks can be specified:

- **Fast Gradient Sign Method (FGSM)**: This is a one step attack introduced in [12]. Here, P is the identity mapping, η is the maximum allowed perturbation



(a) ResNet-18 (b) VGG-16

Fig. 1: *Recognition accuracy (clean vs. adversarial) on the test set of CIFAR-10 for binary networks with using different methods for quantization. Binarized networks consistently outperform robustness accuracy of floating point networks (*REF*). All quantization methods are parameter quantization methods except* BNN-WAQ *which has weight and activations both quantized.*

magnitude, and $\mathbf{g}_{\mathbf{x}}^t = \text{sign}\left(\nabla_{\mathbf{x}} \ell(\mathbf{w}^*; (\mathbf{x}^t, \mathbf{y}))\right)$, where ℓ denotes the loss function, $\mathbf{w}^*$ is the trained weights and $\mathbf{y}$ is the ground truth label corresponding to the image $\mathbf{x}^0$.

- **PGD with L_∞ bound**: Arguably the most popular adversarial attack introduced in [19] and sometimes referred to as Iterative Fast Gradient Sign Method (IFGSM). Here, P is the L_∞ norm based projection, η is a chosen step size, and $\mathbf{g}_{\mathbf{x}}^t = \text{sign}\left(\nabla_{\mathbf{x}} \ell(\mathbf{w}^*; (\mathbf{x}^t, \mathbf{y}))\right)$, the sign of gradient same as FGSM.
- **PGD with L_2 bound**: This is also introduced in [19] which performs the standard PGD in the Euclidean space. Here, P is the L_2 norm based projection, η is a chosen step size, and $\mathbf{g}_{\mathbf{x}}^t = \nabla_{\mathbf{x}} \ell(\mathbf{w}^*; (\mathbf{x}^t, \mathbf{y}))$ is simply the gradient of the loss with respect to the input.

These attacks have been further strengthened by first taking a random step as discussed in [28] and then performing projected gradient steps. In this paper, we always use this random initialization for all the attacks.

3 Robustness Evaluation of Binary Neural Networks

We start by evaluating the adversarial accuracy of BNNs trained using various techniques, namely BC [7], PQ [4], PMF [1], MD-tanh-S [2], BNN-WAQ [15] using the PGD attack with L_∞ bound where the attack details are summarized below:

- **PGD attack details**: perturbation bound of 8 pixels (assuming each pixel in the image is in $[0, 255]$) with respect to L_∞ norm, step size $\eta = 2$ and the total number of iterations $T = 20$. In all attacks, a randomized step is taken to initialize the perturbations as mentioned in Sec. 2.2. The attack details are the same in all evaluated settings unless stated otherwise.

We perform experiments on CIFAR-10 dataset using ResNet-18 and VGG-16 architectures and report the clean and adversarial accuracy results in Fig. 1.

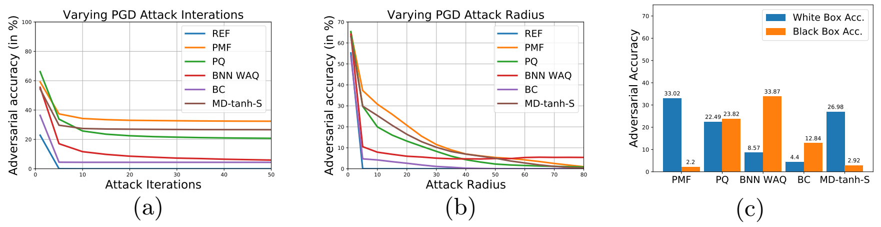


Fig. 2: *Plots to test for obfuscated gradients in ResNet-18 trained on CIFAR-10 via. (a) varying* PGD *attack iterations, (b) varying* PGD *attack radius, and (c) black box attacks. While (a) and (c) show signs of gradient masking, (b) does not, however, we attribute this discrepancy to the random initial step taken before* PGD *iterations as discussed in Sec. 2.2. Also notice, the quantization technique clearly has an influence on the gradient masking issues of the final trained model.*

It can be clearly and consistently observed that binary networks have high adversarial accuracy compared to the floating point counterparts. In fact, in some of the cases adversarial accuracy for a binary network is as close as or even better compared to the adversarially trained floating point networks. Since this result is surprising, we investigate this phenomenon further to understand whether BNNs are actually robust to adversarial perturbations or they show fake sense of security due to some form of obfuscated gradients [3].

3.1 Identifying Obfuscated Gradients

Recently, it has been shown that several defense mechanisms intentionally or unintentionally break gradient descent and cause obfuscated gradients and thus exhibit a false sense of security [3]. Several gradient based adversarial attacks tend to fail to produce adversarial perturbations in scenarios where the gradients are uninformative, referred to as gradient masking. Gradient masking can occur due to shattered gradients, stochastic gradients or exploding and vanishing gradients. We try to identify gradient masking in binary networks based on the empirical checks provided in [3]. If any of these checks fail, it indicates an issue of gradient masking in binarized neural networks.

To illustrate this, we analyse the effects of varying different hyperparameters of PGD attack on various binarized networks trained on CIFAR-10 using ResNet-18 architecture. Even though varying PGD perturbation bound does not show any signs of gradient masking, varying attack iterations and black-box vs white-box results clearly indicate gradient masking issues as depicted in Fig. 2. Here, our black-box model to a BNN is the analogous floating point network trained on the same dataset and the attack is the same PGD with L_∞ bound.

These checks demonstrate that BNNs are prone to gradient masking and exhibit a fake sense of security. Note that, shattered gradients occur due to non-differentiable components in the defense mechanism and stochastic gradients are caused by randomized gradients due to randomly transformed input or randomly transformed network. To this end, since BNNs are trainable from scratch and does not have randomized gradients[3], we can narrow down gradient masking issue to vanishing or exploding gradients. Since, vanishing or exploding gradients occur due to poor signal propagation, by introducing a single scalar, we discuss two different approaches to mitigate this issue, which lead to almost 100% success rate for gradient based attacks on BNNs.

4 Signal Propagation of Neural Networks

In this section, let us first describe the feed-forward dynamics of neural networks and how poor signal propagation can cause vanishing or exploding gradients. We then discuss the idea of introducing a single scalar which can improve the existing gradient descent attacks without affecting the prediction (*i.e.*, decision boundary) of the trained models.

For notational convenience, similar to [23], we consider a fully-connected neural network $f_{\mathbf{w}}$ with weights $\mathbf{W}^l \in \mathbb{R}^{\mathrm{N_l} \times \mathrm{N_{l-1}}}$, biases $\mathbf{b}^l \in \mathbb{R}^{\mathrm{N_{l-1}}}$, pre-activations $\mathbf{h}^l \in \mathbb{R}^{\mathrm{N_l}}$, and post-activations $\mathbf{a}^l \in \mathbb{R}^{\mathrm{N_l}}$, for $l \in \{1 \ldots K\}$ up to K layers. Now, the feed-forward dynamics can be formulated as,

$$\mathbf{a}^l = \phi(\mathbf{h}^l) \ , \qquad \mathbf{h}^l = \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l \ , \tag{4}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is an elementwise nonlinearity, and the input is denoted by $\mathbf{a}^0 = \mathbf{x}^0 \in \mathbb{R}^{\mathrm{N}}$. Now, since softmax cross-entropy is usually used as the loss function, we can write:

$$\ell(\mathbf{a}^K, \mathbf{y}) = -\mathbf{y}^T \log(\mathbf{p}) \ , \qquad \mathbf{p} = \mathrm{softmax}(\mathbf{a}^K) \ , \tag{5}$$

where $\mathbf{y} \in \mathbb{R}^{\mathrm{d}}$ is the one-hot encoded target label and log is applied elementwise. Notice, we have subsumed the dependency of parameters into $\mathbf{a}^K = f_{\mathbf{w}}(\mathbf{x}^0)$ for simplicity of notation.

For various gradient based adversarial attacks discussed in Sec. 2.2, gradient of the loss ℓ is used with respect to the input $\mathbf{x}^0$, which can also be formulated using chain rule as,

$$\frac{\partial \ell(\mathbf{a}^K, \mathbf{y})}{\partial \mathbf{x}^0} = \frac{\partial \ell(\mathbf{a}^K, \mathbf{y})}{\partial \mathbf{a}^K} \frac{\partial \mathbf{a}^K}{\partial \mathbf{x}^0} = \psi(\mathbf{a}^K, \mathbf{y}) \, \mathbf{J} \ , \tag{6}$$

where ψ denotes the error signal and $\mathbf{J} \in \mathbb{R}^{d \times N}$ is the input-output Jacobian. Here we use the convention that $\partial \mathbf{v} / \partial \mathbf{u}$ is of the form $\mathbf{v}$-size $\times$ $\mathbf{u}$-size.

Notice there are two components that influence the gradients, 1) the Jacobian $\mathbf{J}$ and 2) the error signal ψ. Gradient based attacks would fail if either the Jacobian is poorly conditioned or the error signal has saturating gradients, both of these will lead to vanishing gradients in $\partial \ell / \partial \mathbf{x}^0$.

The effects of Jacobian on the signal propagation is studied in dynamical isometry and mean-field theory literature [23,26] and it is known that a network

[3] Even though, BNNs have binary weights, once trained, there is no non-differentiable or randomized component in the parameter quantized network but the gradients are evaluated at a discrete point.

Methods	REF	Adv. Train	BC [7]	PQ [4]	PMF [1]	MD-tanh-S [2]	BNN-WAQ [15]
JSV (Mean)	8.09e+00	5.15e−01	1.61e+01	2.34e+01	4.46e+01	3.53e+01	1.11e+00
JSV (Std.)	6.27e+00	4.10e−01	1.88e+01	2.35e+01	1.11e+02	3.53e+01	1.97e+00
$\|\psi\|_2$ (Mean)	9.08e−03	2.33e−01	1.18e−02	6.75e−03	8.50e−03	6.20e−03	9.46e−03

Table 1: *Mean and standard deviation of Jacobian Singular Values (*JSV*) and mean $\|\psi\|_2$ for different methods on CIFAR-10 with ResNet-18 computed with 500 correctly classified samples. Note the norm of the error signal ψ is very small in all cases except for Adv. Train, indicating that all models are over confident (probabilities close to one-hot) except for Adv. Train which is in fact under confident for correctly classified samples. Furthermore, one can clearly see that* BNN*s (except* BNN-WAQ*) have much higher* JSV *mean and we believe this leads to gradient vanishing,* i.e.*, increased scale for logits $\mathbf{a}^K$ and in turn reduced (if not zero) error signal ψ.*

is said to satisfy dynamical isometry if the singular values of $\mathbf{J}$ are concentrated near 1, *i.e.*, for a given $\epsilon > 0$, the singular value σ_j satisfies $1 - \sigma_j \le \epsilon$ for all j. Under this condition, error signals ψ backpropagate isometrically through the network, approximately preserving its norm and all angles between error vectors. Thus, just like dynamical isometry speeds up the training for the floating point networks by improving the signal propagation, a similar technique can be useful for gradient based attacks as well.

In fact, almost all initialization techniques (*e.g.*, [11]) approximately ensures that the Jacobian $\mathbf{J}$ is well-conditioned for better trainability. For continuous networks trained on the clean samples, it is hypothesized that approximate isometry is preserved even at the end of the training but this is not the case for binary networks. In fact, for BNNs, the weights are constrained to be $\{-1, 1\}$ and hence the weight distribution at end of training is completely different from the random initialization. Furthermore, it is not clear that fully-quantized networks can achieve well-conditioned Jacobian, which guided some research activity in utilizing layerwise scalars (either predefined or learned) to improve BNN training [20,25]. We illustrate the signal propagation properties of various networks in Table 1.

We would like to point out that the focus of this paper is to improve gradient based attacks on already trained BNNs. To this end learning a new scalar to improve signal propagation at each layer is not useful as it can alter the decision boundary of the network and thus cannot really be used in practice on already trained model.

4.1 Temperature Scaling for better Signal Propagation

In this paper, we propose to use a single scalar per network to improve the signal propagation of the network using temperature scaling. In fact, one could replace softmax with a monotonic function such that the prediction is not altered, however, we will show in our experiments that a single scalar with softmax has enough flexibility to improve signal propagation and yields almost 100% success rate with PGD attacks. Essentially, we can use a scalar, $\beta > 0$ without changing the decision boundary of the network by preserving the relative order of the logits. Precisely, we consider the following:

$$\mathbf{p}(\beta) = \mathrm{softmax}(\bar{\mathbf{a}}^K) \;, \qquad \bar{\mathbf{a}}^K = \beta\, \mathbf{a}^K \;. \tag{7}$$

Here, we write the softmax output probabilites $\mathbf{p}$ as a function of β to emphasize that they are the softmax output of temperature scaled logits. Now since in this context, the only variable is the temperature scale β, we denote the loss and the error signal as functions of only β. With this simplified notation the gradient of the temperature scaled loss with respect to the inputs can be written as:

$$\frac{\partial \ell(\beta)}{\partial \mathbf{x}^0} = \frac{\partial \ell(\beta)}{\partial \bar{\mathbf{a}}^K} \frac{\partial \bar{\mathbf{a}}^K}{\partial \mathbf{a}^K} \frac{\partial \mathbf{a}^K}{\partial \mathbf{x}^0} = \psi(\beta)\beta\, \mathbf{J} \;. \tag{8}$$

Note that β affects the input-output Jacobian linearly while it nonlinearly affects the error signal ψ. To this end, we hope to obtain a β that ensures the error signal is useful (*i.e.*, not all zero) as well as the Jacobian is well-conditioned to allow the error signal to propagate to the input.

We acknowledge that while one can find a $\beta > 0$ to obtain softmax output ranging from uniform distribution ($\beta = 0$) to one-hot vectors ($\beta \to \infty$), β only scales the Jacobian. Therefore, if the Jacobian $\mathbf{J}$ already has zero singluar values, this temperature scaling approach would not make any difference in those dimensions. However, since most of the modern networks consist of ReLU nonlinearites, the effect of this single scalar would be equivalent (ignoring the biases) to having a single positive scalar in each layer such as in [20]. In fact this is true for all functions $\phi : \mathbb{R} \to \mathbb{R}$ such that $\phi(ch) = c\phi(h)$ for $c > 0$. To this end, we believe a single scalar is sufficient for our purpose.

5 Improved Gradients for Adversarial Attacks

In this section we discuss strategies to choose a scalar β such that the gradients with respect to input are informative. Let us first analyze the effect of β on the error signal. To this end, we can write

$$\psi(\beta) = \frac{\partial \ell(\beta)}{\partial \mathbf{p}(\beta)} \frac{\partial \mathbf{p}(\beta)}{\partial \bar{\mathbf{a}}^K} = -(\mathbf{y} - \mathbf{p}(\beta))^T \;. \tag{9}$$

where $\mathbf{y}$ is the one-hot encoded target label, and $\mathbf{p}(\beta)$ is the softmax output of temperature scaled logits.

Note that, for adversarial attacks, we only consider the correctly classified images (*i.e.*, $\mathrm{argmax}_j\, y_j = \mathrm{argmax}_j\, p_j(\beta)$) as there is no need to generate adversarial examples corresponding to misclassified samples. From the above formula, it is clear that when $\mathbf{p}(\beta)$ is one-hot encoding then the error signal is $\mathbf{0}$. This is one of the reason for vanishing gradient issue in BNNs. Even if this does not happen for a given image, one can increase $\beta \to \infty$ to make this error signal $\mathbf{0}$. Similarly, when $\mathbf{p}(\beta)$ is the uniform distribution, the norm of the error signal is at the maximum. This can be obtained by setting $\beta = 0$. However, this would also make $\partial \ell(\beta)/\partial \mathbf{x}^0 = \mathbf{0}$ as the singular values of the input-output Jacobian would all be 0. Refer to Eq. (8). To understand how error signal is affected by β please refer to Fig. 3.

This analysis indicates that the optimal β cannot be obtained by simply maximizing the norm of the error signal and we need to balance both the Jacobian as well as the error signal. To summarize, the scalar β should be chosen such that the following properties are satisfied:

1. $\|\psi(\beta)\|_2 > \rho$ for some $\rho > 0$.
2. The input-output Jacobian $\beta\,\mathbf{J}$ is well-conditioned, *i.e.*, the singular values of $\beta\,\mathbf{J}$ is concentrated around 1.

We now discuss two approaches to obtain a β that alleviates vanishing gradients such that already existing gradient based attacks (FGSM, PGD) can be made more effective.

5.1 Network Jacobian Scaling (NJS)

We now discuss a straightforward, two-step approach to attain the aforementioned properties. Firstly, to ensure $\beta\mathbf{J}$ is well-conditioned, we simply choose β to be the inverse of the mean of singular values $\mathbf{J}$. This guarantees that the mean of singular values of $\beta\mathbf{J}$ is 1. Formally, let us choose M samples from the test set, we can derive β as follows:

$$\beta = \frac{M\,d}{\sum_{i=1}^{M}\sum_{j=1}^{d}\mu_j(\mathbf{J}_i)}\ , \tag{10}$$

where $\mu_j(\mathbf{J}_i)$ denotes j^{th} singular value of the Jacobian $\mathbf{J}_i$ corresponding to the i^{th} sample.

This simple strategy not only ensures the network Jacobian singular values to $\mathbf{J}$ be scaled to closer to 1 but also ensures the network output $\bar{\mathbf{a}}^K$ to be scaled down if the network is overconfident (network produces very large logits) or scaled up if the network is underconfident (network produces very small logits) which in turn improves signal propagation of $\partial \mathbf{p}(\beta)/\partial \bar{\mathbf{a}}^K$.

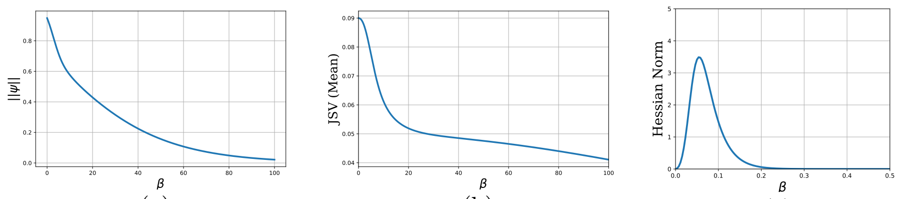
After this Jacobian based scaling, there can be a situation where the error signal is very small. To ensure that $\|\psi(\beta)\|_2 > \rho > 0$, we ensure that the softmax output $p_k(\beta)$ corresponding to the ground truth class k is at least ρ away from 1.

We now state it as a proposition to derive β given a lowerbound on $1 - p_k(\beta)$.

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

Proof. This is derived via a simple algebraic manipulation of softmax. Please refer to Appendix A.1. □

This β can be used together with the one computed in Eq. (10). We provide the pseudocode for our proposed PGD++ attack with NJS scaling in Algorithm 1. Similar approach can also be applied for FGSM++.



(a) (b) (c)

Fig. 3: *Plots to show how variation in* β *affects (a) error signal* $\psi(\beta)$*, (b) Jacobian of softmax,* i.e., $\partial \mathbf{p}(\beta)/\partial \bar{\mathbf{a}}^K$ *by taking a random correctly classified logits and (c) Hessian norm* $\|\partial^2 \ell(\beta)/\partial (\mathbf{x}^0)^2\|$ *on a random correctly classified image. Notice that, while* $\psi(\beta)$ *and* JSV *of* $\partial \mathbf{p}(\beta)/\partial \bar{\mathbf{a}}^K$ *behave similarly, Hessian clearly shows a concave behaviour and the maximum point occurs at a small value where both* $\psi(\beta)$ *and* $\partial \mathbf{p}(\beta)/\partial \bar{\mathbf{a}}^K$ *are non zero. Hessian is computed for* MD*-tanh-*S *network on CIFAR-10, ResNet-18 and the mean* JSV *of* $\mathbf{J} = \partial \mathbf{a}^K/\partial \mathbf{x}^0$ *for this image is* 45.14*. Therefore the input-output Jacobian* $\beta\,\mathbf{J}$ *has the mean* JSV *around* 2.5*.*

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: $\beta_1 = (M\,d)/\big(\sum_{i=1}^{M}\sum_{j=1}^{d}\mu_j(\mathbf{J}_i)\big)$    ▷ $\beta_1$ computed using Network Jacobian.
2: $\mathbf{x}^1 = P_\infty^\epsilon(\mathbf{x}^0 + \text{Uniform}(-1,1))$    ▷ 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    ▷ $\rho = 0.01$
7:         $\beta_2 = -\log(\rho/(d-1)(1-\rho))/\gamma$    ▷ $\gamma$ computed using Proposition 1
8:     $\ell = -\mathbf{y}^T \log(\text{softmax}(\beta_2\beta_1(f_{\mathbf{w}^*}(\mathbf{x}^t))))$
9:     $\mathbf{x}^{t+1} = P_\infty^\epsilon(\mathbf{x}^t + \eta\,\text{sign}(\nabla_{\mathbf{x}}\ell(\mathbf{x}^t)))$    ▷ Update Step with Projection
```

Notice that, this approach is simple and it adds negligible overhead to the standard PGD attacks. However, it has a hyperparameter ρ which is hand designed. To mitigate this, next we discuss a hyperparameter-free approach to obtain β.

5.2 Hessian Norm Scaling (HNS)

We now discuss another approach to obtain informative gradients. Our idea is to maximize the Frobenius norm of the Hessian of the loss with respect to the input, where the intuition is that if the Hessian norm is large, then the gradient $\partial\ell/\partial\mathbf{x}^0$ is sensitive to an infinitesimal change in $\mathbf{x}^0$. This means, the infinitesimal perturbation in the input is propagated in the forward pass to the last layer and propagated back to the input layer without attenuation (*i.e.*, the returned signal is not zero), assuming there are no randomized or non-differentiable components in the network. This clearly indicates that the network has good signal propagation

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: $\|\mathbf{x}^{T+1} - \mathbf{x}^0\|_\infty \le \epsilon$
1: $\mathbf{x}^1 = P^\epsilon_\infty(\mathbf{x}^0 + \text{Uniform}(-1,1))$    ▷ Random Initialization with Projection
2: $\beta^* = \text{argmax}_{\beta>0} \left\|\partial^2\ell(\beta)/\partial(\mathbf{x}^0)^2\right\|_F$    ▷ 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:     $\mathbf{x}^{t+1} = P^\epsilon_\infty(\mathbf{x}^t + \eta\, \text{sign}(\nabla_{\mathbf{x}}\ell(\mathbf{x}^t)))$    ▷ Update Step with Projection
```

as well as the error signals are not all zero. This objective can now be written as:

$$\beta^* = \underset{\beta>0}{\text{argmax}} \left\| \frac{\partial^2 \ell(\beta)}{\partial(\mathbf{x}^0)^2} \right\|_F = \underset{\beta>0}{\text{argmax}} \left\| \beta \left[\psi(\beta) \frac{\partial \mathbf{J}}{\partial \mathbf{x}^0} + \beta \left(\frac{\partial \mathbf{p}(\beta)}{\partial \bar{\mathbf{a}}^K} \mathbf{J} \right)^T \mathbf{J} \right] \right\|_F . \tag{11}$$

The derivation is provided in Appendix A.2. Note, since $\mathbf{J}$ does not depend on β, $\mathbf{J}$ and $\partial\mathbf{J}/\partial\mathbf{x}^0$ are computed only once, β is optimized using grid search as it involves only a single scalar. In fact, it is easy to see from the above equation that, when the Hessian is maximized, β cannot be zero. Similarly, $\psi(\beta)$ cannot be zero because if it is zero, then the prediction $\mathbf{p}(\beta)$ is one-hot encoding (Eq. (9)), consequently $\partial\mathbf{p}(\beta)/\partial\bar{\mathbf{a}}^K = \mathbf{0}$ and this cannot be a maximum for the Hessian norm. Hence, this ensures that $\|\psi(\beta^*)\|_2 > \rho$ for some $\rho > 0$ and β^* is bounded according to Proposition 1. Therefore, the maximum is obtained for a finite value of β. Even though, it is not clear how exactly this approach would affect the singular values of the input-output Jacobian ($\beta\,\mathbf{J}$), we know that they are finite and not zero. To better understand how the error signal, Jacobian of softmax and Hessian norm are influenced by β, we provide an example illustration in Fig. 3.

Nevertheless, there are some recent works [21,24] show that adversarial training makes the network (in fact the loss ℓ) linear and they hypothesize that linear networks would be robust to adversarial attacks. On the contrary, our idea of maximizing the Hessian, *i.e.*, increasing the nonlinearity of ℓ, could make the network more prone to adversarial attacks and we intend to exploit that. Our improved PGD attack using this Hessian based scaling is summarized in Algorithm 2.

6 Related Work

Neural Network Quantization. Recent works on NN quantization has been focused on different aspects such as quantizing parameters [1,2,4,7], gradients [29], loss aware quantization [14], and quantization for specialized hardware [8], to name a few. Note that, parameter quantization methods trivially extend to quantization of activations and/or gradients [15,29]. Parameter quantization is usually formulated as a constraint optimization and minimized via a modified version of projected stochastic gradient descent. In this, the works [1,2,4,7] vary based on the constraint set, the projection used and the procedure used for backpropagation through the projection operator. In this work, we only study the robustness evaluation of recent quantization methods on the extreme case of binarization and we believe it extends to multibit quantization as well.

Adversarial Attacks and Robustness of Binary Networks. Adversarial examples are first observed in [27] and subsequently efficient gradient based attacks such as FGSM [12] and PGD [19] are introduced. There exist recent stronger attacks such as [6,9], however, compared to PGD, they are much slower to be used for adversarial training in practice. Some recent works focus on adversarial robustness of BNNs [5,10,17,18], however, a strong consensus on the robustness properties of quantized networks is lacking. In particular, while [10] claims parameter quantized networks are robust to gradient based attacks based on empirical evidence, [18] shows activation quantized networks are vulnerable to such attacks and proposes a defense strategy assuming the parameters are floating-point. Differently, [17] proposes a combinatorial attack hinting that activation quantized networks would have obfuscated gradients issue. In short, although it has been hinted that there might be some sort of gradient masking in BNNs (especially in activation quantized networks), a thorough understanding is lacking on whether BNNs are robust, if not what is the reason for inferior performance of gradient based attacks on binary networks. We answer this question in this paper and introduce improved PGD attacks.

7 Experiments

In this section, we evaluate floating point networks (REF), parameter quantized networks (BC, PQ, PMF, MD-tanh-S) [7,4,1,2] and weight and activation quantized network (BNN-WAQ) [16]. Note that, while all the parameters are binarized in BC, PMF and MD-tanh-S, PQ and BNN-WAQ are binarized according to the original setup where biases, batchnorm and last layer parameters are kept floating-point. We evaluate our two PGD++ variants corresponding to Hessian Norm Scaling (HNS) and Network Jacobian Scaling (NJS) on CIFAR-10 and CIFAR-100 datasets with VGG-16 and ResNet-18 architectures. Briefly, our results indicate that both of our proposed attack variants yield attack success rate much higher than original PGD attacks not only on L_∞ bounded attack but also on L_2 bounded attacks on both floating point networks and binarized networks. Our proposed PGD++ variants also reduce PGD adversarial accuracy of adversarially trained floating point and adversarially trained binarized neural networks. Among our variants, even though they perform similarly in our experiments, Hessian based scaling (HNS) outperforms Jacobian based scaling (NJS) in most of the cases and this difference is significant for one step FGSM attacks. This indicates that nonlinearity of the network indeed has some relationship to its adversarial robustness.

We use the state of the art models trained for binary quantization from respective methods. For our HNS variant, we sweep β from a range such that the hessian norm is maximized for each image, as explained in Appendix B. For our NJS variant, we set the value of $\rho = 0.01$. In fact, our attacks are not very

Methods	CIFAR10						CIFAR100					
	ResNet-18			VGG-16			ResNet-18			VGG-16		
	PGD	PGD++		PGD	PGD++		PGD	PGD++		PGD	PGD++	
		HNS	NJS		HNS	NJS		HNS	NJS		HNS	NJS
REF	0.00	0.00	0.00	0.04	0.00	0.00	0.14	**0.00**	0.14	1.53	**0.25**	0.95
BC [7]	4.40	**0.00**	**0.00**	9.28	**0.36**	0.37	3.60	**0.00**	1.33	1.84	**0.04**	1.57
PQ [4]	22.49	**0.01**	**0.01**	23.41	**0.00**	**0.00**	4.80	**0.23**	3.09	1.41	**0.14**	1.39
PMF [1]	33.02	**0.00**	**0.00**	54.11	**0.00**	**0.00**	6.22	**0.05**	2.11	8.34	**0.21**	3.06
MD-tanh-S [2]	26.98	**0.00**	**0.00**	47.32	0.02	**0.00**	8.23	**0.00**	2.45	17.44	**0.16**	0.88
BNN-WAQ[4][15]	8.57	0.04	**0.03**	78.01	0.02	**0.01**	-	-	-	-	-	-

Table 2: *Adversarial accuracy on the test set for binary neural networks using different methods for quantization using original L_∞ bounded* PGD *attack and* PGD++ *attack with* NJS *and* HNS*.*

Methods	CIFAR10						CIFAR100					
	ResNet-18			VGG-16			ResNet-18			VGG-16		
	FGSM	FGSM++		FGSM	FGSM++		FGSM	FGSM++		FGSM	FGSM++	
		HNS	NJS		HNS	NJS		HNS	NJS		HNS	NJS
REF	7.62	**5.35**	5.55	11.01	**9.66**	10.04	9.06	**2.70**	9.23	16.28	**9.19**	17.24
BC [7]	11.15	**3.61**	3.77	27.38	5.04	**4.96**	22.67	**2.33**	9.86	11.38	**4.00**	11.02
PQ [4]	52.97	**4.24**	4.50	27.46	5.43	**5.38**	23.00	**3.83**	13.30	9.38	**2.68**	9.40
PMF [1]	48.65	**3.19**	3.22	54.87	**5.15**	5.19	23.11	**3.78**	13.58	25.09	**5.10**	15.73
MD-tanh-S [2]	40.49	**2.51**	3.46	57.55	**3.43**	4.00	25.22	**1.80**	14.08	19.82	**1.76**	7.98
BNN-WAQ [15]	40.84	**19.09**	19.46	79.92	**15.39**	15.96	-	-	-	-	-	-

Table 3: *Adversarial accuracy on the test set for binary neural networks using different methods for quantization using original* FGSM *attack and* FGSM++ *attack with* NJS *and* HNS*.*

sensitive to ρ and we provide the ablation study in the Appendix. Our algorithm is implemented in PyTorch [22] and the experiments are performed on NVIDIA Tesla-P100 GPUs. Our code will be released upon publication.

L_∞ bounded Attacks. Attack details are: **CIFAR-10**: perturbation bound of $\epsilon = 8$ pixels, step size $\eta = 2$ and number of iterations $T = 20$, **CIFAR-100**: perturbation bound of $\epsilon = 4$ pixels, step size $\eta = 1$ and number of iterations $T = 10$. We tested the original PGD as well as both versions (NJS and HNS) of improved PGD++, on CIFAR-10/100 datasets with ResNet-18 and VGG-16 networks and the adversarial accuracies are reported in Table 2. Our PGD++ variants consistently outperform original PGD on all binarized networks. Even being a gradient based attack, our proposed PGD++ variants can in fact reach adversarial accuracy close to 0 on CIFAR-10 dataset, demystifying the fake sense of robustness binarized networks tend to possess due to poor signal propagation.

Similarly, FGSM attack ($\eta = 8.0$ on CIFAR-10 and $\eta = 4.0$ on CIFAR-100) accuracies are reported in Table 3. Even in this one step attack, our modified versions perform well. We would like to point out such an improvement in above

[4] For BNN-WAQ [15], trained models could not be obtained on CIFAR-100. Also, BNN-WAQ was not evaluated on CIFAR-100 dataset in their paper.

Methods	CIFAR10						CIFAR100					
	ResNet-18			VGG-16			ResNet-18			VGG-16		
	PGD	PGD++		PGD	PGD++		PGD	PGD++		PGD	PGD++	
		HNS	NJS		HNS	NJS		HNS	NJS		HNS	NJS
REF	45.18	**0.05**	0.09	2.23	1.10	**0.78**	5.38	**0.15**	0.17	4.87	**1.38**	1.50
BC [7]	61.11	0.24	**0.18**	46.24	**1.58**	1.63	28.98	**0.38**	5.27	6.78	**0.58**	4.66
PQ [4]	69.57	0.50	**0.35**	27.33	**1.72**	2.25	29.57	**1.24**	9.21	5.05	**0.75**	5.10
PMF [1]	74.72	0.04	**0.02**	55.82	2.85	**2.71**	45.97	**0.71**	6.60	30.91	**1.61**	8.62
MD-tanh-S [2]	74.59	0.06	**0.05**	61.90	3.18	**0.35**	42.67	**0.26**	6.79	19.26	**0.63**	3.17
BNN-WAQ [15]	67.84	2.59	**2.33**	85.62	0.62	**0.49**	-	-	-	-	-	-

Table 4: *Adversarial accuracy on the test set for binary neural networks using different methods for quantization using original L_2 bounded* PGD *attack and* PGD++ *attack with* NJS *and* HNS*.*

Methods	FGSM	FGSM++	
		HNS	NJS
REF	62.38	**61.40**	61.43
BC [7]	53.91	**52.27**	52.90
GD-tanh [2]	56.13	**54.81**	55.54
MD-tanh-S [2]	55.10	**53.82**	54.74

(a)

Methods	PGD	PGD++	
		HNS	NJS
REF	48.73	48.54	**47.17**
BC [7]	41.29	**39.34**	39.35
GD-tanh [2]	42.77	42.30	**42.14**
MD-tanh-S [2]	41.34	**40.67**	40.76

(b)

Table 5: *Adversarial accuracy on the test set for adversarially trained floating and binary neural networks using different methods for quantization using (a) Original L_∞ bounded* FGSM *attack and* FGSM++ *attack with* NJS *and* HNS *and (b) Original L_∞ bounded* PGD *attack and* PGD++ *attack with* NJS *and* HNS*.*

two attacks is considerably interesting, knowing the fact that FGSM, PGD with L_∞ attacks only use the sign of the gradients so improved performance indicate, our temperature scaling indeed makes some zero elements in the gradient nonzero.

L_2 bounded Attacks. For L_2 bounded attacks the number of iterations are the same as L_∞ case, but $\eta = 15$ and $\epsilon = 120$ for CIFAR-10 and $\eta = 15$, and $\epsilon = 60$ for CIFAR-100 and the results are reported in Table 4. Similar to the L_∞ case, our variants consistently outperform standard L_2 bounded PGD attack on all the binarized networks. Particularly interesting, our PGD++ variants not only performs well on floating point networks but also even improve the adversarial attack success rate on them. This effectively expands the applicability of our PGD++ variants and encourages to consider signal propagation of any trained network to improve gradient based adversarial attacks.

Adversarially trained models. To further demonstrate the efficacy, we first adversarially trained the parameter quantized networks and floating point networks in a similar manner as in [19], using L_∞ bounded PGD with $T = 7$ iterations, $\eta = 2$ and $\epsilon = 8$. We then evaluate the adversarial accuracies using L_∞ bounded PGD and PGD++ attack with $T = 20$, $\eta = 2$, $\epsilon = 8$ on CIFAR-10 dataset using ResNet-18 and the results are reported in Table 5. The adversarial accuracy results on adversarially trained binary and floating point networks further strengthens the usefulness of our proposed PGD++ variants.

8 Discussion

In this work, we have shown that both parameters quantized and parameters along with activation quantized networks tend to show a fake sense of robustness on gradient based attacks due to poor signal propagation. To tackle this issue, we introduced our two variants of PGD++ adversarial attack, namely NJS and HNS. Our proposed PGD++ variants not only possess near-complete success rate on binarized networks but also outperform standard L_∞ and L_2 bounded PGD attacks on floating-point networks. Since PGD has become a standard attack for adversarial training, our stronger PGD++ variants could provide a future scope of extending proposed PGD++ for improved robustness. In future, we also intend to focus more on improving the robustness of the binarized neural networks.

9 Acknowledgements

This work was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016). We acknowledge the Data61, CSIRO for their support and thank Puneet Dokania for useful discussions.

Appendices

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

A Derivations

A.1 Deriving β given a lowerbound on $1 - p_k(\beta)$

Proposition 2. Let $\mathbf{a}^K \in \mathbb{R}^d$ with $d > 1$ and $a_1^K \geq a_2^K \geq \ldots \geq a_d^K$ and $a_1^K - a_d^K = \gamma$. For a given $0 < \rho < (d-1)/d$, there exists a $\beta > 0$ such that $1 - \mathrm{softmax}(\beta a_1^K) > \rho$, 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 $1 - \mathrm{softmax}(\beta a_1^K) > \rho$.

$$\begin{aligned} 1 - \mathrm{softmax}(\beta a_1^K) &> \rho\ , \qquad (12)\\ \mathrm{softmax}(\beta a_1^K) &< 1 - \rho\ ,\\ \exp(\beta a_1^K)/\sum_{\lambda=1}^{d}\exp(\beta a_\lambda^K) &< 1-\rho\ ,\\ 1/\big(1 + \sum_{\lambda=2}^{d}\exp(\beta(a_\lambda^K - a_1^K))\big) &< 1-\rho\ . \end{aligned}$$

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

$$1/\big(1+\sum_{\lambda=2}^{d}\exp(\beta(a_\lambda^K - a_1^K))\big) \leq 1/\big(1+\sum_{\lambda=2}^{d}\exp(-\beta\gamma)\big) \ . \tag{13}$$

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

$$\begin{aligned}1/\big(1+\sum_{\lambda=2}^{d}\exp(-\beta\gamma)\big) &< 1-\rho \ , \quad a_1^K - a_\lambda^K \leq \gamma \text{ for all } \lambda > 1 \ , \\ 1/\big(1+(d-1)\exp(-\beta\gamma)\big) &< 1-\rho \ , \\ \exp(-\beta\gamma) &> \rho/(d-1)(1-\rho) \ , \\ -\beta\gamma &> \log(\rho/(d-1)(1-\rho)) \ , \quad \text{exp is monotone} \ , \\ \beta &< -\log(\rho/(d-1)(1-\rho))/\gamma \ .\end{aligned} \tag{14}$$

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

A.2 Derivation of Hessian

We now derive the Hessian of the input mentioned in Eq. (11) of the paper. The input gradients can be written as:

$$\frac{\partial \ell(\beta)}{\partial \mathbf{x}^0} = \frac{\partial \ell(\beta)}{\partial \mathbf{p}(\beta)}\frac{\partial \mathbf{p}(\beta)}{\partial \bar{\mathbf{a}}^K(\beta)}\beta\mathbf{J} = \psi(\beta)\beta\mathbf{J} \ . \tag{15}$$

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

$$\begin{aligned}\frac{\partial^2 \ell(\beta)}{\partial (\mathbf{x}^0)^2} &= \beta\left[\psi(\beta)\frac{\partial \mathbf{J}}{\partial \mathbf{x}^0} + \left(\frac{\partial \psi(\beta)}{\partial \mathbf{x}^0}\right)^T \mathbf{J}\right] \ , \\ &= \beta\left[\psi(\beta)\frac{\partial \mathbf{J}}{\partial \mathbf{x}^0} + \left(\frac{\partial \mathbf{p}(\beta)}{\partial \mathbf{x}^0}\right)^T \mathbf{J}\right] \ , \quad \psi(\beta) = -(\mathbf{y}-\mathbf{p}(\beta))^T \ , \\ &= \beta\left[\psi(\beta)\frac{\partial \mathbf{J}}{\partial \mathbf{x}^0} + \beta\left(\frac{\partial \mathbf{p}(\beta)}{\partial \bar{\mathbf{a}}^K}\mathbf{J}\right)^T \mathbf{J}\right] \ .\end{aligned} \tag{16}$$

B Additional Experiments

In this section we first provide experimental details of our HNS variant of PGD++ and then some ablation studies.

Methods	PGD++ (NJS)					
	$\rho = 1e-05$	$\rho = 1e-04$	$\rho = 1e-03$	$\rho = 1e-02$	$\rho = 1e-01$	$\rho = 2e-01$
REF	0.00	0.00	0.00	0.00	0.00	0.00
BC [7]	0.00	0.00	0.00	0.00	0.00	0.01
PQ [4]	0.01	0.01	0.01	0.01	0.01	0.01
PMF [1]	0.00	0.00	0.00	0.00	0.00	0.00
MD-tanh-S [2]	0.00	0.00	0.00	0.00	0.00	0.00
BNN-WAQ [15]	0.15	0.08	0.04	0.03	0.04	0.02

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

B.1 Experimental Details for HNS

For PGD++ with HNS variant, we maximize Frobenius norm of Hessian with respect to the input as specified in Eq. (11) 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.

B.2 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.

B.3 Signal Propagation and Input Gradient Analysis using NJS and HNS

We first provide an example illustration in Fig. 4 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. 4 (b) that within an optimal range of β,

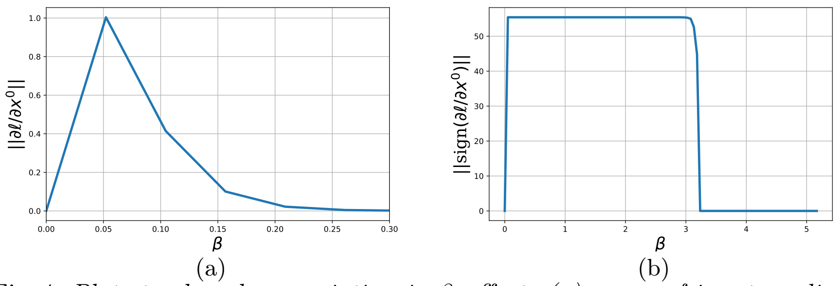


Fig. 4: *Plots to show how variation in β affects (a) norm of input gradient,* i.e., $\|\partial\ell(\beta)/\partial\mathbf{x}^0\|_2$*, (b) norm of sign of input gradient,* i.e., $\|sign(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$ *on a random correctly classified image. Notice that, both input gradient and signed input gradient norm behave similarly, showing a concave behaviour. This plot is computed for* MD*-tanh-*S *network on CIFAR-10, ResNet-18. (b) clearly illustrates how optimum β can avoid vanishing gradient issue since $\|sign(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$ will only be zero if input gradient matrix has only zeros.*

gradient vanishing issue can be avoided. If $\beta \to 0$ or $\beta \to \infty$, it changes all the values in input gradient matrix to zero and inturn $\|\text{sign}(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2 = 0$.

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.

B.4 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, even for robust (locally linear) [21,24] 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 exact same objective while trying to estimate β for each example.

Methods		REF	Adv. Train	BC [7]	PQ [4]	PMF [1]	MD-tanh-S [2]	BNN-WAQ [15]
JSV (Mean)	Orig.	8.09e+00	5.15e−01	1.61e+01	2.34e+01	4.46e+01	3.53e+01	1.11e+00
	NJS	9.51e−01	5.70e−01	9.65e−01	1.00e+00	1.01e+00	9.95e−01	2.24e−01
	HNS	2.38e+00	6.11e+00	1.25e+00	3.09e+00	4.43e+00	1.19e+01	4.65e+00
JSV (Std.)	Orig.	6.27e+00	4.10e−01	1.88e+01	2.35e+01	1.11e+02	3.53e+01	1.97e+00
	NJS	7.58e−01	6.34e−01	8.62e−01	1.02e+00	2.38e+00	9.71e−01	6.73e−01
	HNS	4.41e+00	5.34e+02	2.06e+01	7.70e+00	1.46e+01	2.13e+02	1.24e+02
$\|\psi\|_2$	Orig.	9.08e−03	2.33e−01	1.18e−02	6.75e−03	8.50e−03	6.20e−03	9.46e−03
	NJS	4.66e−01	2.35e−01	5.08e−01	5.35e−01	6.65e−01	5.37e−01	1.20e−01
	HNS	1.48e−01	2.57e−01	2.18e−01	2.28e−01	2.17e−01	2.07e−01	2.44e−01
$\|\partial\ell/\partial\mathbf{x}^0\|_2$	Orig.	2.42e−01	8.52e−02	3.04e−01	1.57e−01	3.08e−01	2.27e−01	6.33e−02
	NJS	9.52e−01	1.10e−01	7.90e−01	6.26e−01	8.15e−01	8.91e−01	1.24e−01
	HNS	7.49e−01	8.18e−01	1.25e−01	7.05e−01	1.19e−01	3.70e−01	2.70e−01
$\|\text{sign}(\frac{\partial\ell}{\partial\mathbf{x}^0})\|_2$	Orig.	5.55e+01	5.54e+01	5.48e+01	5.48e+01	4.99e+01	4.39e+01	5.55e+01
	NJS	5.55e+01	5.54e+01	5.55e+01	5.55e+01	5.55e+01	5.55e+01	5.55e+01
	HNS	5.55e+01	5.54e+01	5.55e+01	5.55e+01	5.55e+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 $\|sign(\partial\ell/\partial\mathbf{x}^0)\|_2$ 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 $\|sign(\partial\ell(\beta)/\partial\mathbf{x}^0)\|_2$ 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.*

Methods	PGD++					
	$\rho = 1e-15$	$\rho = 1e-09$	$\rho = 1e-05$	$\rho = 1e-01$	$\rho = 2e-01$	$\rho = 5e-01$
REF	0.0	0.0	0.0	0.0	0.0	0.0
MD-tanh-S	9.61	0.04	0.0	0.0	0.0	0.0
GD-tanh	24.26	3.81	0.0	0.0	0.0	0.0
REF*	48.18	**47.66**	48.00	53.09	54.58	57.57
MD-tanh-S*	40.66	**40.01**	40.04	45.09	46.57	49.72
GD-tanh*	42.55	**41.97**	42.06	47.72	49.21	52.45

Table 8: *Adversarial accuracy on the test set for adversarially trained networks and binary neural networks with different methods for quantization 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 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.*

References

1. Ajanthan, T., Dokania, P.K., Hartley, R., Torr, P.H.: Proximal mean-field for neural network quantization. ICCV (2019)
2. Ajanthan, T., Gupta, K., Torr, P.H., Hartley, R., Dokania, P.K.: Mirror descent view for neural network quantization. arXiv preprint arXiv:1910.08237 (2019)
3. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv preprint arXiv:1802.00420 (2018)
4. Bai, Y., Wang, Y.X., Liberty, E.: Proxquant: Quantized neural networks via proximal operators. ICLR (2019)
5. Bernhard, R., Moellic, P.A., Dutertre, J.M.: Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks. In: 2019 International Conference on Cyberworlds (CW). pp. 308–315. IEEE (2019)
6. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE Symposium on Security and Privacy (2017)
7. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. NeurIPS (2015)
8. Esser, S.K., Appuswamy, R., Merolla, P.A., Arthur, J.V., Modha, D.S.: Backpropagation for energy-efficient neuromorphic computing. NeurIPS (2015)
9. Finlay, C., Pooladian, A.A., Oberman, A.: The logbarrier adversarial attack: making effective use of decision boundary information. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4862–4870 (2019)
10. Galloway, A., Taylor, G.W., Moussa, M.: Attacking binarized neural networks. In: International Conference on Learning Representations (2018)
11. Glorot, X., Bengio, Y.: 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)
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
13. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1321–1330. JMLR. org (2017)
14. Hou, L., Yao, Q., Kwok, J.T.: Loss-aware binarization of deep networks. ICLR (2017)
15. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. NeurIPS (2016)
16. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: Training neural networks with low precision weights and activations. JMLR (2017)
17. Khalil, E.B., Gupta, A., Dilkina, B.: Combinatorial attacks on binarized neural networks. In: International Conference on Learning Representations (2019)
18. Lin, J., Gan, C., Han, S.: Defensive quantization: When efficiency meets robustness. In: International Conference on Learning Representations (2019)
19. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
20. McDonnell, M.D.: Training wide residual networks for deployment using a single bit for each weight. ICLR (2018)
21. Moosavi-Dezfooli, S.M., Fawzi, A., Uesato, J., Frossard, P.: Robustness via curvature regularization, and vice versa. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9078–9086 (2019)

22. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch (2017)
23. Pennington, J., Schoenholz, S., Ganguli, S.: Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In: Advances in neural information processing systems. pp. 4785–4795 (2017)
24. Qin, C., Martens, J., Gowal, S., Krishnan, D., Dvijotham, K., Fawzi, A., De, S., Stanforth, R., Kohli, P.: Adversarial robustness through local linearization. In: Advances in Neural Information Processing Systems. pp. 13824–13833 (2019)
25. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. ECCV (2016)
26. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120 (2013)
27. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014)
28. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204 (2017)
29. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. CoRR (2016)