An Integrated Approach to Produce Robust Deep Neural Network Models with High Efficiency

Zhijian Li[1(✉)], Bao Wang[2], and Jack Xin[1]

[1] University of California, Irvine, CA 92697, USA
{zhijil2,jack.xin}@uci.edu
[2] The University of Utah, Salt Lake City, UT 84112, USA

Abstract. Deep Neural Networks (DNNs) need to be both efficient and robust for practical uses. Quantization and structure simplification are promising ways to adapt DNNs to mobile devices, and adversarial training is one of the most successful methods to train robust DNNs. In this work, we aim to realize both advantages by applying a convergent relaxation quantization algorithm, i.e., Binary-Relax (BR), to an adversarially trained robust model, i.e. the ResNets Ensemble via Feynman-Kac Formalism (EnResNet). We discover that high-precision quantization, such as ternary (tnn) or 4-bit, produces sparse DNNs. However, this sparsity is unstructured under adversarial training. To solve the problems that adversarial training jeopardizes DNNs' accuracy on clean images and break the structure of sparsity, we design a trade-off loss function that helps DNNs preserve natural accuracy and improve channel sparsity. With our newly designed trade-off loss function, we achieve both goals with no reduction of resistance under weak attacks and very minor reduction of resistance under strong adversarial attacks. Together with our model and algorithm selections and loss function design, we provide an integrated approach to produce robust DNNs with high efficiency and accuracy. Furthermore, we provide a missing benchmark on robustness of quantized models.

Keywords: Quantization · Channel pruning · Adversarial training

1 Introduction

1.1 Background

Deep Neural Networks (DNNs) have achieved significant success in computer vision and natural language processing. Especially, the residual network (ResNet)[8] has achieved remarkable performance on image classification and has become one of the most important neural network architectures in the current literature.

This work was partly supported by NSF Grants DMS-1854434, DMS-1924548, DMS-1952644, DMS-1924935, and DMS-1952339.

G. Nicosia et al. (Eds.): LOD 2021, LNCS 13164, pp. 451–465, 2022.
https://doi.org/10.1007/978-3-030-95470-3_34

Despite the tremendous success of DNNs, researchers still try to strengthen two properties of DNNs, robustness and efficiency. In particular, for security-critic and on the edge applications. Robustness keeps the model accurate under small adversarial perturbation of input images, and efficiency enables us to fit DNNs into embedded system, such as smartphone. Many adversarial defense algorithms [7,11,14,25] have been proposed to improve the robustness of DNNs. Among them, adversarial training is one of the most effective and powerful methods. On the other hand, quantization [4,16], producing models with low-precision weights, and structured simplification, such as channel pruning [9,26], are promising ways to make models computationally efficient.

2 Related Work

2.1 Binary Quantization

Based on the binary-connect (BC) [24] proposed an improvement of BC called Binary-Relax, which makes the weights converge to the binary weights from the floating point weights gradually. Theoretically, [24] provided the convergence analysis of BC, and [12] presented an ergodic error bound of BC. The space of m-bit quantized weights $\mathcal{Q} \subset \mathbb{R}^n$ is a union of disjoint one-dimensional subspaces of R^n. When $m \geq 2$, it is formulated as:

$$\mathcal{Q} = \mathbb{R}_+ \times \{0, \pm 1, \pm 2, \cdots, \pm 2^{m-1}\}^n = \bigsqcup_{l=1}^{p} \mathcal{A}_l$$

where $\mathbb{R}_+$ is a float scalar. For the special binary case of $m = 1$, we have $\mathcal{Q} = \mathbb{R}_+ \times \{\pm 1\}^n$.

We minimize our objective function in the subspace $\mathcal{Q}$. Hence, the problem of binarizing weights can be formulated in the following two equivalent forms:

– I.

$$\operatorname*{argmin}_{u \in \mathcal{Q}} \mathcal{L}(u)$$

– II.

$$\operatorname*{argmin}_{u \in \mathbb{R}^n} \mathcal{L}(u) + \chi_{\mathcal{Q}}(u) \quad \text{where} \quad \chi_{\mathcal{Q}}(u) = \begin{cases} 0 & u \in \mathcal{Q} \\ \infty & else \end{cases} \tag{1}$$

Based on the alternative form II, [24] relaxed the optimization problem to:

$$\operatorname*{argmin}_{u \in \mathbb{R}^n} \mathcal{L}(u) + \frac{\lambda}{2} dist(u, \mathcal{Q})^2 \tag{2}$$

Observing (2) converges to (1) as $\lambda \to \infty$, [24] proposed a relaxation of BC:

$$\begin{cases} w_{t+1} = w_t - \gamma \nabla \mathcal{L}_t(u_t) \\ u_{k+1} = \operatorname{argmin}_{u \in \mathbb{R}^n} \frac{1}{2} \|w_{t+1} - u\|^2 + \frac{\lambda}{2} dist(u, \mathcal{Q})^2 \end{cases}$$

It can be shown that u_{k+1} above has a closed-form solution

$$u_{t+1} = \frac{\lambda \text{proj}_{\mathcal{Q}}(w_{t+1}) + w_{t+1}}{\lambda + 1}$$

Algorithm 1. Binary-Relax quantization algorithm

Input: mini-batches $\{(\mathbf{x}_1, \mathbf{y}_1), \cdots (\mathbf{x}_m, \mathbf{y}_m)\}$, $\lambda_0 = 1$, growth rate $\rho > 1$, learning rate γ, initial float weight w_0, initial binary weight $u_0 = w_0$, cut-off epoch M
Output: a sequence of binary weights $\{u_t\}$
for $t = 1, \cdots, N$ **do** ▷ N is the number of epochs
  if $t < M$ **then**
    for $k = 1, \cdots, m$ **do**
      $w_t = w_{t-1} - \gamma_t \nabla_k \mathcal{L}(u_{t-1})$
      $u_t = \frac{\lambda_t \cdot \text{Proj}_{\mathcal{Q}}(w_t) + w_t}{\lambda_t + 1}$
      $\lambda_{t+1} = \rho \cdot \lambda_t$
  else
    for $k = 1, \cdots, m$ **do**
      $w_t = w_{t-1} - \gamma_t \nabla \mathcal{L}(u_{t-1})$
      $u_t = \text{Proj}_{\mathcal{Q}}(w_t)$ ▷ This is precisely Binary-Connect
return quantized weights u_N

2.2 Adversarial Attacks

As [18] discovered the limited continuity of DNNs' input-output mapping, the outputs of DNNs can be changed by adding imperceptible adversarial perturbations to input data. The methods that generate perturbed data can be adversarial attacks, and the generated perturbed data are called adversarial examples. In this work, we focus on three benchmark adversarial attacks Fast gradient sign method (FGSM) [6], iterative FGSM (IFGSM), and Carlini and Wagner method (C&W) [2]. In this study, we denote attacks FGSM, IFGSM, and CW to be A_1, A_2, and A_3 respectively.

2.3 Adversarial Training

[5] rigorously established a rigorous benchmark to evaluate the robustness of machine learning models and investigated almost all current popular adversarial defense algorithms. They conclude that adversarially trained models are more robust models with other types of defense methods. [25] also shows that adversarial training is more powerful than other methods such as gradient mask and gradient regularization [11]. While a natural training has objective function:

$$\mathcal{L}(\omega) = \frac{1}{N} \sum_{i=1}^{N} l\big(f(w, \mathbf{x}_i), y_i\big) \tag{3}$$

Adversarial training [1,14] generates perturbed input data and train the model to stay stable under adversarial examples. It has the following objective function:

$$\mathcal{L}(w) = \frac{1}{N}\sum_{n=1}^{N} \max_{\tilde{x}_n \in D_n} l(f(w, \tilde{x}_n), y_n) \tag{4}$$

where $D_n = \{x | \|x - x_n\|_\infty < \delta\}$. l and f are the loss function and the DNN respectively. In this work, we denote (3) to be $\mathcal{R}_{nat}$ and (4) to be $\mathcal{R}_{rob}$. A widely used method to practically find $\tilde{x}_n$ is the projected gradient descent (PGD) [14]. [17] investigated the properties of the objective function of the adversarial training, and [22] provided convergence analysis of adversarial training based on the previous results.

Feynman-Kac Formalism Principled Robust DNNs: Neural ordinary differential equations (ODEs) [3] are a class of DNNs that use an ODE to describe the data flow of each input data. Instead of focusing on modeling the data flow of each individual input data, [13,20,21] use a transport equation (TE) to model the flow for the whole input distribution. In particular, from the TE viewpoint, [21] modeled training ResNet as finding the optimal control of the following TE

$$\begin{cases} \frac{\partial u}{\partial t}(\boldsymbol{x}, t) + G(\boldsymbol{x}, \boldsymbol{w}(t)) \cdot \nabla u(\boldsymbol{x}, t) = 0, \quad \boldsymbol{x} \in \mathbb{R}^d, \\ u(\boldsymbol{x}, 1) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d, \\ u(\boldsymbol{x}_i, 0) = y_i, \quad \boldsymbol{x}_i \in T, \quad \text{with } T \text{ being the training set.} \end{cases} \tag{5}$$

where $G(\boldsymbol{x}, \boldsymbol{w}(t))$ encodes the architecture and weights of the underlying ResNet, $u(\boldsymbol{x}, 0)$ serves as the classifier, $g(\boldsymbol{x})$ is the output activation of ResNet, and y_i is the label of $\boldsymbol{x}_i$.

Based on Eq. (5), [21] interpreted adversarial vulnerability of ResNet as arising from the irregularity of $u(\boldsymbol{x}, 0)$ of the above TE. To enhance $u(\boldsymbol{x}, 0)$'s regularity, they added a diffusion term, $\frac{1}{2}\sigma^2 \Delta u(\boldsymbol{x}, t)$, to the governing equation of (5) which results to the convection-diffusion equation (CDE). By the Feynman-Kac formula, $u(\boldsymbol{x}, 0)$ of the CDE can be approximated by the following two steps:

- Modify ResNet by injecting Gaussian noise to each residual mapping.
- Average the output of n jointly trained modified ResNets, and denote it as En$_n$ResNet.

[21] have noticed that EnResNet, comparing to ResNet with the same size, can significantly improve both natural and robust accuracies of the adversarially trained DNNs. In this work, we leverage the robust advantage of EnResNet to push the robustness limit of the quantized adversarially trained DNNs.

TRADES: It is well-known that adversarial training will significantly reduce the accuracy of models on clean images. For example, ResNet20 can achieve about 92% accuracy on CIFAR10 dataset. However, under the PGD training using $\epsilon = 0.031$ with step-size 0.007 and the number of iterations 10, it only has 76% accuracy on clean images of CIFAR10. [25] designed a trade-off loss function,

TRADES, that balances the natural accuracy and adversarial accuracy. The main idea of TRADES is to minimize the difference of adversarial error and the natural error:

$$\min_f \mathbb{E}\Big(l(f(X), Y) + \max_{X' \in \mathbb{B}(X,\epsilon)} l\big(f(X), f(X')\lambda\big)\Big) \tag{6}$$

TRADES is considered to be the state-of-the-art adversarial defense method, it outperforms most defense methods in both robust accuracy and natural accuracy. [5] investigated various defense methods, and TRADES together with PGD training outperforms all other methods they tested. In this work, we will design our own trade-off function that works with PGD training. TRADES will provide important baselines for us.

3 Quantization of EnResNet

We know that the accuracy of a quantized model will be lower than its counterpart with floating point weights because of loss of precision. However, we want to know that whether a quantized model is more vulnerable than its float equivalent under adversarial attacks? In this section, we study this question by comparing the accuracy drops of the natural accuracy and robust accuracy from float weights to binary weights. Meanwhile, we also investigate the performances between two quantization methods BC and BR.

3.1 Experimental Setup

Dataset. We use one of the most popular datasets CIFAR-10 [10] to evaluate the quantized models, as it would be convenient to compare it with the float models used in [21,25].

Baseline. Our baseline model is the regular ResNet. To our best knowledge, there has been work done before that investigated the robustness of models with quantized weights, so we do not have an expected accuracy to beat. Hence, our goals are to compare the robustness of binarized ResNet and binarized EnResNet and to see how close the accuracy of quantized models can be to the float models in [21,25].

Evaluation. We evaluate both natural accuracy and robust accuracy for quantized adversarial trained models. We examine the robustness of models FGSM (A_1), IFGSM (A_2), and C&W (A_3). In our recording, N denotes the natural accuracy (accuracy on clean images) of models. For FGSM, we use $\epsilon = 0.031$ as almost all works share this value for FGSM. For IFGSM, we use $\alpha = 1/255$, $\epsilon = 0.031$, and number of iterations 20. For C&W, we have learning rate 0.0006 and the number of iterations 50.

Algorithm and Projection. We set BC as our baseline algorithm, and we want to examine that whether the advantage of the relaxed algorithm 13 in [24]

is preserved under adversarial training. In both BC and BR, we use the wildly used binarizing projection proposed by [16], namely:

$$\text{Proj}_{\mathcal{Q}}(w) = \mathbb{E}[|w|] \cdot sign(w) = \frac{||w||_1}{n} \cdot sign(w)$$

where $sign(\cdot)$ is the component-wise sign function and n is the dimension of weights.

3.2 Result

First, we verify that the Ensemble ResNet consistently outperforms ResNet when binarized. We adversarially train two sets of EnResNet and ResNet with the similar number of parameters. As shown in Table 1, EnResNet has much higher robust accuracy for both float and quantized models.

Second, we investigate the performances of Binary-Connect method (BC) and Binary-Relax method (BR). We verify that BR outperforms BC (Table 2). A quantized model trained via BR provides higher natural accuracy and robust accuracy. As a consequence, we use this relaxed method to quantize DNNs in all subsequent experiments in this paper.

Table 1. Ensemble ResNet vs ResNets. We verify that EnResNet outperforms ResNet with the similar number of parameters with both float and binary weights.

Net(#params)	Model	N	A_1	A_2	A_3
En_1ResNet20 (0.27M)	BR	**69.60**%	**47.17**%	**43.89**%	**58.79**%
ResNet20 (0.27M)	BR	66.81%	43.37%	40.72%	52.14%
En_2ResNet20 (0.54M)	BR	**72.58**%	**49.29**%	**44.72**%	**60.36**%
ResNet34 (0.56M)	BR	70.31%	46.42%	43.26%	54.75%

Table 2. Binary Connect vs Binary Relax. Models quantized by Binary-Relax have higher natural accuracy and adversarial accuracies than those quantized by Binary-Connect

Model	Quant	N	A_1	A_2	A_3
En_1ResNet20	Float	78.31%	56.64%	49.00%	66.84%
	BC	68.84%	46.31%	42.45%	58.52%
	BR	**69.60**%	**47.17**%	**43.89**%	**58.79**%
En_2ResNet20	Float	80.10%	57.48%	49.55%	66.73%
	BC	71.48%	47.83%	43.03%	59.09%
	BR	**72.58**%	**49.29**%	**44.72**%	**60.36**%
En_5ResNet20	Float	80.64%	58.14%	50.32%	66.96%
	BC	75.54%	51.03%	46.01%	60.92%
	BR	**75.40**%	**51.60**%	**46.91**%	**61.52**%

4 Trade-Off Between Robust Accuracy and Natural Accuracy

4.1 Previous Work and Our Methodology

It is known that adversarial training will decrease the accuracy for classifying the clean input data. This phenomenon is verified both theoretically [19] and experimentally [11,21,25] by researchers. [25] proposed a trade-off loss function (6) for robust training to balance the adversarial accuracy and natural accuracy. In practice, it is formulated as following:

$$\mathcal{L} = \mathcal{L}_{nat} + \beta \cdot \frac{1}{N} \sum_{n=1}^{N} l\big(f(x_n), f(\tilde{x}_n)\big) \tag{7}$$

Motivated by [25], we study the following trade-off loss function for our quantized models:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{nat} + \beta \cdot L_{rob} \tag{8}$$

Note that adversarial training is a special case $\alpha = 0$, $\beta = 1$ in (8). Loss function (7), TRADES, improves the robustness of models by pushing the decision boundary away from original data points, clean images in this case. However, intuitively, if the model classifies a original data point wrong, the second term of (7) will still try to extend this decision boundary, which can prevent the first term of the loss function from leading the model to the correct classification. In this section, we will experimentally compare (7) and (8) and theoretically analyze the difference between them.

Table 3. Comparison of loss function. Trade-off loss function (7) outperforms TRADES (8) in most cases.

Model	Loss	N	A_1	A_2	A_3
En_1ResNet20	(7) ($\beta = 1$)	**84.49%**	45.96%	34.81%	51.94%
En_1ResNet20	(8) ($\alpha = 1, \beta = 1$)	83.47%	**54.46%**	**43.86%**	**64.04%**
En_1ResNet20	(7) ($\beta = 4$)	80.05%	51.24%	45.43%	58.85%
En_1ResNet20	(8) ($\alpha = 1, \beta = 4$)	**80.91%**	**55.92%**	**47.17%**	**66.53%**
En_1ResNet20	(7) ($\beta = 8$)	75.82%	51.63%	46.95%	59.31%
En_1ResNet20	(8) ($\alpha = 1, \beta = 8$)	**79.31%**	**56.28%**	**48.02%**	**66.07%**

4.2 Experiment and Result

To compare the performances of two loss functions, we choose our neural network and dataset to be En_1ResNet20 and CIFAR-10 respectively. Based on [25], who studied β of (7) in the range $[1, 10]$, we vary the trade-off parameter β, the

weight of adversarial loss, in the set $[1, 4, 8]$ to emphasize the robustness in different levels.

The experiment results are listed in Table 3. We observe that, when the natural loss and the adversarial loss are equally treated, the model trained by (7) has higher natural accuracy while (8) makes its model more robust. As the trade-off parameter β increases, natural accuracy of the model trained (7) drops rapidly, while (8) trades a relatively smaller amount of natural loss for robustness. As a result, when $\beta = 4$ and $\beta = 8$, the model trained by (8) has both higher natural accuracy and higher adversarial accuracy. Hence, we say that (8) has better trade-off efficiency than (7)

4.3 Analysis of Trade-Off Functions

In this subsection, we present a theoretical analysis that why (8) outperforms (7). Let us consider the binary classification case, where have our samples $(\mathbf{x}, y) \in \mathcal{X} \times \{-1, 1\}$. Let $f : \mathcal{X} \to \mathbb{R}$ be a classifier and $\sigma(\cdot)$ be an activation function. Then our prediction for a sample is $\hat{y} = sign(f(\mathbf{x}))$ and the corresponding score is $\sigma(f(\mathbf{x}))$. Above is the theoretical setting provided by [25]. Then, we have the errors $\mathcal{R}_\phi(f)$ and $\mathcal{R}^*_\phi(f)$ corresponding to (7) and (8) respectively:

$$\mathcal{R}_\phi(f) = \mathbb{E}[\phi(\sigma \circ f(\mathbf{x}) \cdot y)] + \beta \cdot \mathbb{E}[\phi(\sigma \circ f(\mathbf{x}) \cdot \sigma \circ f(\mathbf{x}'))]$$

$$\mathcal{R}^*_\phi(f) = \alpha \cdot \mathbb{E}[\phi(\sigma \circ f(\mathbf{x}) \cdot y)] + \beta \cdot \mathbb{E}[\phi(\sigma \circ f(\mathbf{x}') \cdot y)]$$

Table 4. Trade-off loss function for binarized models with different parameters. When $\alpha = 1$ and $\beta = 8$, models can achieve about the same accuracies under FGSM and C&W attacks as solely adversarially trained models, while the natural accuracy is improved.

Model	Loss	N	A_1	A_2	A_3
En_1ResNet20	$\alpha = 0, \beta = 1$	69.60%	**47.81%**	**43.89%**	58.79%
En_1ResNet20	$\alpha = 1, \beta = 4$	**73.40%**	47.41%	41.86%	57.83%
En_1ResNet20	$\alpha = 1, \beta = 8$	71.35%	47.42%	42.46%	**59.01%**
En_2ResNet20	$\alpha = 0, \beta = 1$	71.58%	49.29%	**44.62%**	60.36%
En_2ResNet20	$\alpha = 1, \beta = 4$	**75.92%**	48.97%	43.41%	59.40%
En_2ResNet20	$\alpha = 1, \beta = 8$	74.72%	**49.66%**	43.96%	**60.65%**
En_5ResNet20	$\alpha = 0, \beta = 1$	75.40%	51.60%	**46.91%**	**61.52%**
En_5ResNet20	$\alpha = 1, \beta = 4$	**78.50%**	50.85%	45.02%	60.96%
En_5ResNet20	$\alpha = 1, \beta = 8$	77.35%	**51.62%**	45.63%	61.11%

Now, we consider several common loss functions: the hinge loss ($\phi(\theta) = \max\{1-\theta, 0\}$), the sigmoid loss ($\phi(\theta) = 1-\tanh\theta$), and the logistic loss ($\phi(\theta) = \log_2(1+e^{-\theta})$). Note that we want a loss function to be monotonically deceasing

in the interval $[-1, 1]$ as -1 indicates that the prediction is completely wrong, and 1 indicates the prediction is completely correct. Since our classes is 1 and -1, we will choose hyperbolic tangent as our activation function.

Table 5. Sparsity and structure of sparsity of quantized models. We use $\alpha = 1$ and $\beta = 8$ in trade-off loss (8). While models under both natural training and adversarial have large proportion of sparse weights, sparsity of adversarially trained models are much less structured. Trade-off loss function (6) can improve the structure.

Model	Quant	Loss	Weight Sparsity	Channel Sparsity	N	A_1	A_2
ResNet20	tnn	Natural	53.00%	11.16%	90.54%	12.71%	0.00%
En_1ResNet20	tnn	Natural	52.19%	9.57%	90.61%	26.21%	0.71%
ResNet20	tnn	AT	50.71%	2.55%	68.30%	44.80%	42.53%
En_1ResNet20	tnn	AT	50.31%	4.14%	71.30%	48.17%	43.27%
En_1ResNet20	tnn	(8)	55.66%	7.02%	73.05%	48.10%	42.65%
ResNet20	4-bit	Natural	42.79%	9.53%	91.75%	12.38%	0.00%
En_1ResNet20	4-bit	Natural	44.73%	10.52%	91.42%	27.99%	0.62%
ResNet20	4-bit	AT	43.93%	2.55%	71.49%	47.63%	44.08%
En_1ResNet20	4-bit	AT	48.35%	4.94%	73.05%	51.43%	45.10%
En_1ResNet20	4-bit	(8)	55.57%	7.42%	76.61%	51.92%	44.39%
ResNet56	tnn	Natural	60.96%	31.86%	91.91%	15.58%	0.00%
En_1ResNet56	tnn	Natural	60.66%	28.97%	91.46%	38.22%	0.36%
ResNet56	tnn	AT	54.21%	15.37%	74.56%	51.73%	46.62%
En_1ResNet56	tnn	AT	54.70%	16.74%	76.87%	53.16%	47.89%
En_1ResNet56	tnn	(8)	58.89%	21.36%	77.24%	52.96%	46.01%
ResNet56	4-bit	Natural	67.94%	39.16%	93.09%	16.02%	0.00%
En_1ResNet56	4-bit	Natural	71.07%	41.10%	92.39%	39.79%	0.33%
ResNet56	4-bit	AT	55.29%	17.10%	77.67%	52.43%	48.22%
En_1ResNet56	4-bit	AT	55.09%	18.11%	78.25%	55.48%	49.03%
En_1ResNet56	4-bit	(8)	67.31%	33.18%	79.44%	55.41%	47.80%

Proposition 1. *Let ϕ be any loss function that is monotonically decreasing on $[-1, 1]$ (all loss functions mentioned above satisfy this), and $\sigma(\theta) = \tanh\theta$. Define $B = \{\mathbf{x}|f(\mathbf{x})y \geq 0, f(\mathbf{x}')y \geq 0\}$ as in proposition 1. Then:*

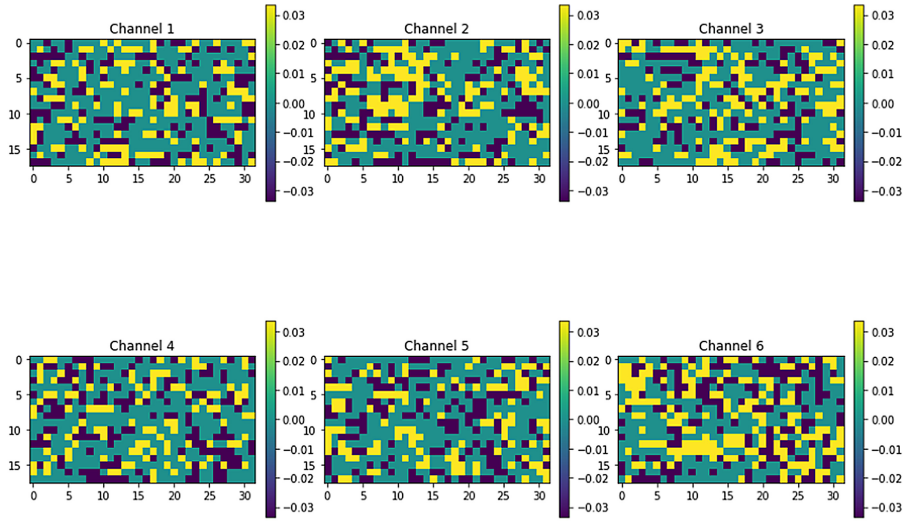
$$\mathcal{R}_\phi(f) \geq \mathcal{R}^*_\phi(f) \text{ on } B \text{ and } \mathcal{R}_\phi(f) \leq \mathcal{R}^*_\phi(f) \text{ on } B^C$$

Proof: We first define a set E:

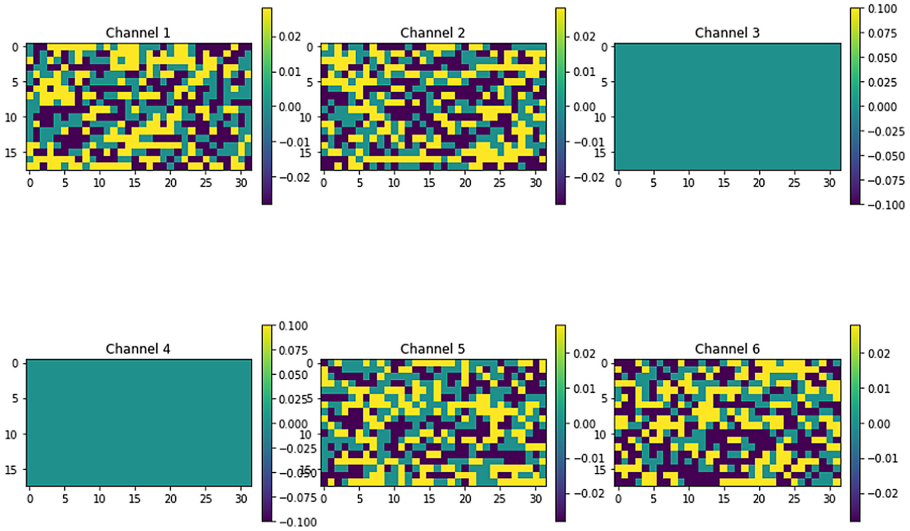
$$E = \{\mathbf{x}|f(\mathbf{x})y < 0, f(\mathbf{x}')y > 0\}$$

By the definition of adversarial examples in (4),

$$\mathbb{E}[\mathbf{1}\{E\}] = 0$$



(a) Ternary AT



(b) Ternary Trade-off

Fig. 1. Visualization of 4 ternary channels (reshaped for visualization) of a layer in $\text{En}_1\text{ResNet56}$ under natural training. There are less 0 weights in non-sparse channels. Nonzero weights of ternary channels under natural training are more concentrated. As a result, there are more sparse channels under natural training.

where $\mathbf{1}\{E\}$ *is the indicator function of the set. That is, the set that the classifier predicts original data point wrong but the perturbed data point correctly should have measure* 0. *Now, we define the following sets:*

$$D = \{\mathbf{x} | f(\mathbf{x})y \geq 0, f(\mathbf{x}')y < 0\}$$

$$F = \{\mathbf{x} | f(\mathbf{x})y < 0, f(\mathbf{x}')y < 0\}$$

We note that the activation function $\sigma(x)$ *preserves the sign of* x, *and* $|\sigma(x)| \leq 1$.

On the set B, $f(\mathbf{x})$, $f(\mathbf{x}')$, *and* y *have the same sign, so are* $\sigma(f(\mathbf{x}))$, $\sigma(f(\mathbf{x}'))$ *and* y. *Therefore*

$$\phi(\sigma(f(\mathbf{x}')) \cdot \sigma(f(\mathbf{x})) \geq \phi(\sigma(f(\mathbf{x}') \cdot y)$$

as $0 \leq \sigma(f(\mathbf{x}')) \cdot \sigma(f(\mathbf{x})) \leq \sigma(f(\mathbf{x}')) \cdot y$. *This shows*

$$\mathcal{R}_\phi(f) \geq \mathcal{R}^*_\phi(f) \text{ on } B$$

We note that $B^C = E \cup D \cup F$. *Since set* E *has measure zero, we only consider* D *and* F.

On D, *as* f *classifies* $\mathbf{x}$ *correct and* $\mathbf{x}'$ *wrong, we have*

$$\sigma(f(\mathbf{x}')) \cdot y \leq \sigma(f(\mathbf{x}')) \cdot \sigma(f(\mathbf{x})) \leq 0$$

$$\Rightarrow \phi(\sigma(f(\mathbf{x}')) \cdot \sigma(f(\mathbf{x})) \leq \phi(\sigma(f(\mathbf{x}') \cdot y)$$

On F, *as* f *classifies both* $\mathbf{x}$ *and* $\mathbf{x}'$ *wrong, we have*

$$\sigma(f(\mathbf{x}')) \cdot y \leq 0 \leq \sigma(f(\mathbf{x}')) \cdot \sigma(f(\mathbf{x}))$$

$$\Rightarrow \phi(\sigma(f(\mathbf{x}')) \cdot \sigma(f(\mathbf{x})) \leq \phi(\sigma(f(\mathbf{x}') \cdot y)$$

In summary, we have

$$\mathcal{R}_\phi(f) \leq \mathcal{R}^*_\phi(f) \text{ on } B^C$$

□

We partition our space into several sets based on a given classifier f, and we examine the actions of loss functions on those sets. We see that (7) penalize set B heavier than (8), but the classifier classifies both the natural data and the perturbed data correct on B. On the other hand, (7) does not penalize sets E and F, where the classifier makes mistakes, enough, especially on set F. Therefore, (8) as a loss function is more on target. Based on both experimental results and theoretical analysis, we believe (8) is a better choice to balance natural accuracy and robust accuracy.

As our experiments on the balance of accuracies with different parameters in our loss function (8) in Table 4. We find that it is possible to increase the natural accuracy while maintaining the robustness under relatively weak attacks (FGSM & CW), as the case of $\alpha = 1$ and $\beta = 8$ in Table 4. However, the resistance under relatively strong attack (IFGSM) will inevitably decrease when we trade-off.

5 Further Balance of Efficiency and Robustness: Structured Sparse Quantized Neural Network via Ternary/4-Bit Quantization and Trade-Off Loss Function

5.1 Sparse Neural Network Delivered by High-Precision Quantization

When we quantize DNNs with precision higher than binary, such as ternery and 4-bit quantization, zero is in quantization levels. In fact, we find that a large proportion of weights will be quantized to zero. This suggests that a ternary or 4-bit quantized model can be further simplified via channel pruning. However, such simplification requires structure sparsity of DNN architecture. In our study, we use the algorithm 1 as before with the projection replaced by ternary and 4-bit respectively. As shown in Table 5, we find that sparsity of quantized DNNs under regular training are significantly more structured than those under adversarial training. For both ternary and 4-bit quantization, quantized models with adversarial training have very unstructured sparsity, while models with natural training have much more structured sparsity. For example, 50.71% (0.135M out of 0.268M) of weights in convolutional layers are zero in a ternary quantized ResNet20, but there are only 2.55% (16 out of 627) channels are completely zero. If the sparsity is unstructured, it is less useful for model simplification as channel pruning cannot be applied. A fix to this problem is our trade-off loss function, as factor natural loss into adversarial training should improve the structure of sparsity. Our experiment (Table 5) shows that a small factor of natural loss, $\alpha = 1$ and β in (8), can push the sparsity to be more structured. Meanwhile, the deepness of models also has an impact on the structure of sparsity. The deeper the more structured the sparsity is. We see in Table 5 that, under the same settings, the structure of sparsity increases as the model becomes deeper. Figure 1 shows the difference between a unstructured sparsity of a ternary ResNet20 with adversarial training and a much more structured sparsity of ResNet56 with natural training. The trade-off function not only improves the natural accuracy of models with merely minor harm to robustness but also structures the sparsity of high-precision quantization, so further simplification of models can be done through channel pruning.

6 Benckmarking Adversarial Robustness of Quantized Model

Based on previous discussions, integrating the relaxation algorithm, ensemble ResNet, and our trade-off loss function, we can produce very efficient DNN models with high robustness. To our best knowledge, there is no previous work that systematically study the robustness of quantized models. As a result, we do not have any direct baseline to measure our results. Therefore, we benchmark our results by comparing to models with similar size and current state-of-the-art

Table 6. Generalization of quantized models to more datasets

Model	Size	Dataset	Loss	Quant	Ch sparsity	N	A_1	A_2	A_3
En_2ResNet20	0.54M	MNIST	(8)	BR	N/A	99.22%	98.90%	98.90%	99.12%
ResNet44	0.66M	MNIST	TRADES	Float	N/A	99.31%	98.98%	98.91%	99.14%
En_2ResNet20	0.54M	MNIST	(8)	Float	N/A	99.21%	99.02%	98.91%	99.14%
En_2ResNet20	0.54M	FMNIST	(8)	BR	N/A	91.69%	87.85%	87.22%	89.74%
ResNet44	0.66M	FMNIST	TRADES	Float	N/A	91.37%	88.13%	87.98%	90.12%
En_2ResNet20	0.54M	FMNIST	(8)	Float	N/A	92.74%	89.35%	88.68%	91.72%
En_1ResNet56	0.85M	Cifar10	(8)	4-bit	33.18%	79.44%	55.71%	47.81%	65.50%
ResNet56	0.85M	Cifar10	TRADES	Float	N/A	78.92%	55.27%	50.40%	59.48%
En_1ResNet56	0.85M	Cifar10	(8)	Float	N/A	81.63%	56.80%	50.17%	66.56%
En_1ResNet110	1.7M	Cifar100	(8)	4-bit	23.93%	53.08%	30.76%	25.73%	42.54%
ResNet110	1.7M	Cifar100	TRADES	Float	N/A	51.65%	28.23%	25.77%	40.79%
En_1ResNet110	1.7M	Cifar100	(8)	Float	N/A	56.63%	32.24%	26.72%	43.99%
En_2ResNet56	1.7M	SVHN	(8)	4-bit	49.03%	91.21%	70.91%	57.99%	72.44%
ResNet110	1.7M	SVHN	TRADES	Float	N/A	88.33%	64.80%	56.81%	69.79%
En_2ResNet56	1.7M	SVHN	(8)	Float	N/A	93.33%	78.08%	59.11%	75.79%

defense methods. We verify that the performance of the quantized model with our approach on popular datasets, including Cifar 10, Cifar 100 [10], MNIST, Fashion MNIST (FMNIST) [23], and SVHN [15]. In our experiments, we learn SVHN dataset without utilizing its extra training data. Our results are displayed in Table 6. In this table, size refers to the number of parameters. We find that quantization has very little impact on learning small datasets MNIST and FMNIST. En_2ResNet20 and ResNet40 have about the same performance while the previous is binarized. In fact, a binary En_2ResNet20 has about the same performance as its float equivalent, which means we get efficiency for 'free' on these small datasets. We benchmark the robustness of quantized models on large datasets, SVHN, Cifar10, and Ciar100, using models with 4-bit quantization. As in Table 6, 4-bit EnResNets with loss function (8) have better performance than float TRADES-trained ResNets with the same sizes. However, quantized models on these larger datasets are outperformed by their float equivalents. In another word, efficiency of models are not 'free' when learning large datasets, we have to trade-off between performance and efficiency. Although 4-bit quantization requires higher precision and more memories, the highly structured sparsity can compensate the efficiency of models. The 4-bit quantized models of En_1ResNet56 for Cifar10, En_1ResNet110 for Cifar 100, and En_2ResNet56 for SVNH in Table 6 have sizes of 0.58M, 1.43M, and 0.80M respectively if the sparse channels are pruned. Our codes as well as our trained quantized models listed in Table 6 are available at https://github.com/lzj994/Binary-Quantization.

7 Conclusion

In this paper, we study the robustness of quantized models. The experimental results suggest that it is totally possible to achieve both efficiency and robustness as quantized models can also do a good job at resisting adversarial attack. Moreover, we discover that adversarial training will make the sparsity from high-precision quantization unstructured, and a trade-off function can improve the sparsity structure. With our integrated approach to balance efficiency and robustness, we find that keeping a model both efficient and robust is promising and worth paying attention to. We hope our study can serve as a benchmark for future studies on this interesting topic.

References

1. 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)
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE (2017)
3. Chen, T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems, pp. 6571–6583 (2018)
4. Courbariaux, M., Bengio, Y., David, J.-P.: BinaryConnect: training deep neural networks with binary weights during propagations. In: Advances in Neural Information Processing Systems, pp. 3123–3131 (2015)
5. Dong, Y., et al.: Benchmarking adversarial robustness on image classification. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
6. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
7. Guo, C., Rana, M., Cisse, M., Van Der Maaten, L.: Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117 (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
9. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1389–1397 (2017)
10. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009)
11. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 (2016)
12. Li, H., De, S., Xu, Z., Studer, C., Samet, H., Goldstein, T.: Training quantized nets: a deeper understanding. In: Advances in Neural Information Processing Systems, pp. 5811–5821 (2017)
13. Li, Z., Shi, Z.: Deep residual learning and PDEs on manifold. arXiv preprint arXiv:1708.05115 (2017)
14. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
15. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)

16. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-net: imagenet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 525–542. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_32
17. Sinha, A., Namkoong, H., Duchi, J.: Certifying some distributional robustness with principled adversarial training. arXiv preprint arXiv:1710.10571 (2017)
18. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
19. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. arXiv preprint arXiv:1805.12152 (2018)
20. Wang, B., Luo, X., Li, Z., Zhu, W., Shi, Z., Osher, S.: Deep neural nets with interpolating function as output activation. In: Advances in Neural Information Processing Systems, pp. 743–753 (2018)
21. Wang, B., Shi, Z., Osher, S.: ResNets ensemble via the Feynman-Kac formalism to improve natural and robust accuracies. In: Advances in Neural Information Processing Systems, pp. 1655–1665 (2019)
22. Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., Gu, Q.: On the convergence and robustness of adversarial training. In: International Conference on Machine Learning, pp. 6586–6595 (2019)
23. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
24. Yin, P., Zhang, S., Lyu, J., Osher, S., Qi, Y., Xin, J.: BinaryRelax: a relaxation approach for training deep neural networks with quantized weights. SIAM J. Imaging Sci. **11**(4), 2205–2223 (2018)
25. Zhang, H., Yu, Y., Jiao, J., Xing, E.P., El Ghaoui, L., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: International Conference on Machine Learning (2019)
26. Zhuang, Z., et al.: Discrimination-aware channel pruning for deep neural networks. In: Advances in Neural Information Processing Systems, pp. 875–886 (2018)