Adversarial Example Defense and Detection using Quantized Neural Networks with Inference Time Dropout

Yushi Guan
Electrical & Computer Engineering - University of Toronto
ECE1784 Trustworthy Machine Learning
yushi.guan@mail.utoronto.ca

Abstract

Neural Network has seen a wide range of applications in recent years[1][2][3]. However, it is known that neural networks are susceptible to adversarial examples: examples with perturbations imperceptible to human beings that are carefully crafted to cause the neural networks to misclassify. One of the attempts to explain such a phenomenon is the linear explanation: the activation functions in the neural networks operate mostly in their linear regions [4]. Binary and Quantized Neural Networks can be potentially effective defenses against adversarial inputs since they have quantized activations that do not have a linear region. It has also been demonstrated that neural networks with dropout at inference time can be an effective tool for detecting adversarial examples through uncertainty estimation [5]. In this work, the effectiveness of using Binary and Quantized Neural Networks as adversarial defense and detection methods is evaluated. It is demonstrated that Binary and Quantized Neural Networks have higher classification accuracy on adversarial inputs compared to the full-precision networks. Monte Carlo estimation through randomized dropout further improve the quantized model's performance significantly. However, they are less capable at distinguishing the adversarial examples from the clean examples compared to their full-precision counterparts due to their high uncertainty on clean examples.

1 Introduction

Neural Network has been used in a wide range of applications since AlexNet demonstrated superior performance in the ImageNet 2012 challenge [1]. It has since been used in image classification, object detection, natural language processing, autonomous vehicles, and other tasks [1][6][2][3]. However, it is known that neural networks are susceptible to adversarial examples [4][7][8]. Examples with perturbations imperceptible to human beings that are carefully crafted can cause the neural networks to misclassify. Even worse, the neural networks often misclassify with high confidence. One attempt to explain why the adversarial examples are easy to craft is the linear explanation [4]: the activation functions in the neural networks operate mostly in their linear regions. The Fast Gradient Sign Method (FGSM) for adversarial example generation is proposed based on this intuition. The FGSM performs a one-step perturbation of all pixels up to an ϵ bound in the direction that minimizes the probability of the true class. Since the proposal of FGSM, different types of adversarial example generation methods have been proposed. For example, the Jacobian-based Saliency Attack (JSMA) perturbs pixels in the original input one at a time [9]. Madry et al. proposed the Projected Gradient Descent (PGD) attack which finds the adversarial examples by performing FGSM iteratively from a random point around the clean example [8].

As a result of these rapidly improving adversarial attacks, the defense methods are also quickly evolving. These defenses fall into roughly three categories. The first one is adversarial training, where adversarial examples are being used as training inputs to the neural network. Adversarial training aims to let the neural network learn about the adversarial inputs and classify them correctly. The second method is to defend adversarial examples through randomization. Inspired by differential privacy, Luceyer et al. proposed a method to inject a randomization layer into the neural network, through Monte Carlo estimation the true label can be predicted [10]. Xie et al. proposed to defend adversarial examples by adding random padding and shift to the input [11]. The third type of defense is to reject making inference for adversarial examples by detecting it. Feinman et al. to use inference time dropout as a Bayesian Neural Network, and then use the uncertainty of the estimation to reject potential adversarial examples [5]. It is shown that adversarial examples have higher uncertainties compared to the clean examples.

Galloway et al. studied the effectiveness of using Binary Neural Networks (BNNs) as an adversarial defense, following the intuition of Goodfellow et al [12][4]. There are two types of BNNs tested. The first type is a BNN that has weights and activations restricted to $\{1, -1\}$. And the second type is a randomized BNN that has weights quantized to $\{1, -1\}$ stochastically during inference. It is found that these BNNs generally have lower accuracy compared to their full-precision counterparts on clean examples, and higher accuracy on adversarial examples. After adversarial training, BNNs again perform worse than the full-precision networks. Since the introduction of randomized BNNs by [13], there hasn't been work that tries to evaluate the performance of a randomized BNN by performing Monte Carlo estimation, for which I will demonstrate in this paper.

To the best of the author's knowledge, this paper is the first to bridge model quantization and the effectiveness of inference time dropout as adversarial examples defenses. This work has two main contributions. First, it is showed that higher-precision neural networks have significantly lower uncertainty about the clean examples compared to the quantized lower-precision models. With similar uncertainties on adversarial examples regardless of model precision, higher-precision networks are better at detecting adversarial inputs. Second, Monte Carlo estimation using randomized dropout improves the neural network's prediction accuracy on adversarial inputs, and this improvement is more significant for lower-precision models that are heavily quantized.

2 Related Work

2.1 Binary Neural Network

The type of BNNs that are used in this work is the one introduced by Courbariaux [13]. Courbariaux's BNN uses real value during the training process and has its weights binarized during inference. The BNN has several notable features including Deterministic or Stochastic Binarization as well as the use of Straight Through Estimator (STE).

Deterministic or Stochastic Binarization During the forward pass of BNN, the real-valued weights can be binarized in one of two ways. Using Deterministic Binarization, the real-valued variables are passed through a deterministic Sign function:

$$x^b = Sign(x) = \begin{cases} +1 & if\ x \geq 0 \\ -1 & otherwise \end{cases} \tag{1}$$

where x^b is the binarized variable, and x is the real-valued variable. The binarization can also be done stochastically, x^b has an increased chance of being $+1$ as the value of x increases:

$$x^b = \begin{cases} +1 & with\ probability\ p = \sigma(x) \\ -1 & with\ probability\ 1 - p \end{cases} \tag{2}$$

where σ is the Hard Sigmoid function

$$\sigma(x) = clip\left(\frac{x+1}{2}, 0, 1\right) = \max\left(0, \min\left(1, \frac{x+1}{2}\right)\right) \tag{3}$$

Straight Through Estimator (STE) Courbariaux's BNN takes advantage of backpropagation methods that were developed for full precision Neural Networks such as Stochastic Gradient Descent (SGD) and Adam. To do so, the real-valued weights are kept during training. The gradients flow directly through the Sign function or the stochastic binarization function. Consider the sign function quantization:

$$q = Sign(r). \tag{4}$$

The gradient of the cost function C with respect to the real-valued weight r is:

$$g_r = g_q 1_{|r|\leq 1} \tag{5}$$

The function $1_{|r|\leq 1}$ has value 1 if $r <= 1$ and 0 otherwise. The authors found that clipping the gradient when the real-valued weight is too large is essential for the performance of the network.

2.2 Quantized Neural Network

Hubara et al. extended Courbariaux's work on Binary Neural Networks and proposed the Quantized Neural Network. The following quantization scheme is being used:

$$LinearQuant(x, bitwidth) = Clip\left(round\left(\frac{x}{bitwidth}\right) \times bitwidth, minV, \max V\right) \tag{6}$$

STE is also being used in the training process.

2.3 Attacks against Neural Networks using Adversarial Examples

In this section, overview for white-box attack methods that are being used for evaluation in this work is given. In a white-box attack, the adversary has access to the trained model of its target victim. In a black-box attack, the adversary does not have such knowledge.

Fast Gradient Sign Method (FGSM) FGSM is a simple attack proposed by Goodfellow et al. based on the linear assumption of neural networks [4]. The adversarial example is found using a linear approximation of the neural network. A perturbation based on Jacobian of the cost function with respect to the clean image is used to construct the adversarial example. The maximum perturbation allowed is ϵ.

$$x^{adv} = x + \varepsilon sign\left(\nabla_x J\left(x, y_{true}\right)\right) \tag{7}$$

Jacobian-based Saliency Map Attack (JSMA) JSMA is an iterative method that uses the forward gradient of the neural network [9]. The attack first construct an adversarial saliency map:

$$S(X,t)[i] = \begin{cases} 0, if \frac{\partial F_t(X)}{\partial X_i} < 0 or \sum_{j\neq t} \frac{\partial F_j(X)}{\partial X_i} > 0 \\ \left(\frac{\partial F_t(X)}{\partial X_i}\right)\left|\sum_{j\neq t} \frac{\partial F_j(X)}{\partial X_i}\right|, otherwise \end{cases} \tag{8}$$

From the saliency map, the pixel that will cause the maximum perturbation for the desired output adversarial class and the true class will be perturbed one at a time.

Projected Gradient Descent (PGD) PGD is an iterative attack method [8]. The vanilla iterative attack method "Basic Iterative Method (BIM)" extends the FGSM attack by performing it iteratively:

$$\mathbf{X}_0^{adv} = \mathbf{X}, \mathbf{X}_{n+1}^{adv} = clip_{\mathbf{X}}^{\epsilon}\left\{\mathbf{X}_n^{adv} + \alpha sign\left(\nabla_{\mathbf{X}} L\left(\mathbf{X}_n^{adv}, y\right)\right)\right\} \tag{9}$$

The PGD attack further improves on BIM by starting the adversarial example search from a random point in the ϵ norm ball.

It significantly improved the performance of I-FGSM by starting the adversarial example search at a random point in the ϵ norm ball.

2.4 Adversarial Example Defense and Detection

2.4.1 Adversarial Training Defense

One of the most popular choices of defense is training with adversarial examples [4][14][8]. Briefly, towards the later stage of training, adversarial examples are being generated using the white-box attack methods, which are then used as new training examples. The expectation is that by forcing the neural networks to properly handle the adversarial examples, it can become more robust to them.

However, it is found that this kind of defense often leads to gradient masking: The neural network defends itself against white-box attacks by rendering gradient around clean image points useless [15][16]. Instead of substantially move the decision boundaries to accommodate for the adversarial examples, the model simply does not produce adversarial examples that it has a hard time classifying.

The phenomenon of gradient masking is often seen as a result of performing adversarial example generation and training using the same model. Tramer et al. proposed to decouple this process by first generating adversarial examples using an ensemble of adversarial examples using multiple static neural networks, and then subsequently train the neural network using these adversarial examples [16]. Such defense is found to be more effective against black-box attacks.

2.4.2 Randomization Defense

Randomization has been a popular defense method considered by many. For example, Xie et al. introduced a defense by simply adding a random padding layer and a random resizing layer to the input of the neural network [11].

Lecuyer et al. proposed a certified robustness method against adversarial examples inspired by differential privacy (DP) [10]. Using a randomized function A, the $(\epsilon, \delta) - DP$ can be guaranteed as the expectation of the output is stable:

$$\mathbb{E}(A(x)) \leq e^{\epsilon}\mathbb{E}(A(x+\alpha)) + b\delta \tag{10}$$

This randomization is enabled by adding a noise layer to the neural network. The authors proved that the $(\epsilon, \delta) - DP$ robustness propagates through typical neural network operations. Thus, an autoencoder with a noise layer can be separately trained and attached to deep neural networks, avoiding retraining of the full model. The network's robustness is guaranteed by Monte Carlo estimation.

2.4.3 Detection of Adversarial Examples through Uncertainty Estimation

Neural Networks typically do not capture model uncertainty. They can often give overly confident predictions on unseen data [17]. In addition, adversarial examples can be easily crafted such that the neural network will predict incorrectly with high confidence [14][7].

Bayesian Neural Network is a framework that captures the uncertainty of the prediction [17]. However, it is not being widely adapt due to its prohibitive computation cost compared to its non-Bayesian counterparts. Gal et al. proved that using dropout in neural network inference is equivalent to Bayesian inference in deep Gaussian processes [17]. Sampling multiple times through a neural network with dropout can give the uncertainty of the prediction. The use of dropout can also be seen as using an ensemble of models.

Feinman et al. showed that when using dropout at test time as Bayesian inference, the model's uncertainty on clean examples and adversarial examples are significantly different [5]. The model has much higher uncertainty on the adversarial examples. Therefore, the model can refuse to make predictions on inputs that it thinks is uncertain. They also proposed Kernel Density Estimator to measure if the input falls into the submanifold of the clean data. With these two methods combined, they can detect adversarial examples at a high success rate.

2.5 Binary Neural Network Attacks and Defenses

There have been very few works evaluating the robustness of BNNs against adversarial examples. Galloway et al. evaluated BNNs' robustness on several gradient-based white-box and black-box

attacks [12]. Khalil et al. argued that using attack methods designed for full-precision networks to attack BNNs in a white-box setting is ill-formed and proposed combinatorial attacks that are suitable for the nature of BNNs [18].

Galloway et al. performed white-box attacks by calculating the gradient using the real-valued form of the BNNs. The attacks include both single-step attacks and iterative-steps attacks. It is found that BNNs are slightly more robust than full-precision networks when no adversarial training is used as a defense [12]. When adversarial training is used as a defense, the BNNs perform worse than full-precision networks on adversarial examples due to their limited expressive power. If the stochastic binarization is used, the model becomes very robust against iterative-step white-box attacks as a result of non-consistent gradient information from step to step [13][12]. It is also found that BNNs performed no better than full-precision networks under black-box attacks.

2.6 Quantized Neural Network Attacks and Defenses

Rakin et al. and Lin et al. have studied the effectiveness of using variations of Quantized Neural Network as adversarial input defenses. They have used Dynamic Quantization and Defensive Quantization respectively.

2.6.1 Dynamic Quantization

Rakin et al. proposed Dynamic Quantization [19]. During adversarial training of the Quantized Neural Network, the quantization thresholds become tunable parameters in adversarial training. The authors claimed higher accuracy on adversarial examples compared to the fixed quantization models, but the improvement seems to be minimal. Lin et al. also noted that Rakin's methods are susceptible to gradient masking.

2.6.2 Defensive Quantization

Lin et al.'s Defensive Quantization combines model quantization and controlling the network layer's Lipschitz constant by weight regularization as inspired by Cisse et al [20][21]. They demonstrated that model quantization can potentially increase the noise level if the noise jumps to the next quantization level. Applying the regularization effectively prevents the noises from amplifying. They showed that controlling the Lipschitz constant of a quantized neural network is an effective method to improve its robustness on adversarial examples.

3 Approach

In the experiments, I evaluated the performance of full-precision and BNNs in adversarial examples classification accuracy, and the ability to classify adversarial and clean inputs through uncertainty estimation. Figure 1 summarizes the experiment flow.

The experiments are performed on four types of neural networks: full-precision neural network, static BNNs, randomized BNNs and bit-quantized BNNs. I used the LeNet structure for all the networks with the only difference being if the weights are quantized and how they are quantized. The BNNs are trained using Straight Through Estimator (STE). During training, the gradient will propagate through the quantization operations without being changed, allowing effective training of earlier layers in the neural network. The first two types of BNNs have the same mechanism as the ones defined by Courbariaux et al. [13].

The bit-quantized BNN is a static BNN with quantized weight values:

$$quantized(x) = clip\left(\frac{round\left(x * 2^{b-1}\right)}{2^{b-1}}, -0, 1\right) \tag{11}$$

where b is the number of bits required to represent all possible weight values in addition to 0. For example, when $b = 2$, possible quantized weights are $\{-1.0, -0.5, 0.0, 0.5, 1.0\}$.

The dataset used is the MNIST dataset with 60000 training examples and 500 testing examples. All networks are trained over 15 epochs using Adam optimizer. The dropout rate for both training and

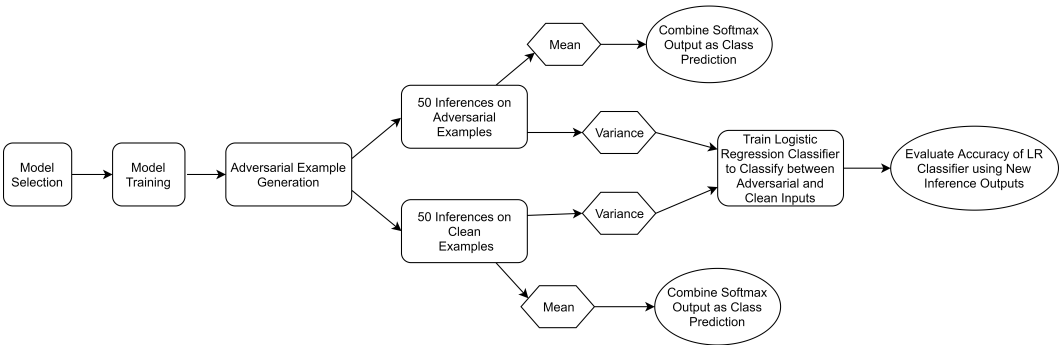


Figure 1: Summary of the Experiment FLow

inference is 0.5 unless otherwise stated. The trained network does not go through adversarial training. I refer readers interested in adversarial training's effectiveness on BNNs to [12].

The attack methods used are FGSM, JSMA and PGD. The implementations in Cleverhans open-source package are used [22]. For FGSM, the input variation parameter ϵ is set to 0.3. In JSMA, the maximum distortion gamma is set to 0.1, the change made to each feature theta is set to 1. In PGD, the maximum total distortion ϵ is set to 0.3, distortion per iteration is 0.01, the number of iterations is 40.

The trained network is then used to perform adversarial example classification and uncertainty estimation. There are 500 clean test inputs and 500 adversarial test inputs. 50 inferences are done for each input. The final output of the neural network (normalized class score after the Softmax layer) is used for both the classification and uncertainty estimation.

In classification, the output from 50 inferences are summed together, the class with the maximum summed value is used as the predicted result.

In uncertainty estimation, the variance of the final outputs are calculated:

$$\frac{\sum_{i=1}^{n} Var(O_i)}{n} \tag{12}$$

where n is the number of classes, and O_i all 50 inferences' output for the ith class.

The variances and adversarial/clean labels are then used to train a logistic regression classifier. New adversarial examples are generated and more test examples are used to evaluate the accuracy of the logistic regression classifier.

4 Experiment

4.1 Classification Accuracy

Table 1 and 2 summarize the classification accuracy on clean examples and adversarial examples generated using different methods. The -S postfix stands for single inference, where only 1 inference is done; the -M postfix stands for multiple inferences, where the outputs from 50 inferences are combined to make the final prediction.

Three observations can be drawn from the result. First, higher precision networks have better accuracy on clean inputs in most cases. This is potentially due to their higher expressivity, thus they perform better on the original task they are trained on. Second, for the single-step attacks, Quantized Neural Networks with lower bitwidth generally performed better on the adversarial inputs. Third, Monte Carlo estimation benefited binarized models the most in adversarial inputs prediction, despite already higher prediction accuracy compared to the higher-precision models. We will demonstrate in the next section that lower precision models have slightly higher output variance given the same input example. This high variance means lower-precision networks can benefit more from combining the predictions.

Randomized BNN with a higher dropout rate was tested but its accuracy on clean inputs is very low, making it unsuitable for any practical use so the results are not presented here. It can be seen

Table 1: Classification Accuracy on Clean and FGSM Inputs

Topology	Dropout	Clean-S	Clean-M	FGSM-S	FGSM-M
Rand BNN	0.05	62.4%	83.2%	45.6%	58.8%
Rand BNN	0	82.8%	93.6%	56.4%	76.4%
BNN	0.5	92.0%	97.6%	46.20%	56.2%
2bit Quantized	0.5	96.6%	99.2%	64.2%	70.2%
3bit Quantized	0.5	96.8%	99.2%	52.0%	55.0%
4bit Quantized	0.5	97.6%	99.0%	47.4%	51.6%
Full Precision	0.5	98.2%	99.4%	45.8%	45.0%

Table 2: Classification Accuracy on JSMA and PGD Inputs

Topology	Dropout	JSMA-S	JSMA-M	PGD-S	PGD-M
Rand BNN	0.05	12.8%	11.4%	9.8%	5.2%
Rand BNN	0	47.0%	68.0%	7.4%	6.6%
BNN	0.5	38.2%	51.4%	11.60%	9.00%
2bit Quantized	0.5	32.8%	35.8%	1.6%	1.6%
3bit Quantized	0.5	40.4%	44.2%	0.8%	0.8%
4bit Quantized	0.5	23.8%	21.4%	3.6%	3.4%
Full Precision	0.5	32.8%	35.2%	0.60%	0.60%

that dropout severely hinders the network's ability to classify even with such a small dropout rate. It seems that the combination of random weight and random dropout really hinders the training process of randomized BNNs. The exact reason why static BNNs can handle 50% dropout rate while randomized BNNs are having difficulties handling 5% dropout rate can be an interesting future research direction. Lastly, we note that none of these methods seem to be able to handle PGD inputs.

4.2 Uncertainty Estimation

Table 3 and 4 summarize the uncertainty of the output for clean examples and adversarial examples. Logistic regression (LR) classifiers have been trained; they classify the inputs as either adversarial or clean. The LR classification accuracies are also presented in the tables.

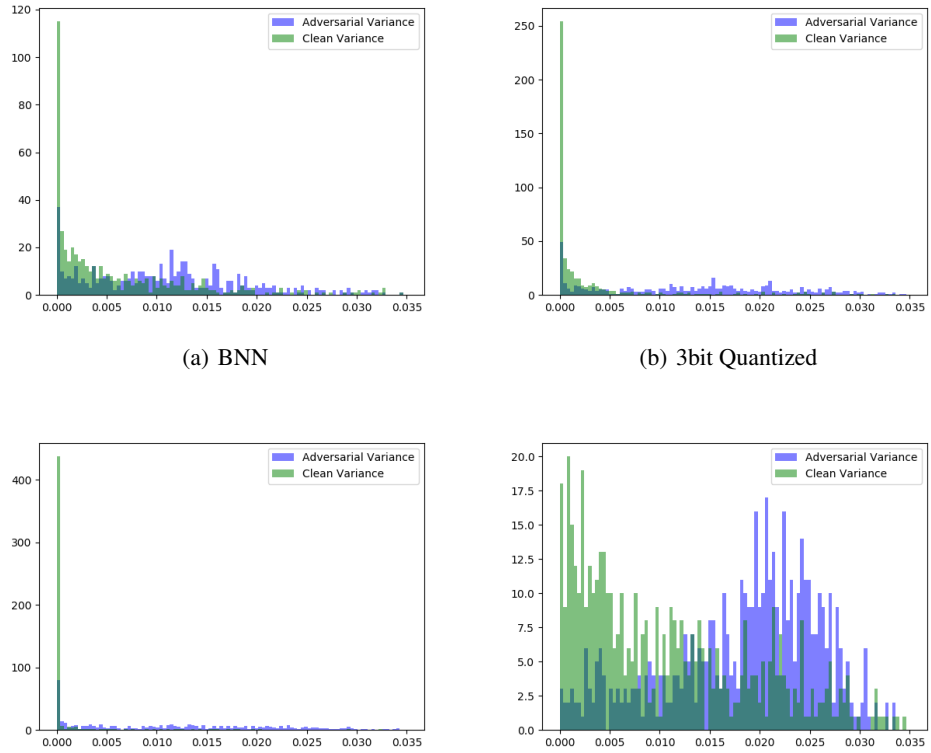
As shown in table 3 and 4, compared to lower precision networks, the higher precision networks have similar uncertainty on single-step attack inputs compared to the lower precision networks, while having considerably lower uncertainty on clean inputs. This is also demonstrated in Figure 2. Due to the larger uncertainty difference, the higher precision networks generally have better performance when classifying between adversarial and clean inputs. In particular, the full-precision network can reach 82.4% and 93.9% accuracy when classifying FGSM and JSMA adversarial inputs from the clean inputs, making it possible to be deployed for practical use for classifying single-step attack inputs. It is also showed that stronger multi-step attacks generate inputs that cause the neural networks to classify incorrectly with low uncertainty. All networks have very low uncertainty on the PGD inputs, making it impractical for them to distinguish PGD adversarial inputs from the clean examples.

Table 3: Classification Uncertainty on Clean and FGSM Inputs

Topology	Dropout	Clean μ,σ	FGSM μ,σ	FGSM LR
Rand BNN	0.05	1.11e-02, 1.23e-05	1.10e-02, 8.96e-06	49.7%
Rand BNN	0	1.08e-02, 7.91e-05	1.84e-02, 5.75e-05	71.1%
BNN	0.5	5.97e-03, 5.25e-05	1.12e-02, 6.80e-05	66.8%
2bit Quantized	0.5	2.84e-03, 3.68e-05	1.31e-02, 9.32e-05	77.6%
3bit Quantized	0.5	3.08e-03, 4.15e-05	1.39e-02, 9.36e-05	78.2%
4bit Quantized	0.5	2.46e-03, 3.88e-05	1.51e-02, 1.07e-04	80.5%
Full Precision	0.5	9.62e-04, 1.42e-05	1.07e-02, 7.86e-05	82.4%

Table 4: Classification Uncertainty on JSMA and PGD Inputs

Topology	Dropout	JSMA μ,σ	JSMA LR	PGD μ,σ	PGD LR
Rand BNN	0.05	6.51e-04, 3.55e-07	63.3%	2.49e-03, 1.20e-06	52.4%
Rand BNN	0	1.65e-02, 1.46e-05	76.0%	1.06e-02, 7.91e-06	54.2%
BNN	0.5	1.80e-02, 5.06e-05	74.6%	1.06e-02, 4.98e-05	68.3%
2bit Quantized	0.5	1.82e-02, 8.50e-05	86.6%	1.73e-04, 1.51e-06	52.9%
3bit Quantized	0.5	1.97e-02, 1.24e-04	85.1%	2.85e-04, 3.50e-06	51.5%
4bit Quantized	0.5	1.97e-02, 1.83e-04	80.7%	4.80e-03, 9.99e-05	55.9%
Full Precision	0.5	3.17e-02, 1.19e-04	93.9%	6.68e-04, 1.10e-05	51.4%



(a) BNN (b) 3bit Quantized

(c) Full Precision (d) Randomized BNN

Figure 2: Prediction Uncertainty on FGSM Inputs vs. Clean Inputs. Each histogram contains 100 bins between 0 and 0.035. Compare (a), (b) and (c), higher-precision networks have lower uncertainty on clean examples. Their uncertainties on adversarial examples have smaller differences. Randomized BNNs have significantly higher uncertainty on both adversarial and clean inputs.

5 Future Research

Improved adversarial input detection classifier : the logistic regression classifier used currently is a simple method. The adversarial input classifier can be improved by modeling the inference time uncertainties as probabilistic distributions. For example, a probability distribution mixture with an exponential component (that models the clean input uncertainties well) and a Gaussian component (models the adversarial input uncertainties) can be used. The classifier's performance can be significantly improved.

Experiment with more complex dataset : MNIST may be ideal for Binary Neural Networks since the image is black and white and there are potentially less interactions between the different color channels like in CIFAR10 or ImageNet. It would be interesting to evaluate these topologies' performance on these datasets and see if the results are different.

6 Conclusion

This work bridged the gap between model quantization and inference time dropout in the context of adversarial example defenses. Models with different levels of quantization have their unique advantages and disadvantages. It is demonstrated that Binary and Quantized Neural Networks are more robust against single-step adversarial attacks compared to the full-precision neural networks. Monte Carlo estimation with randomized dropout further increases their robustness significantly. However, they are not good candidates for adversarial input detection through uncertainty estimation. It is shown that higher precision networks, especially the full-precision network, have very low uncertainty about the clean test inputs. The uncertainty of networks with different precision on the adversarial examples are very close. As a result, the uncertainty gap between the clean and the adversarial is larger for higher precision networks. Making the higher precision networks more suitable for classifying between adversarial and clean inputs.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016.

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[5] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts, 2017.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2016.

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[9] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings, 2015.

[10] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy, 2018.

[11] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization, 2017.

[12] Angus Galloway, Graham W. Taylor, and Medhat Moussa. Attacking binarized neural networks, 2017.

[13] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016.

[14] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2016.

[15] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2016.

[16] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses, 2017.

[17] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2015.

[18] Elias B. Khalil, Amrita Gupta, and Bistra Dilkina. Combinatorial attacks on binarized neural networks, 2018.

[19] Adnan Siraj Rakin, Jinfeng Yi, Boqing Gong, and Deliang Fan. Defend deep neural networks against adversarial examples via fixed anddynamic quantized activation functions, 2018.

[20] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness, 2019.

[21] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples, 2017.

[22] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.