Relative Robustness of Quantized Neural Networks Against Adversarial Attacks

Kirsty Duncan*, Ekaterina Komendantskaya†, Robert Stewart‡ and Michael Lones§
Department of Computer Science, Heriot-Watt University
Edinburgh, UK
Email: *krd1@hw.ac.uk, †ek19@hw.ac.uk, ‡R.Stewart@hw.ac.uk, §m.lones@hw.ac.uk

***Abstract*—Neural networks are increasingly being moved to edge computing devices and smart sensors, to reduce latency and save bandwidth. Neural network compression such as quantization is necessary to fit trained neural networks into these resource constrained devices. At the same time, their use in safety-critical applications raises the need to verify properties of neural networks. Adversarial perturbations have potential to be used as an attack mechanism on neural networks, leading to "obviously wrong" misclassification. SMT solvers have been proposed to formally prove robustness guarantees against such adversarial perturbations. We investigate how well these robustness guarantees are preserved when the precision of a neural network is quantized. We also evaluate how effectively adversarial attacks transfer to quantized neural networks. Our results show that quantized neural networks are generally robust relative to their full precision counterpart (98.6%–99.7%), and the transfer of adversarial attacks decreases to as low as 52.05% when the subtlety of perturbation increases. These results show that quantization introduces resilience against transfer of adversarial attacks whilst causing negligible loss of robustness.**

***Index Terms*—neural network, verification, adversarial attack**

I. Introduction

Deep neural networks are widely used in everyday applications, from healthcare [24] and finance [9] to self driving vehicles [23] and aircraft collision avoidance systems [17]. They are capable of classifying data into a finite set of classes to a high degree of accuracy, by training on a set of inputs and corresponding classes. However, even high accuracy on a given data set does not guarantee the network's robustness against adversarial attacks [35]. Adversarial attacks are given by small modifications to the input of a neural network which can cause the network to misclassify. Figure 1 gives an example of a benign and an adversarial input image for a given neural network. As use of deep learning is growing in safety critical scenarios, it becomes crucial to be able to verify that a network will behave as expected.

Many techniques exist to generate adversarial attacks. They follow several ideas: efficiently search the input space for adversarial examples using optimization techniques [26], exploit sensitive features of a specific input and modify these to search for nearby adversarial examples via the Fast Gradient Sign Method (FGSM) [13], attacks through geometric transformations [10], and generative adversarial networks, where two models are pitted against each other in order to learn the input data generation distribution and generate adversarial examples around inputs or from noise [12].

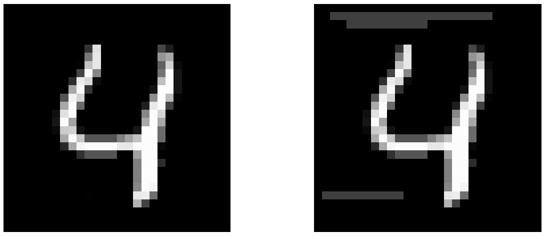


Fig. 1. Example of an image from MNIST dataset [8] correctly classified as a 4 by a neural network and the same image, slightly modified, which is misclassified by the same network as a 6.

Defenses aim to strengthen a network's resilience to adversarial attacks and include adding adversarial examples to the training dataset [35], transforming inputs, e.g. reducing input dimensionality [14]. Some approaches (e.g. [15], [17], [18], [32], [39]) formally verify a network's robustness to adversarial attacks, which involves giving proven guarantees of correct classification modulo constrained inputs.

In this paper, we will be working with one such tool, the *Deep Learning Verifier* (DLV) [15]. Unlike all other papers on neural network verification, we are interested in a different verification scenario, where a trained and verified neural network is quantized in order to be efficiently deployed.

Our research questions are: *do the robustness guarantees transfer from full precision neural networks to their quantization form? To what degree? And can quantization be favourable for robustness?* These research questions are motivated by (a) increasing demand for deploying deep learning on special purpose hardware, and (b) increasing pressure to guarantee that such applications are secure.

a) Deep Learning on Special Purpose Hardware: There is a rising number of implementations of deep learning on special purpose hardware AI accelerators (e.g. using FPGAs [37]), mobile phones [20], and special-purpose robotics hardware. Due to hardware resource constraints, these processor architectures are often incapable of accommodating state-

Supported by grant *SecCon-NN: Neural Networks with Security Contracts towards lightweight, modular security for neural networks* funded by the National Cyber Security Center UK, and EPSRC grant *Border Patrol: Improving Smart Device Security through Type-Aware Systems Design* (EP/N028201/1).

of-the-art, *"full precision"* deep neural networks, and hence require compression techniques to produce faster and less memory-intensive models, e.g. to speedup convolutional neural networks on embedded GPUs [29]. There are two categories of approximation algorithms for neural networks: quantization and weight reduction. Quantization [16] reduces the precision of weights and/or activations, reducing the memory size and bandwidth requirements, which can increase the throughput speed of the network. In some cases, precision is reduced to binary values [7], with very little reduction in accuracy compared to a full-precision model. Weight reducing algorithms remove redundant parameters, reducing the number of weights and activations in the network, and as with quantization, can result in negligible accuracy loss [38]. Quantization can be built into the training process e.g. using the Xilinx FINN framework [37] for FPGAs, or performed after training a network e.g. using Tensorflow LITE [1].

Little work has examined the effect that compressing a neural network has on its robustness. Verification methods for binarized neural networks have been proposed (e.g. [27].) Due to network parameters having binary values, scalability of techniques can be substantially increased [5]. Lin et al [21] generate adversarial examples via FGSM [13]. They observe that quantized networks misclassify examples generated within a smaller radius from the original image. This differs from the verification approach but raises the question of *whether quantized networks are less robust than full-precision networks.* The first aim of this paper is to systematically study this question.

b) Security Implications: The majority of adversarial attacks on neural networks happen in a black-box manner, i.e. without access to the network's architecture or parameters. This is possible because adversarial inputs tend to *transfer* across networks trained for the same task. Two neural networks trained with different hyperparameters (i.e. different numbers and sizes of hidden layers and different activation functions) or trained on a different subset of the training examples are sensitive to the same adversarial examples [35]. Universal adversarial attacks, i.e. small, image-agnostic perturbations which result in misclassification when applied to different images, generalize across different neural networks [25]. It has been shown that adversarial inputs span a contiguous subspace, where the subspaces generated for two models contain a significant overlap [36].

Quantization has been used successfully as a defense mechanism against adversarial attacks. This can be achieved by compressing inputs to filter adversarial noise [28] or by compressing network parameters during training resulting in a more robust network [19], [30]. Poor transfer of adversarial examples across networks with different levels of low-bitwidth quantization has been observed [4].

The networks studied in these works are retrained during quantization. In contrast, we are interested in the impact on verifiable robustness when parameters of full-precision networks are quantized but *not* retrained. Little is known about how well *adversarial attacks transfer from full precision to this kind of quantized neural networks*. This is relevant for black-box attacks when attackers do not know whether the neural network they attack is quantized. Providing the first insights into this problem is the second main aim of this paper.

In order to study these research questions, we introduce the notion of *relative robustness*. Intuitively, a quantized neural network is robust relative to its full precision version if, whenever the full precision network is proven robust against adversarial attacks, its quantized version remains robust against the same attacks. In this paper, we use the DLV framework [15] to verify robustness of full precision networks (Section III). We say that *a quantized neural network is robust relative to a full precision network* for a given input, if the quantized neural network correctly classifies the perturbed inputs generated by the DLV when verifying the full-precision network.

The key results of this paper are:

- Quantised networks are generally robust relative to their full precision counterpart, e.g. for 98.63-99.69% of inputs from the MNIST and CIFAR-10 datasets (Section IV-B).
- Notably, transfer of adversarial attacks to quantized networks is surprisingly low. It is also dependent on the nature of the perturbation, e.g. as low as 52.05% transfer for subtle adversarial perturbations and as high as 89.61% for sharp perturbations on CIFAR-10 (Section IV-C).

II. Background

A. Neural Networks and Quantized Neural Networks

Neural networks are machine learning systems capable of learning to classify previously unseen inputs, typically after training on a set of human-labeled inputs.

Definition II.1 (Neural Network)**.** Let $X \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$ and $Y \subset \mathbb{N}$. A *neural network* N is a function $N : X \to Y$ which transforms an *input* $x \in X$ to a *classification* $y_x \in Y$.

Let $N(x) = \Phi_i(\Phi_{i-1}(\dots\Phi_1(x)))$, for some functions $\Phi_1, \dots, \Phi_i$. We call $0, \dots, i$ *layers* of N, and $\Phi_1, \dots, \Phi_i$ the *activation functions* of N. The *activation* $a_{i,x}$ of a layer i for an input $x \in X$ is calculated by applying the activation function Φ_i to the activation $a_{i-1,x}$ of the previous layer $i-1$:

$$a_{i,x} = \Phi_i(\Phi_{i-1}(...\Phi_1(x)))$$

Thanks to the abstract use of activation functions, the above definition captures a broad range of architectures. In particular, we assume that an activation function may subsume operations with weight matrices. For example, to define a fully-connected input layer, it suffices to say that $\Phi_1 = \sum(x \times w_{0,1})$, where $w_{0,1}$ is the weight matrix connecting layers 0 and 1. Similarly for convolutional layers, their shapes are determined by the shapes and sizes of weight matrices associated with the activation functions of those layers. Generally, we use $w_{0,1}, \dots, w_{i-1,i}$ to refer to weight matrices associated with the activation functions of layers $1, \dots, i$, respectively. We use notation A and W to refer respectively to the sets of all activations and all weights for a neural network N.

We say a neural network is a *full-precision neural network* (FPNN) if its weights and activations are full precision 32-bit floating point values.

Definition II.2 (Quantized Neural Network)**.** Given an FPNN $N : X \rightarrow Y$ with weights W and activations A, an *n-bit quantized neural network* (QNN) N^Q is a function $N^Q : X \rightarrow Y$ with n-bit precision weights W^Q and activations A^Q which approximate W and A, respectively.

B. Adversarial Perturbations

Definition II.3 (Dataset)**.** A dataset $D \subseteq X$ is made up of inputs $x \in D$ each with a corresponding ground-truth label, $l_x \in \mathcal{L}_D$ where $\mathcal{L}_D \subset \mathbb{N}$. A neural network N *trained on* D has input space X and output space $Y = \mathcal{L}_D$.

Perturbed inputs known as adversarial perturbations are statistically likely to cause misclassifications across different networks trained with different architectures and even trained on different data sets. The formal definition is as follows:

Definition II.4 (Adversarial Perturbation)**.** Let N be a neural network trained on D, $x \in X$ be an input to N and $l_x \in \mathcal{L}_D$ be the ground-truth label for x. If the output classification $y_x = N(x)$ is $y_x = l_x$, the input x has been correctly labeled by N. An *adversarial perturbation* is a vector $\epsilon \in X$ which can be used to generate a a *perturbed input* $x' = x + \epsilon$.

If $y_{x'} \neq l_{x'}$, the adversarial perturbation is *effective*.

If $y_{x'} = l_{x'}$, the adversarial perturbation is *ineffective*.

C. Deep Learning Verifier (DLV) Tool

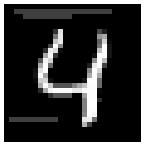
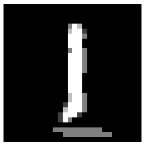
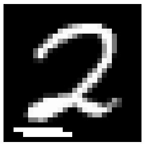
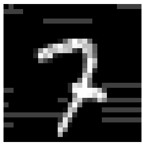
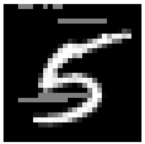
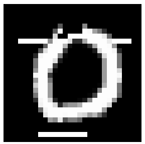
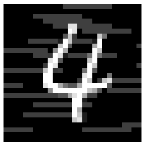
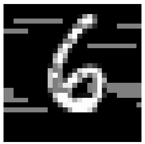
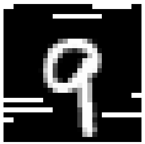
We define neural network robustness similarly to [15], [17], [32], [39]. A neural network is verified as robust with respect to an input image if it correctly classifies all images within a small region around the image.

This work uses the Deep Learning Verifier (DLV) tool [15] to verify that a neural network is robust. DLV generates a set of perturbed images within a certain region around input images in the colour space. This generated set effectively "covers" the region. We use these generated sets to test the robustness of quantized neural networks. The region is parameterized by a radius and set of manipulations.

1) Manipulations: The notion of manipulations was introduced in [15] in order to discretise the search space. Manipulations can be defined to represent camera scratches, weather or any application-specific modification to input images. In the context of this work, a manipulation is a change to a single pixel value, with manipulation step size $|\delta|$ equal to a value between 0, no change, and 1. In the greyscale dataset MNIST, +1 represents change from black to white and -1 represents change from white to black. For CIFAR-10, the change is made to one of 3 colour channels: $+1$ applied to the R channel will turn a pixel red. The choice of channel is random.

With the same search radius but a smaller manipulation step, more pixels may be modified, but the change to each pixel is smaller. Table I provides examples of perturbed inputs generated for different values of search radius and manipulation step size. In examples (d) and (g), many more pixels have been modified than in (f) and (i) respectively, but the change to each individual pixel is less sharp. Smaller manipulation steps create a more subtle difference to pixels, making the perturbed image less distinguishable from the original.

TABLE I
THE EFFECT OF MODIFYING SEARCH RADIUS AND MANIPULATION STEP SIZE ON ADVERSARIAL IMAGES GENERATED BY DLV.

	$\|\delta\| = \pm 0.25$	$\|\delta\| = \pm 0.5$	$\|\delta\| = \pm 1$
r=10	 (a) 4 misclassified as 6	 (b) 1 misclassified as 9	 (c) 2 misclassified as 7
r=20	 (d) 7 misclassified as 2	 (e) 5 misclassified as 6	 (f) 0 misclassified as 7
r=40	 (g) 7 misclassified as 2	 (h) 6 misclassified as 2	 (i) 9 misclassified as 2

2) Search Radius: The radius r in the input space X is defined as the Manhattan distance between two images. All possible combinations of manipulations δ are applied to the original image to exhaustively cover the region defined by radius r. DLV does this by building a complete and covering ladder, using an SMT solver. If the image corresponding to every node on the ladder is classified correctly by the network, it is certified as robust for that image, manipulation step and search radius. If an image on the ladder results in misclassification, it is an effective adversarial perturbation. Existence of such an image shows that the neural network is not robust (within the given parameters).

DLV is more likely to find effective adversarial perturbations with a greater search radius, however the perturbed image is more obviously altered from the original input image. As can be seen from Table I, (a), (d) and (g) increasing radius for a constant manipulation step size increases the number of pixels that have been changed. Increasing manipulation step size for a constant radius decreases the number of pixels that have been changed, but increases the intensity of the change.

Definition II.5 (Neural Network Robustness)**.** A network N is *robust* for input image $x \in D$ with network classification $y_x \in Y$, within radius r with respect to manipulations δ if generating an exhaustive list of perturbations ϵ where $|\epsilon| < r$

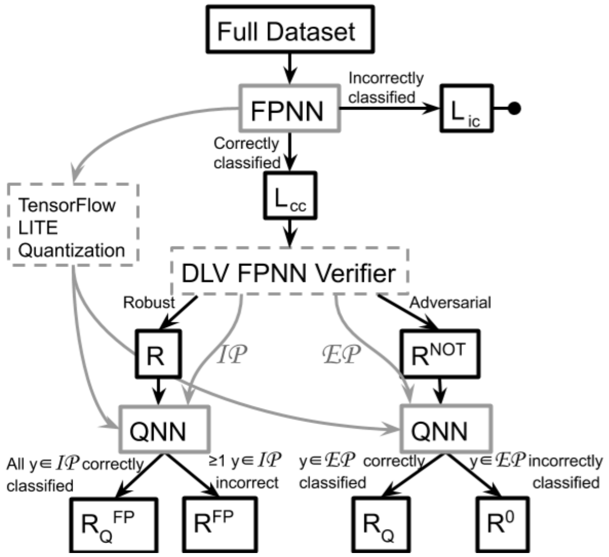

Full Dataset
FPNN
Incorrectly classified
L_{ic}
Correctly classified
L_{cc}
TensorFlow LITE Quantization
DLV FPNN Verifier
Robust
Adversarial
R
$\mathcal{IP}$
$\mathcal{EP}$
R^{NOT}
QNN
QNN
All $y \in \mathcal{IP}$ correctly classified
≥1 $y \in \mathcal{IP}$ incorrect
$y \in \mathcal{EP}$ correctly classified
$y \in \mathcal{EP}$ incorrectly classified
R_Q^{FP}
R^{FP}
R_Q
R^0

Fig. 2. Verifying Robustness of FPNNs and QNNs

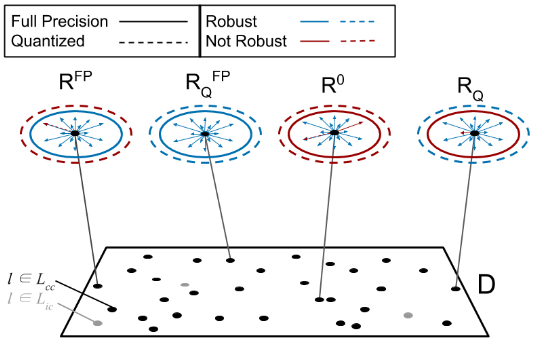


Fig. 3. The test dataset, D, is split into two sets: the correctly classified inputs, L_{cc}, and incorrectly classified inputs, L_{ic}. The inputs in L_{ic} are ignored. From L_{cc}, inputs are verified as robust for: the full precision and quantized network, R_Q^{FP}, the full precision but not the quantized network, R^{FP}, the quantized but not full precision network, R_Q, neither network, R^0.

from combinations of manipulations δ, does not generate a perturbation such that $y_{x+\epsilon} \neq y_x$.

III. Methodology

Figure 2 shows the steps to count the number of perturbed images that DLV uses to check the robustness of a FPNN and/or the QNN. The FPNN is trained using Keras [6]. First, the FPNN is used to identify correctly classified images. Misclassified images are discarded. Second, we have modified DLV to either produce effective adversarial perturbations (an image set $\mathcal{EP}$, described shortly) from a single image that prove the FPNN is not robust, or to otherwise produce ineffective perturbed images $\mathcal{IP}$ that were correctly classified, verifying the FPNN robust. Third, we use TensorFlow LITE to quantize the 32-bit FPNN to produce the 8-bit QNN. Lastly, the ineffective adversarial perturbations $\mathcal{IP}$ are tested against the QNN and if no misclassifications occur then we define the QNN as robust relative to the FPNN for the original image, otherwise the QNN is not relatively robust.

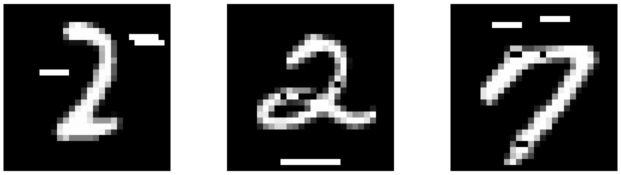


Fig. 4. Images in the R^{FP} set: adversarial pertubations of images from MNIST which were correctly classified by the FPNN but misclassified by the QNN. From left to right: 2 misclassified as an 8, 2 misclassified as a 9 and 7 misclassified as a 0. Manipulation step is ±1.

All accompanying code for this paper is available online[1].

Let N be a full precision network and N^Q the corresponding quantized network. Figure 3 refers to the following data sets: the full dataset D, the set of incorrectly classified images:

$$L_{ic} = \{x \in D | N(x) \neq l_x\},$$

the set of correctly classified images:

$$L_{cc} = \{x \in D | N(x) = l_x\}.$$

For every $x \in L_{cc}$ with classification y_x, a set of perturbed images is generated using DLV by applying all possible perturbations ϵ, made up of manipulations of step size δ within search radius r (Definition II.4).

For a given network N and an image $x \in L_{cc}$, the set of ineffective adversarial perturbations $\mathcal{IP}$, generated by DLV by exhaustively performing manipulations in the search radius is:

$$\mathcal{IP}(N, x) = \{\epsilon \in perturb(x) | \; |\epsilon| < r, N(x) = N(x + \epsilon)\}$$

The set of *effective adversarial perturbations* $\mathcal{EP}$ is:

$$\mathcal{EP}(N, x) = \{\epsilon \in perturb(x) | \; |\epsilon| < r, N(x) \neq N(x + \epsilon)\}$$

Each image $x \in L_{cc}$ is then categorised according to the robustness of the full precision network N, either as robust (R) or not robust (R^{NOT}):

$$R = \{x \in L_{cc} | \mathcal{EP}(N, x) = \varnothing\}$$

$$R^{NOT} = \{x \in L_{cc} | \mathcal{EP}(N, x) \neq \varnothing\}$$

Four image sets are then calculated from these. The set of images R_Q^{FP} are those for which DLV verified the full precision N as robust, and the quantized network N^Q correctly classifies all adversarial perturbations of the original image which were ineffective on N:

$$R_Q^{FP} = \{x \in R | \forall y \in \mathcal{IP}(N, x), N^Q(y) = l_x\}$$

The image set for which N is robust but N^Q misclassifies one or more adversarial perturbations of the original image which were ineffective on N is:

$$R^{FP} = \{x \in R | \exists y \in \mathcal{IP}(N, x), N^Q(y) \neq l_x\}$$

Examples of MNIST images in R^{FP} are in Figure 4.

[1] https://github.com/KirstyRD/RelativeRobust

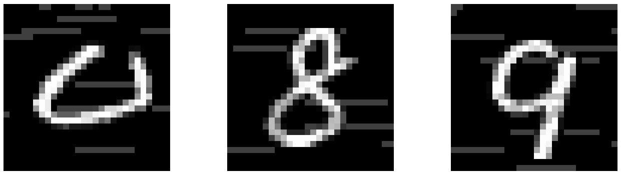


Fig. 5. Images in the R_Q set: perturbations of images from MNIST which were misclassified by the FPNN, but correctly classified by the QNN. From left to right: 0 misclassified as a 6, 8 misclassified as a 3 and 9 mislcassified as a 7. Manipulation step is ±0.25

TABLE II

RELATIVE ROBUSTNESS: PERCENTAGE OF IMAGES IN R WHICH ARE ALSO IN R_Q^{FP}, FOR MNIST AND CIFAR-10 DATASETS.

D	r	$\|\delta\|$	$\|R\|$	$\|R_Q^{FP}\|$	Relative Robustness
MNIST	10	± 1	2323	2312	99.53%
	20	± 1	648	646	99.69%
		$\pm\frac{1}{4}$	3282	3239	98.69%
CIFAR-10	20	± 1	1814	1797	99.06%
		$\pm\frac{3}{4}$	2039	2020	99.07%
		$\pm\frac{1}{2}$	2352	2325	98.85%
		$\pm\frac{1}{4}$	2775	2737	98.63%

The image set for which N is not robust but N^Q correctly classifies the effective adversarial perturbation y for N generated by DLV from the original image is:

$$R_Q = \{x \in R^{NOT} | \forall y \in \mathcal{EP}(N,x), N^Q(y) = l_x\}$$

Examples of MNIST images in R_Q are in Figure 5.

The image set for which N is not robust and N^Q also misclassifies the effective adversarial perturbation y is:

$$R^0 = \{x \in R^{NOT} | \forall y \in \mathcal{EP}(N,x), N^Q(y) \neq l_x\}$$

where $|R_Q^{FP}| + |R^{FP}| + |R_Q| + |R^0| = |L_{cc}|$.

Definition III.1 (Relative Robustness)**.** Given a data set D, input $x \in D$ with the class label l_x, an FPNN N that is robust for x and the set $\mathcal{IP}(N,x)$, N^Q is robust relative to N if $\forall x' \in \mathcal{IP}(N,x), N^Q(x') = l_x$.

IV. EVALUATION

This section uses the methodology from Section III to evaluate robustness in three ways. (1) Section IV-A evaluates the robustness of an FPNN against perturbed images with different labels, to analyse whether the susceptibility to adversarial attack may depend on targeting specific classes. (2) Section IV-B evaluates the relative robustness (Definition III.1) of 8-bit QNNs of different neural networks with two datasets, MNIST and CIFAR-10. (3) Section IV-C evaluates the transferability of adversarial attacks, found for FPNNs, to their QNN counterparts.

The network used to analyse the MNIST dataset has 2 hidden layers, each with 512 neurons using the ReLU activation function, and a dropout of 0.2. The network used for CIFAR-10 experiments has 2 hidden layers, each with 256 neurons also using the ReLU activation function, and dropout of 0.5.

A. Adversarial Robustness Across Labels

Figure 6 presents the percentage of inputs for each class which resulted in an effective adversarial attack (i.e. perturbed images that fool the network) by DLV for different parameter. The MNIST test set is made up of 10,000 inputs with between 892 and 1135 inputs for each of the 10 classes. Certain classes, e.g. 1, 4 and 9, are more frequently found to be susceptible to attack, especially for smaller DLV search spaces with δ values 0.25 and 0.5. The same network is more often verified as robust for labels 0, 2 and 6, i.e. DLV cannot find effective adversarial perturbations around these inputs using an exhaustive search. A difference of up to 63.87% adversarial discovery is observed between labels 1 and 6 for $r = 10$, $\delta = 0.5$. These results show that black box adversarial attacks may have far greater success if targeting specific labels, e.g. by perturbing MNIST digit 1 rather than 6.

B. Relative Robustness of QNNs

Table II shows the relative robustness of two QNNs. The robust test sets $\mathcal{IP}$ for each element in R were generated for each network and set of parameters, by checking the 10,000 images in MNIST test dataset and 10,000 in CIFAR-10 test set using DLV. The size of $\mathcal{IP}$ varies between 7119 and 19801. The number of inputs in $|R|$ for which all perturbations in the associated $\mathcal{IP}$ are correctly classified by the QNN are then recorded, along with the percentage. Figure 4 provides examples of perturbed inputs from $\mathcal{IP}$ for three images in $|R_Q^{FP}|$. The drop in test accuracy of the network after quantization is 0.03% for MNIST and 0.00% for CIFAR-10.

These results show there is very little difference in adversarial robustness between full-precision networks and their quantized counterparts. Relative robustness ranges from 98.63% to 99.69% and changes negligibly with variation of DLV parameters. In each case where an input was not verified as relatively robust, only one of the images in $\mathcal{IP}$ was misclassified. This follows from observations [22] that adversarial examples are found in certain subspaces of the input space.

The quantization applied in this paper works by making small distinct changes to the value of each parameter in the FPNN. We hypothesize that the FPNN's adversarial subspaces of the input space are very similar to those of the QNN. The perturbations are generated in the FPNN's "safe" subspaces of the input space. The small number of misclassifications by the QNN suggest that the safe subspaces of the QNN and FPNN overlap to a high degree.

C. Transfer of Adversarial Attacks to QNNs

The sets $\mathcal{EP}$ for each element in R^{NOT} were generated for each network and set of parameters, by checking the 10,000 images in MNIST and 10,000 in CIFAR-10 test dataset using DLV. Table III shows the number of sets which transfer from R^{NOT} to R^0, and the percentage. Figure 5 provides examples of images from $\mathcal{EP}$ sets corresponding to images in R^0.

Adversarial inputs from $\mathcal{EP}$ transfer well for large values of $|\delta|$, where adversarial noise is sharp, *e.g.* in Table I, (c), (f) and (i) where a single manipulation step δ modifies a pixel by

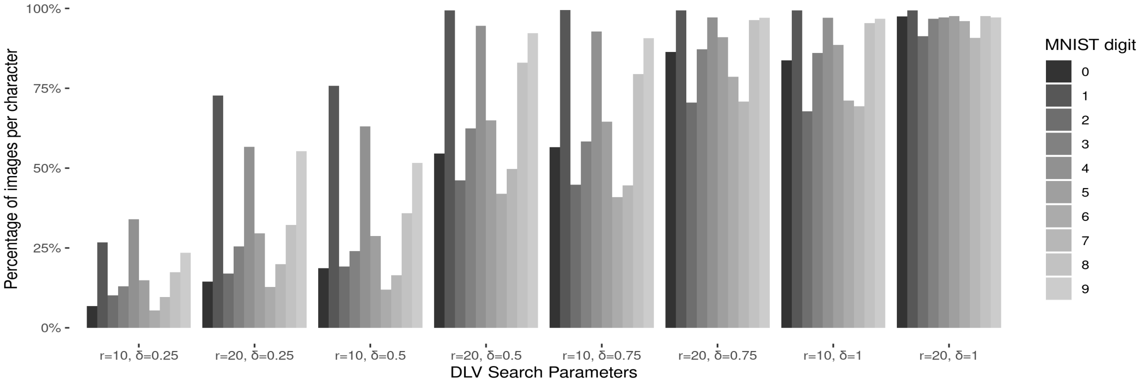


Fig. 6. Percentage of inputs from the MNIST test dataset per label for which at least one effective adversarial perturbation is found by DLV for the FPNN.

TABLE III
ADVERSARIAL TRANSFER: PERCENTAGE OF PERTURBED IMAGES WHICH FOOL THE FPNN, BUT ARE CORRECTLY CLASSIFIED BY THE QNN.

D	$\lvert\delta\rvert$	r	$\lvert R^{NOT}\rvert$	$\lvert R^{0}\rvert$	Adversarial Transfer
MNIST	$\pm\frac{1}{64}$	10	116	70	60.34%
		20	96	76	79.17%
	$\pm\frac{1}{32}$	10	228	176	77.19%
		20	212	146	68.87%
	$\pm\frac{1}{16}$	10	446	392	87.89%
		20	458	392	85.59%
	$\pm\frac{1}{8}$	10	975	925	94.87%
		20	1154	1077	93.33%
	$\pm\frac{1}{4}$	10	1500	1456	97.07%
		20	3548	3448	97.18%
	$\pm\frac{1}{2}$	10	3393	3351	98.76%
		20	7157	7064	98.70%
	± 1	10	8647	8551	98.89%
		20	9888	9756	98.66%
CIFAR-10	$\pm\frac{1}{64}$	10	219	114	52.05%
		20	224	118	52.68%
	$\pm\frac{1}{32}$	10	340	193	56.76%
		20	376	208	55.32%
	$\pm\frac{1}{16}$	10	469	275	58.64%
		20	576	340	59.03%
	$\pm\frac{1}{8}$	10	627	428	68.26%
		20	666	447	67.12%
	$\pm\frac{1}{4}$	10	962	741	77.03%
		20	1286	981	76.28%
	$\pm\frac{1}{2}$	10	1157	993	85.82%
		20	1723	1477	85.72%
	± 1	10	1714	1536	89.61%
		20	2255	2017	89.45%

±1. Where the adversarial noise is more subtle, *e.g.* in Table I, (a), (d) and (g) and Figure 5 where $|\delta| = \pm 0.25$, fewer adversarial attacks transfer. This is shown in Table III where for radius 10 on MNIST, $|\delta| = \pm\frac{1}{64}$ results in 60.34% transfer, whereas $|\delta| = \pm 1$ results in a transfer of 98.89%. Doubling the radius to 20 increases the number of pixels which are modified. For manipulations $|\delta| = \pm\frac{1}{64}$, doubling the radius increases adversarial transfer by 18.83% to 79.17%. Adversarial transfer is more successful when the difference between the original image and perturbed image is more sharp.

When perturbations resemble subtle noise e.g. with $|\delta| = \pm\frac{1}{64}$ and $r = 10$, quantization appears to remove this noise, possibly due to reduced precision acting as a filter, reducing adversarial transfer to 52.05%. Adversarial transfer is more successful for larger values of r, when more pixels have been modified by the tool, reducing the effectiveness of the filter.

The difference in adversarial transfer (Table III) is much more substantial than the loss of robustness caused by quantization (Table II), suggesting that neural network quantization could act as a defense mechanism, particularly when an attacker is unaware of the quantization, without affecting the usefulness of a neural network during normal use.

V. CONCLUSION

It has been widely shown that *accuracy* can generally be preserved after quantization (e.g. [2], [33]), but one might expect that quantizing a neural network would negatively affect its *robustness*. However, this paper shows that not only can robustness be preserved, it can also be *increased* by the quantization process.

Quantized neural networks, which in this paper are only 0.03% less accurate than full-precision, are found to be robust to adversarial perturbations for 98.07–99.69% of the inputs that the corresponding FPNN is robust to. Notably, our results suggest that quantizing a neural network can reduce the transfer of adversarial examples to 52.05%, possibly because quantization acts as a filter for adversarial noise. In addition, we find that neural networks are significantly more robust for certain classes in a dataset. We observed up to 63.87% difference in robustness between two classes in MNIST.

The small drop in precision and robustness after quantization alongside a large drop in adversarial transfer suggests that quantization can improve a network's resilience to adversarial attack overall. Quantization of a network could therefore be favourable in applications where robustness is the goal, rather than targeting resource constrained hardware accelerators which, until now, has been the primary motivation for neural network compression.

A. Related Work

The literature studying adversarial attacks and neural network robustness splits into two big groups of methods. *Machine learning approaches* [14], [35] usually study these problems from a statistical point of view, including evolutionary or gradient descent algorithms for producing attacks and modelling defenses [31], or studying geometric properties of classifiers relative to the given data [10].

In contrast, *verification approaches* [15], [17], [18], [32], [39] study methods that can certify, or give proven guarantees, of certain properties of neural networks. This paper belongs to the second group of methods. Similarly to machine learning approaches, we generate perturbations and evaluate network performance. However, our focus is on properties of neural networks that were first *verified* and then quantized. This distinguishes our methodology from other machine-learning approaches that study robustness (but not verification) of quantized neural networks [4], [19], [28], [30].

There exist related papers that verify properties of binarized [5], [27] or quantized [3] neural networks. In these approaches binarization and quantization are used as means of reducing the proof search space and/or satisfying the practical engineering requirements of the chosen prover. For example, there are limitations to how real numbers are implemented in constructive provers like Coq, while SAT and SMT solvers cannot handle real-valued non-linear functions. In contrast to these papers, we consider the scenario when a full precision neural network (of arbitrary complex architecture) is verified, and is later quantized when deployed in edge devices. We are interested in the scenario where verification after quantization is not feasible because it may be a dynamic, possibly even runtime, process and because there may not be enough resources to perform verification after quantization or at run-time.

Finally, in machine learning literature, quantization has been used successfully as a defense mechanism against adversarial attack. This can be achieved by compressing a network's input or parameters.

1) Input Quantization as a Defense: Input compression can defend against adversarial attack by acting as a filter for adversarial noise. The aggressive quantization which would be required to remove all noise could result in a loss of image quality, making classification more difficult. In [28] variable JPEG compression is used to apply stronger quantization to image regions which have lower classification saliency.

2) Network Quantization as a Defense: Binarized nets were shown to misclassify fewer adversarial attacks than full precision networks, with substantially fewer misclassifications when binarized networks were presented with white-box attacks, cf. [11]. In [4] attack methods were tested on a full precision model and 4 low-bitwidth models. Poor transfer of attacks generated on each of these networks was observed when tested on other networks, with the lowest transfer to and from binarized networks. Attacks which rely on non-approximated gradients applied directly to QNNs are almost as effective as when applied to FPNNs, despite these attacks not transferring well across models of different precision. The authors suggest this is due to gradients of the different models misaligning.

Literature varies in the methods of quantization. For example, [19] adds a quantization layer as the input layer of a network. In this case, quantization is not linear, the threshold for each quantized value is determined during training. This layer acts as a filter, followed by a full-precision network. Applying quantization to the full network [30], one can learn the quantization thresholds for the activations in the network. Adversarial examples are included during the training which determines the threshold values.

Our work confirmed the previous results showing that quantisation of neural networks can improve adversarial robustness. But in addition, our studies highlighted several aspects of this problem that have not been studied before.

Firstly, most of the previous approaches measured how the level of quantization applied to the network affects its robustness against a *fixed* number of *randomly generated* attacks. We answered a different question: given a chosen mode of quantisation of a neural network, do changes in subtlety of the perturbations (i.e. varying search radius and manipulation step) influence its effectiveness against the attack. The previous papers found that more quantization resulted in a greater increase in the accuracy against adversarial attack. We found that, the more subtle the perturbation is, the more effective quantization is against adversarial attack. For attacks that perturb more pixels, and perturb more drastically, quantization may be less effective. This effect has not been observed prior to this work.

Secondly, due to the focus on verification, we apply *post-training quantization* on a verified neural network. Thus, unlike the related papers, we do not retrain during or after quantization. Retraining modifies the parameters of the network as well as its precision, resulting in two distinct networks. Keeping a tighter link between the verified full precision network and its quantized counterpart was key for the chosen verification scenario.

B. Future Work

1) Increasing the variety and the complexity of neural networks: All models used in this work are multi-layer perceptrons. An investigation on how different types of networks respond to quantization would be interesting. As we generated attacks in a black-box manner, all our experiments could easily be repeated with different models. Our preliminary studies of varying neural network architectures showed similar trends to those we presented in this paper, but this question requires a more systematic study.

2) Comparing DLV performance with other verification tools: DLV was used in this paper because it generates an exhaustive set of perturbed images within the search parameters. DLV perturbs images by randomly picking pixels and modifying the surrounding pixels. It often generates horizontal lines in the background pixels. It does not use domain knowledge to recognise the “important” pixels, so it would be interesting to repeat the experiment with different verification methods.

3) Relation to one-pixel attacks: The subtlety of adversarial perturbations generated by DLV was varied in our experiments, however these perturbations are more perceptible than recent literature on adversarial attacks. The extreme example is effective adversarial perturbations by modifying just a single pixel [34]. Future work would investigate the transfer of imperceptible adversarial attacks from full precision networks to quantized networks.

References

[1] Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/

[2] Bacchus, P., Stewart, R., Komendantskaya, E.: Accuracy, Training Time and Hardware Efficiency Trade-Offs for Quantized Neural Networks on FPGAs. In: Applied Reconfigurable Computing. Architectures, Tools, and Applications - 16th Int. Symp., Proc. vol. 12083, pp. 121–135. Springer (2020)

[3] Bagnall, A., Stewart, G.: Certifying the true error: Machine learning in coq with verified generalization guarantees. In: The 33rd AAAI Conf. on AI, The 31st Innovative Applications of AI Conf., The 9th AAAI Symp. on Educational Advances in AI. pp. 2662–2669 (2019)

[4] Bernhard, R., Moëllic, P., Dutertre, J.: Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks. In: Int. Conf. on Cyberworlds. pp. 308–315. IEEE (2019)

[5] Cheng, C.H., Nührenberg, G., Huang, C.H., Ruess, H.: Verification of binarized neural networks via inter-neuron factoring. In: Working Conf. on Verified Software: Theories, Tools, and Experiments. pp. 279–290. Springer (2018)

[6] Chollet, F., et al.: Keras: Deep learning library for theano and tensorflow. URL: https://keras. io/k **7**(8), T1 (2015)

[7] Courbariaux, M., Bengio, Y.: Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. CoRR **abs/1602.02830** (2016)

[8] Deng, L.: The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine **29**, 141–142 (2012)

[9] Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: 24th Int. Joint Conf on AI (2015)

[10] Engstrom, L., Tsipras, D., Schmidt, L., Madry, A.: A rotation and a translation suffice: Fooling cnns with simple transformations. CoRR **abs/1712.02779** (2017)

[11] Galloway, A., Taylor, G.W., Moussa, M.: Attacking binarized neural networks. In: 6th Int. Conf. on Learning Representations, Conf. Track Proc. OpenReview.net (2018)

[12] Goodfellow, I., et al.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)

[13] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples. In: 3rd Int. Conf. on Learning Representations, Conf. Track Proc. (2015)

[14] Guo, C., Rana, M., Cissé, M., van der Maaten, L.: Countering Adversarial Images using Input Transformations. In: 6th Int, Conf. on Learning Representations, Conf. Track Proc. (2018)

[15] Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Int. Conf. on Computer Aided Verification. pp. 3–29. Springer (2017)

[16] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: Training neural networks with low precision weights and activations. The Journal of Machine Learning Research **18**(1), 6869–6898 (2017)

[17] Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: International Conference on Computer Aided Verification. pp. 97–117. Springer (2017)

[18] Katz, G., et al.: The Marabou framework for verification and analysis of deep neural networks. In: CAV 2019, Part I. LNCS, vol. 11561, pp. 443–452. Springer (2019)

[19] Khalid, F., et al.: Qusecnets: Quantization-based defense mechanism for securing deep neural network against adversarial attacks. In: 25th IEEE Int. Symp. on On-Line Testing and Robust System Design. pp. 182–187. IEEE (2019)

[20] Lane, N.D., Georgiev, P.: Can deep learning revolutionize mobile sensing? In: Proc. of the 16th Int Workshop on Mobile Computing Systems and Applications. pp. 117–122. ACM (2015)

[21] Lin, J., Gan, C., Han, S.: Defensive Quantization: When Efficiency Meets Robustness. In: 7th Int. Conf. on Learning Representations (2019)

[22] Ma, X., Li, B., Wang, Y., Erfani, S.M., Wijewickrema, S.N.R., Schoenebeck, G., Song, D., Houle, M.E., Bailey, J.: Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. In: 6th Int. Conf. on Learning Representations, Conf. Track Proc. (2018)

[23] Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On Detecting Adversarial Perturbations. In: 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings (2017)

[24] Miotto, R., Wang, F., Wang, S., Jiang, X., Dudley, J.T.: Deep learning for healthcare: review, opportunities and challenges. Briefings in bioinformatics **19**(6), 1236–1246 (2017)

[25] Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proc. of the IEEE conf. on computer vision and pattern recognition. pp. 1765–1773 (2017)

[26] Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proc. of the IEEE conf. on computer vision and pattern recognition. pp. 2574–2582 (2016)

[27] Narodytska, N., Kasiviswanathan, S., Ryzhyk, L., Sagiv, M., Walsh, T.: Verifying properties of binarized deep neural networks. In: 32nd AAAI Conf. on AI (2018)

[28] Prakash, A., Moran, N., Garber, S., DiLillo, A., Storer, J.A.: Protecting JPEG images against adversarial attacks. In: 2018 Data Compression Conf. pp. 137–146. IEEE (2018)

[29] Radu, V., Kaszyk, K., Wen, Y., Turner, J., Cano, J., Crowley, E.J., Franke, B., Storkey, A.J., O'Boyle, M.: Performance Aware Convolutional Neural Network Channel Pruning for Embedded GPUs. In: IEEE Int. Symp. on Workload Characterization. pp. 24–34. IEEE (2019)

[30] Rakin, A.S., Yi, J., Gong, B., Fan, D.: Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions. CoRR **abs/1807.06714** (2018)

[31] Samangouei, P., Kabkab, M., Chellappa, R.: Defense-gan: Protecting classifiers against adversarial attacks using generative models. In: 6th Int. Conf. on Learning Representations, Conf. Track Proc. OpenReview.net (2018)

[32] Singh, G., Gehr, T., Püschel, M., Vechev, M.T.: An abstract domain for certifying neural networks. PACMPL **3**(POPL), 41:1–41:30 (2019)

[33] Su, J., et al.: Accuracy to Throughput Trade-Offs for Reduced Precision Neural Networks on Reconfigurable Logic. In: Applied Reconfigurable Computing. Architectures, Tools, and Applications - 14th Int. Symp., Proceedings. pp. 29–42 (2018)

[34] Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Trans. Evolutionary Computation **23**(5), 828–841 (2019)

[35] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: 2nd Int. Conf. on Learning Representations, Conf. Track Proc. (2014)

[36] Tramèr, F., Papernot, N., Goodfellow, I.J., Boneh, D., McDaniel, P.D.: The space of transferable adversarial examples. CoRR (2017)

[37] Umuroglu, Y., Fraser, N.J., Gambardella, G., Blott, M., Leong, P.H.W., Jahre, M., Vissers, K.A.: FINN: A framework for fast, scalable binarized neural network inference. In: Proceedings of the 2017 ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays. pp. 65–74 (2017)

[38] Wang, E., Davis, J.J., Zhao, R., Ng, H.C., Niu, X., Luk, W., Cheung, P.Y., Constantinides, G.A.: Deep neural network approximation for custom hardware: Where we've been, where we're going. ACM Computing Surveys (CSUR) **52**(2), 40 (2019)

[39] Wicker, M., Huang, X., Kwiatkowska, M.: Feature-guided black-box safety testing of deep neural networks. In: Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems. pp. 408–426. Springer (2018)