Adversarial Robustness of Quantized Embedded Neural Networks

Rémi Bernhard[1], Pierre-Alain Moellic[1], Jean-Max Dutertre[2]

[1]CEA Tech, Systemes et Architectures Sécurisées (SAS),
Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne
Gardanne, France
[2]Mines Saint-Etienne, CEA-Tech,
Centre CMP,
Gardanne, France

remi.bernhard@cea.fr, pierre-alain.moellic@cea.fr, dutertre@emse.fr

Keywords Machine Learning; Neural Netwroks; Adversarial Machine Learning; Adversarial Examples; Embedded Systems; Security.

1 Abstract

As the will to deploy neural network models on embedded systems grows, and considering the related memory footprint and energy consumption requirements, finding lighter solutions to store neural networks such as parameter quantization and more efficient inference methods becomes major research topics. Parallel to that, adversarial machine learning has risen recently, unveiling some critical flaws of machine learning models, especially neural networks. In particular, perturbed inputs called *adversarial examples* have been shown to fool a model into making incorrect predictions. In this paper, we investigate the adversarial robustness of quantized neural networks under different attacks. We show that quantization is not a robust protection when considering advanced threats and may result in severe form of *gradient masking* which leads to a false impression of security. However, and interestingly, we experimentally observe poor transferability capacities between full-precision and quantized models and between models with different quantization levels which we explain by the *quantization value shift* phenomenon and gradient misalignment. We finally explore how these results can be exploited with an ensemble-based defense.

2 Introduction

2.1 Context

Neural networks achieve state-of-the art performances in various domains such as speech translation or image recognition. These outstanding performances have been allowed – among others – by tremendous computation power (e.g., popularization of GPU) and the resulting trained architectures come with thousands or even millions parameters. As the desire to run pre-trained neural network based application (e.g image recognition) on embedded or mobile systems grows, one must investigate the ways to solve the practical issues involved. First of all, the memory footprint can quickly be a limiting factor for constrained devices. For example, a typical ARM Cortex-M4-based microcontroller such as STM32F4 has up to 384 KBytes RAM and a maximum of 2MBytes of Flash memory[1]. Secondly, inference cost in terms of energy is critical for devices like mobile phones or a large variety of connected objects (e.g., industrial sensors). Thirdly, inference speed is necessary to avoid critical latency issues.

[1]https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html

Some APIs like the Android Neural Network API (NNAPI[2]) have been already developed to allow to run efficiently trained models with famous frameworks (TensorFlow[3], etc.) on Android systems. Tensorflow Lite (TFLite[4]) allows to transfer pre-trained model to mobile or embedded devices thanks to model compression techniques and 8-bit post-training weights quantization. ARM-NN[5] is another SDK that does the link for applications between various machine learning frameworks and diverse Cortex CPUs or Mali GPUs types (note that CMSIS-NN[6] is dedicated to Cortex-M MCU). STMicroelectronics also proposes an AI expansion pack for the STM32CubeMX, called X.Cube.AI[7], to map pre-trained neural network models into different STM32 microcontroller series thanks to 8-bit post-training quantization and other optimization tricks related to the specificity of these platforms.

On a more theoretical side, research about reducing the number of parameters to directly impact the memory footprint of models [1–5], or developing quantization schemes coupled with efficient computation methods to reduce inference time and energy consumption has arisen [6–15]. At the same time, neural networks have been shown to be vulnerable to malicious tampering of inputs [16]. From a clean observation correctly classified by a model, an adversary optimally crafts a so-called *adversarial example*, which is very similar to the clean observation and fools the model. Many attack methods ([17–22] for some of the most famous) and defense methods ([16, 23–26] for some of the most famous) have been developed and evaluated in benchmarks or competition tracks such as the NIPS *Adversarial Vision Challenge*[8].

2.2 Motivation and related works

In terms of security, as embedded systems with neural networks models become ubiquitous, it is a particularly interesting topic to evaluate the robustness of state-of-the-art quantization methods under different threat models. Moreover, studying the transferability of adversarial examples between original (i.e. full precision) and quantized neural networks may at the same time highlight weaknesses or strengths of future embedded systems, and allow to better understand if quantization in itself could be a relevant defense against adversarial examples or, on the contrary, exacerbates these flaws.

Some authors have already investigated the link between quantization and robustness. [27] claimed that neural networks trained with weights and activation values binarized to $\{-1, 1\}$ have an interesting robustness against adversarial examples. However, this robustness was demonstrated thanks to the only MNIST dataset and use stochastic quantization. This quantization scheme induces the stochastic gradient phenomenon [28], which can mislead to the true efficiency of this defense by causing what [29] called *obscurity*. [30] tries to explain some weaknesses of quantization-based defense methods against adversarial examples. They show experimentally that these defense methods can, in fact, denoise an adversarial example or enlarge its perturbation, depending on the size of the perturbation in the input space and the number of bits used for quantization. Thus, quantization can participate in an error amplification or attenuation effect. However, they only apply the FGSM attack [17] in a white-box setting against simple activation quantization. Although focused on model compression (pruning), [31] studied the robustness of quantized neural networks against adversarial examples with a fixed-point quantization scheme applies to both weight and activation values, no less than 4-bit model, and a restricted set of (gradient-based) attacks. [32] proposes a defense method based on activation quantization coupled with adversarial training [23], which has been shown by [30] to introduce gradient masking [33]. Interestingly, [34] notes that the gradients obtained via the use of a *Straight Through Estimator* (hereafter STE, [35]) – a common technique to compute gradients when quantization operations lead to differentiability issues – may not be representative of the true gradient. This observation leads to questions about efficiency of gradient-based attacks against quantized neural networks, and strengthens up the motivation to study gradient masking issues and black-box attacks against such models. The authors propose a Mixed Integer Linear Programming (MILP) based attack, which shows good results on the MNIST data set but is not scalable to large neural networks, due

[2] https://developer.android.com/ndk/guides/neuralnetworks
[3] https://www.tensorflow.org/
[4] https://www.tensorflow.org/lite
[5] https://developer.arm.com/products/processors/machine-learning/arm-nn
[6] https://github.com/ARM-software/CMSIS_5
[7] https://www.st.com/en/embedded-software/x-cube-ai.html
[8] https://www.crowdai.org/challenges/adversarial-vision-challenge

to computation cost issues.

2.3 Contributions

In this work, we study the robustness of natural and quantized models and against adversarial examples under different threat models against various types of attacks. First, we show that quantization in itself offers poor protection against various well-known adversarial crafting methods and we explain why activation quantization can lead to severe gradient masking, a phenomenon which leads to non-useful gradients to craft adversarial examples [21] and causes ineffective defense [29]. Then, we show very poor transferability capacities of adversarial examples between full-precision and quantized models and between quantized models with different bitwidths. We advance hypothesis to explain it, including a quantization shift phenomenon and gradient misalignment.

3 Background

3.1 Quantization of neural networks

The purpose of this article is to study the impact of quantization techniques on the adversarial robustness, for embedded neural networks. However, other complementary approaches are extensively studied to compress as well as to speed up models at inference time. During inference, energy consumption grows with memory access, which itself grows with memory footprint. Thus, reducing the number of parameters has been logically investigated. For example, [1] show, for specific architectures and datasets, that some of the parameters are predictable from the others. [4] develop a three-step method (pruning, clustering, tuning) to efficiently compress a neural network achieving a reduction of AlexNet memory footprint by a factor of 35. [2] propose a method to reduce the memory size of a convolutional neural network by pruning connections based on a deterministic rule. This method is also coupled with weight binarization ([6]) and an efficient hardware architecture on a FPGA in order to reduce inference time.

Reducing the precision of the weights or developing efficient computation methods for some precise format of weight values is an important field of investigation. Quantization can be performed as a post-training process or during training. For the first case, as previously described in introduction, several tools have been recently proposed to map full precision pre-trained models for inference purpose (TFLite, ARM-NN, STMCubeMX A.I.[9]) by coarsely quantizing some weights into – usually – no more than 8-bit integers. More advanced methods propose clustering methods [5] or information theoretical vector quantization methods (inspired by [1]) such as [3] who achieved about 20 times compression of the model with only 1% loss of classification accuracy on the Imagenet benchmark.

In this article, we focus our work on quantization techniques at training time since these approaches enable to reach state-of-the-art performance with lower bitwidth precision. Hereunder, we detail some of the most popular works on that field that we consider for our experimentations.

Binary Connect and Binary Net. [6] presents a method to train neural networks with weights w binarized to $w_b \in \{-1, 1\}$. During training, weights are binarized for the forward pass, and as the binarization operation can be not differentiable or lead to the vanishing gradient problem, the STE given in Equation 1 is used for the backward pass:

$$\frac{\partial C(w)}{\partial w} \approx \frac{\partial C}{\partial w}\bigg|_{w=w_b} \mathbf{1}_{|w|\leq 1} \tag{1}$$

Where C is the cost function and $\mathbf{1}(.)$ the indicator function. [7] pursue this idea by training binary networks (BNN) with weight values w and activation function a^k values binarized to $(w_b, a_b^k) \in \{-1, 1\}^2$. During the backward pass, the authors used the same STE principle for activations as in Equation 1 above. Improvements have been proposed as in [36] by adding regularization and more complex approximation of the derivative on the backward pass.

[9] https://www.st.com/en/embedded-software/x-cube-ai.html

Xnor Net. [9] binarizes weigths and activation values no more to $\{-1, 1\}$ but to $\{-\alpha, \alpha\}$ with $\alpha \in \mathbb{R}^{*,+}$. They formalize the search of the best binarization approximation of the real-valued weights as the following optimization problem.:

$$\underset{B,\alpha}{\operatorname{argmin}} \|W - \alpha B\|_2 \tag{2}$$

Where W is the weight matrix and B is a matrix with only -1 and 1. During the backward pass, a STE is used.

Ternarization. In [10], the authors propose a method to train a neural network with weight values ternarized to $\{-1, 0, 1\}$ during the forward pass. [11] also propose a method to train a neural network with weight values ternarized to $\{-\alpha_l, 0, \alpha_u\}$ during the forward pass, where α_l, $\alpha_u \in \mathbb{R}_+^{2,*}$ and α_l and α_u are updated during training.

Low bitwidth quantization. [12] successfully train networks with good precisions on MNIST and CIFAR10 data sets while limiting the bitwidth of the weights values to 16 bits and using stochastic rounding. [13] proposes a method to train neural networks with low-bitwidth weight values, gradients and activation function values. They claim that taking advantage of this technique during the forward pass could help speed up the training of neural network on resource-limited hardware, and naturally speed up the inference. For a real value $x \in [0, 1]$ and a number of bits n, the function

$$Q(x, n) = \frac{round((2^n - 1)x)}{2^n - 1} \tag{3}$$

is the quantization function used for weights, activation values and gradients. The weight and activation values are quantized on the forward pass only. The authors also found that quantizing gradients on the backward pass (with a STE) was requisite. $Q(x, n)(2^n - 1)$ results in 2^n values, each of them representable with a n bits integer. During the forward pass, one can take advantage of the bit convolution kernel method (see [13] for details) with respect to the $Q(x, n)(2^n - 1)$ values, and then scale afterwards with the $2^n - 1$ value.
[14] proposes another weight quantization method with a constraint on the number of "1" in the binary representation of weights, along with some efficient computation method.
[15] proposes a method which involves distillation and quantization of the weight values to decrease the storage size of a model. For the quantization part, given some weight value w, it is first scaled to a value v in $[0, 1]$, then mapped with a function Q to the nearest mapping points among the $s + 1$ points in $[0, 1]$ and then scaled back to the original scale.

In this article, we use the Binary Net method and Binary Connect respectively from [7] and [6], and the quantization method (Dorefa-Net) from [13].

3.2 Adversarial machine learning

Machine learning systems have been shown to be vulnerable against different types of attacks threatening their confidentiality, integrity or accessibility. We can distinguish three different types of attacks:

- *Data/Model leakage* occurs once the model has been trained. An adversary aims at stealing models parameters or architecture, or stealing confidential or private (training) data [37–39].
- With *data poisoning*, which steps in during the training phase, an attacker targets the integrity or availability of the system according to the level of the perturbation. In order to decrease the model accuracy, corrupted data are introduced in the training set when the data are collected in the physical world or directly in the model input domain [40, 41].
- Adversaries may also alter the inputs at inference time, by crafting malicious observations looking like clean ones but designed such as to fool the model [16, 17], striking the model integrity.

Here we focus on the latter type of integrity-based attack, i.e *adversarial examples crafting*.

3.2.1 Adversarial examples

Adversarial examples are highly worrying threats to machine learning. Roughly stated, considering a classifier model, an adversarial example is a slightly modified version of a correctly classified clean example in a way such that the classifier will output two different classes for those two examples. The reason of existence of adversarial examples lead to various hypothesis. [16] propose a first explanation to the existence of adversarial examples: they would be in fact located in low-probability pockets of the input space. On their side, for [17], it is some local linearity assumption which eases the crafting of adversarial examples, and not the global non-linear nature of neural networks. In [42], the authors give a more geometric explanation to the existence of adversarial examples, saying that the learned boundary "extends beyond the submanifold of sample data and can be – under certain circumstances – lying close to it" (*boundary tilting* effect), and argue that the linearity assumption is not sufficient to explain adversarial examples that are – for some of them – the result of overfitting issues. [43], based on theoretical results linking the generalization error to the average distance to a misclassified point for a very particular type of dataset, include high dimensionality as a possible power factor of adversarial examples. Another common hypothesis, used among others to detect adversarial examples, is that adversarial examples are not on the data manifold [44]. Recently, [45] show that adversarial examples are the consequence of *non-robust features* derived from patterns with a big predictive power, yet these patterns are meaningless to humans and they can be adversarially modified to fool the target classifier.
More precisely, given a classifier model M learning a mapping function $f : \mathbb{R}^m \to \{1, ..., K\}$, given an initial clean observation $x \in \mathbb{R}^m$, given a target label $t \in \{1, ..., K\}$, a targeted adversarial example $x' \in \mathbb{R}^m$ crafted from a correctly classified x is defined such as $f(x) \neq f(x') = t$ and $d(x, x') < \epsilon$ with d a distance function being often the distance derived from the l_2 or l_∞ norm. Based on [16], the search of such an adversarial example can be written as:

$$\begin{aligned} \epsilon &= \underset{\epsilon}{\operatorname{argmin}} \|\epsilon\|_p \\ s.t \quad & f(x+\epsilon) = t \ (\textit{targeted attack}) \\ s.t \quad & f(x+\epsilon) \neq f(x) \ (\textit{untargeted attack}) \end{aligned} \tag{4}$$

Usually, the adversary may also wants that $x + \epsilon$ be bounded (for example, $x + \epsilon \in [0, 1]$ as in [16]).

3.2.2 Attacks

In this article, we use five different adversarial crafting methods. These attacks are presented in their untargeted version, where adversarial examples are crafted from a clean observation x of label y, $logit_j(x)$ designates the logit output for the j^{th} class, and $f_j(x)$ designates the softmax output for the j^{th} class.

Fast Gradient Sign Method (FGSM). Presented by [17], this method, derives an adversarial example x maximizing $J(\theta, x + \alpha, y) - J(\theta, x, y)$ with respect to α ,given that $\|\alpha\|_\infty < \epsilon$, by performing a linear approximation of the loss function $J(\theta, x, y)$ around x, one gets:

$$x' = x + \epsilon Sign(\frac{\partial J}{\partial x}(\theta, x, y)) \tag{5}$$

x' is then clipped to respect a possible box constraint (for images for example one may want $x' \in [0, 1]$)

Basic Iterative Method (BIM) [46] presents the Basic Iterative Method, derived from the FGSM method, which allows to craft targeted adversarial perturbations. Given a maximum adversarial perturbation ϵ:

$$x_0 = x, \ x_{n+1} = Clip_{B_\infty(x,\epsilon)}(x_n + \alpha sign(\frac{\partial J}{\partial x}(\theta, x_n, y))) \tag{6}$$

where $B_\infty(x, \epsilon)$ is the l_∞-ball of radius ϵ and center x, and we set $\alpha = \frac{\epsilon}{T}$ with T the total number of iterations. In fact, we just repeat the targeted FGSM method for K iterations, performing

clipping at each iteration. x' is then clipped to respect a possible box constraint (for images for example one must have $x' \in [0, 1]$).

Carlini-Wagner l_2 (CWl2). Presented by [18], the Carlini-Wagner l_2 method consists of considering the following objective:

$$\begin{aligned} \min_{\epsilon} \quad & \|\epsilon\|_2 + cF(x+\epsilon, y), \\ s.t. \quad & x + \epsilon \in [0, 1] \end{aligned} \tag{7}$$

where:

$$F(x+\epsilon, y) = max(logit_y(x+\epsilon) - max_{j \neq y} logit_j(x+\epsilon), -\kappa) \tag{8}$$

with $\kappa \geq 0, c > 0$. We set $\kappa = 0$ and thus we have $F(x+\epsilon, y) = 0 \iff label(x) \neq y$. $c \in \mathbb{R}^+$ is a constant for which binary search is performed a decided amount of time. Then, the change of variable $x = \frac{1}{2}(tanh(w) + 1)$ is performed to get rid of the box constraint. The resulting optimization problem with respect to the new variable w can then be solved with classical optimization methods like Stochastic Gradient Descent (SGD) or Adam.

SPSA attack. In [29], the authors propose a very effective gradient-free attack to evaluate defense strategies. The authors propose the constrained optimization problem given in Equation 9 that they solve by using the Adam update rule,approximating the gradients with finite difference estimates thanks to the SPSA (*Simultaneous Perturbation Stochastic Approximation*, [47]) technique which is suitable for noisy high dimensional optimization problems, and performing clipping at each iteration to respect the constraint $\|\alpha\|_\infty < \epsilon$:

$$\begin{aligned} \min_{\epsilon} \quad & logit_y(x+\alpha) - max_{j \neq y} logit_j(x+\alpha), \\ & s.t. \, \|\alpha\|_\infty < \epsilon \end{aligned} \tag{9}$$

Zeroth Order Optimization (ZOO). The ZOO attack [22] is based on the CWl2 attack with a discrete approximation of the gradients:

$$g'(x)_i \simeq \frac{g(x + he_i) - g(x - he_i)}{2h} \tag{10}$$

where e_i is the basis vector with only the i^{th} element equal to 1, the others equal 0 and h is a small constant. The ZOO attack does not consider the logits values as the CWl2 attack does but the logarithm of the softmax output values, i.e we have:

$$F_{ZOO}(x+\epsilon, y) = max(log(f_y(x+\epsilon)) - max_{j \neq y} log(f_j(x+\epsilon)), -\kappa) \tag{11}$$

Characteristics. We sum up the main characteristics of these attacks in table 1.

	FGSM	BIM	CWL2	SPSA	ZOO
Gradient-based	✓	✓	✓		
Gradient-free				✓	✓
one-step	✓				
iterative		✓	✓	✓	✓
$\boldsymbol{l_\infty}$	✓	✓		✓	
$\boldsymbol{l_2}$			✓		✓

Table 1: Main characteristics of the considered adversarial examples crafting methods

3.2.3 Defenses

Many defenses have been investigated to counter adversarial examples. As the core of this article does not either aim at testing defense schemes or a specific attack, we refer to [48] for an overview of

protections. The authors distinguish mainly the reactive defenses, which encompass pre-processing inputs and detection methods [25, 26, 44, 49–56], the proactive defenses, which encompass techniques to make a network in itself more robust to adversarial examples [17, 23, 24, 57–62], and provable defense methods [63–67].

3.3 Threat model

3.3.1 Main characteristics of threat models

The threat model encompasses assumptions about the adversary's goals, capabilities and knowledge.

Adversarial goal Here we focus on an adversary that aims to fool a supervised model at inference time. From a clean observation x correctly labeled as y, the adversary wants to craft an adversarial example x' labeled as a precise class $t \neq y$ (*targeted attacks*) or any class $y' \neq y$ (*untargeted attacks*). Given some threat model, a defense method claiming robustness against untargeted attacks is stronger than a one claiming robustness against targeted attacks. Similarly, it is often more difficult to craft targeted adversarial examples than untargeted ones.

Adversarial capability It is crucial to properly define how much an adversary can alter a process of the machine learning pipeline. In the scope of adversarial examples crafting, the adversary's ability is almost all the time defined as an upper bound ϵ of the distance $D(x, x')$ between a clean observation x and the adversarial example x' crafted from x. The distance D is derived from an l_p norm, usually the l_0, l_1, l_2 or l_∞ norm.

Adversarial knowledge Traditionally, two main different settings are used to describe the way an adversary can operate. Each setting contains its own nuances but for the sake of simplicity we only present those we will later consider in this article.

In the *white-box* setting, the adversary is assumed to have a full access to the target model. This includes the type of model (SVM, architecture of a neural network, etc.), the parameters of the model (network's weights, etc.), any preprocessing component, etc.

In a rigorous *black-box* setting, the adversary has no information about the model but can (only) query it (in a limited or unlimited way). However, this setting can be loosened (some talk about *grey-box* settings) according to the kind of information the adversary can get when querying the model (full prediction outputs – softmax or logits outputs – or just the predicted label) as well as a full or partial access to the training data. Note that in order to thwart a possible restriction concerning the access to the training set, [21] proposes a way to train a substitute model by synthetically generating data labeled by the neural network under attack.

3.3.2 Specificity induced by embedded models

For our experiments, we do not consider a strict black-box setting since we assume an attacker will try to transfer the adversarial examples from one full-precision model to a quantified one or one quantified model to another one with a different level of quantization. This means that we assume a worst-case scenario where an adversary knows the model architecture and can query it without limitation with a full access to the softmax output. Moreover, since we use classical image collections, we assume that the attacker has access to the same datasets. Considering the global context of embedded neural networks for inference, numerous *popular* and proven architectures (such as the ResNet networks) are directly applied for a large scope of applications. Then, a scenario where an adversary craft malicious inputs from a known full precision models to attack an optimized (i.e. quantized) model in, for example, a mobile device is a realistic scenario.

However, we must highlight an important characteristic of the threat models when dealing with an attacker who aims to target an embedded machine learning model. In that case, both the *architecture* of the model itself *and* its *implementation* are important. That means we need to consider a twofold white/black box paradigm: on one hand, the adversary can have – classically – full or no knowledge of the model architecture (abstraction level) and, on the other hand, he may also have full or no knowledge of the model implementation within the *target device* (physical level).

As previously said, in this work and more particularly in the section dealing with the use of quantized networks as a defense mechanism, we mainly focus on a threat model where an attacker has a white-box access to the model architecture but not for its implementation in the target device. Then, the most natural scenario corresponds to an attacker that tries to directly transfer the adversarial examples crafted from a full precision model to the embedded system.

We are conscious of the limitation of such a scenario since, obviously, the attacker may guess – thanks to information about the hardware platform (i.e. memory, precision constraints, etc.) – relevant optimization methods applied to the model (weights and activations quantization, pruning...). That means an advanced adversary could try to craft adversarial examples from a quantized model of its own (without knowing the quantization method used for the target device neither if additional optimizations have been performed).

4 Experiments

We start by performing adversarial robustness experiments for full-precision and quantized models with gradient-based and gradient-free attacks that may be used in black-box settings (unfeasible gradient computation). In both case, we find that quantization does not provide reliable protection. We notice that quantization causes some gradient masking, which tampers some gradient-based attacks (FGSM and BIM) and may prevent gradient-free attacks relying on the approximation of the gradient of the output function (ZOO) to perform well. However, some gradient-based attacks using the STE to mount gradient-based attacks (CWl2) seem to avoid the gradient masking effect. Secondly, we perform transferability experiences between full-precision and quantized models and show poor transferability capacities, which we explain with the *quantization value shift* phenomenon and gradient misalignment.

4.1 Data

We conduct our experiments on the CIFAR10[10] and SVHN[11] (Street View House Numbers) two classical natural scene image datasets. CIFAR10 is composed of 60,000 images, with 10 classes. We use a training set of size 50,000 and a testing set of 10,000. The SVHN dataset is composed of 99,289 images, with 10 classes. We use a training set of size 73,257 and a testing test of 26,032.

4.2 Experience details

For each dataset, we trained a full-precision (32-bit floating point) neural network (hereafter called "float model" in tables), and various quantized neural networks. More precisely, for each data set, the neural network architecture is based on the one presented in [7]. It consists of convolutional blocks, each of them being the stack of a convolution layer, a batch-normalization layer and the relu activation function, followed by dense blocks being a stack of a dense layer, a batch-normalization layer and the ReLu activation function. At the top of the network, we chose a dense layer with the softmax activation function, contrary to [7], where there is no activation function but a final batch normalization layer. Models architecture are detailed in Appendix A. Full-precision and quantized networks were trained with the cross-entropy loss, contrary to [7] where the hinge-loss is used, as we found it to converge faster. The optimization is done with Adam [68], using a staircase decay for the learning rate.

Four different quantization bitwidth are considered: 1,2,3 and 4 bits. For each bitwidth, we consider quantization on the weights only (weight quantization) or the weights and the output of each convolutional or dense block (full quantization). For the weight binarization, the full binarization and the 2,3,4-bit quantization, we use respectively the Binary Connect method [6], the Binary Net method [7] and the Dorefa Net method [13] described in 3.1. The input layer and the last dense layer are never quantized, to allow an efficient training [7].

The performance (accuracy) of each model on the test sets is presented in Table 2. Quantization does not affect significantly the accuracy, except for fully binarized models which achieve only 0.79 and 0.89 accuracy on CIFAR10 and SVHN respectively, which represents a non negligible drop of performance. For quantized models with more than 1 bit, the test set accuracy is comparable to

[10] https://www.cs.toronto.edu/ kriz/cifar.html
[11] http://ufldl.stanford.edu/housenumbers/

the one obtained for full-precision models. These results are consistent with [7] and [13]. Note that in [7] the authors explain the performance of binarized networks with a regularization effect brought by quantization and [13] show that the architecture as well as the size of the data set can have an impact on the performance of quantized networks.

	CIFAR10				SVHN			
Full-precision	0.89				0.96			
Bitwidth	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Full quantization	0.79	0.87	0.88	0.88	0.89	0.95	0.95	0.95
Weight quantization	0.88	0.88	0.88	0.88	0.96	0.95	0.96	0.95

Table 2: Models accuracy on test sets. Full quantization means that both weights and activation values are quantized.

For each data set, we begin by evaluating the robustness of the full-precision and quantized models when the adversary uses three classical white-box gradient-based attacks: FGSM, BIM and CWl2. Then, we evaluate two gradient-free attacks, suitable for black-box settings: ZOO and SPSA. For FGSM, BIM, CWl2 and SPSA we use the Cleverhans library [69], and for ZOO we use the original code provided by the authors[12]. The attacks parameters are detailed in Appendix B. BIM, CWl2, SPSA and ZOO are performed on 1000 randomly samples from the test set.
Over a second phase, we evaluate the transferability of attacks between full-precision and quantized models.

4.3 Evaluation metrics

For each attack, we report two evaluation metrics (see [70] for an extended review of the adversarial robustness evaluation):

- The adversarial accuracy, which is the accuracy of the model on adversarial examples (noted *acc* in the result tables). The crafting method generates an adversarial example x' from each input x of the test set X. Hereafter we note X' the adversarial test set on which is computed the adversarial accuracy. The higher the adversarial accuracy, the less the model is fooled by adversarial examples, i.e. the more the model is robust against the attack.
- The average minimum-distance of the adversarial perturbation, i.e. in our case, the average l_2 norm and l_∞ norm of the difference between clean and adversarial examples which succeed to fool the target model (simply noted l_2 and l_∞ in the result tables). This quantifies the average distortion needed by the attacker to fool the model.

5 Results

5.1 Robustness against gradient-based and gradient-free attacks

Results of direct attacks against fully quantized and weight-only quantized models are presented respectively in tables 3 and 4. For these tables and the following ones dealing with quantized models, first row of the results is for 1-bit model (binarized model), second row for the 2-bit model, third row for the 3-bit model and fourth row for the 4-bit model.

5.1.1 Robustness of binarized neural networks

A first observation from the comparison of table 3 and 4 is that the weight-only quantization has no impact on the robustness. Then, with table 3 we see that fully binarized models are far more robust to FGSM and BIM than their full-precision counterparts, as noted by [27], but achieve only 0.79 and 0.89 accuracy on (respectively) the CIFAR10 and SVHN test datasets (see table 2) which represents a non negligible drop of performance.

[12] https://github.com/huanzhang12/ZOO-Attack

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.12	1.65	0.03	**0.66**	1.65	0.03	0.29	1.66	0.03	**0.78**	1.64	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
				0.17	1.65	0.03				0.37	1.66	0.03
				0.18	1.65	0.03				0.4	1.66	0.03
BIM	0.07	1.17	0.03	**0.66**	1.01	0.03	0.05	1.16	0.03	**0.79**	1.0	0.03
				0.06	1.14	0.03				0.11	1.13	0.03
				0.11	1.17	0.03				0.11	1.13	0.03
				0.06	1.14	0.03				0.1	1.13	0.03
CWl2	0.03	0.58	0.04	**0.11**	0.78	0.08	0.02	0.64	0.06	**0.06**	1.02	0.1
				0.06	0.6	0.04				0.03	0.67	0.07
				0.09	0.55	0.04				0.02	0.66	0.07
				0.05	0.6	0.04				0.02	0.68	0.07
SPSA	0.0	1.37	0.03	**0.16**	1.31	0.03	0.01	1.38	0.03	**0.4**	1.32	0.03
				0.0	1.34	0.03				0.14	1.34	0.03
				0.0	1.36	0.03				0.07	1.35	0.03
				0.0	1.36	0.03				0.04	1.37	0.03
ZOO	0.0	0.72	0.09	0.56	0.1	0.05	0.0	0.91	0.11	0.82	0.07	0.05
				0.83	0.13	0.06				0.93	0.1	0.06
				0.76	0.24	0.07				**0.94**	0.11	0.05
				0.73	1.09	0.14				0.93	0.38	0.1

Table 3: Adversarial accuracy and distortions for gradient-based and gradient-free attacks against full-precision (32-bit) and fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.12	1.65	0.03	0.11	1.65	0.03	0.29	1.66	0.03	0.28	1.66	0.03
				0.18	1.65	0.03				0.38	1.66	0.03
				0.18	1.65	0.03				**0.4**	1.66	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
BIM	0.07	1.17	0.03	0.07	1.19	0.03	0.05	1.16	0.03	0.07	1.16	0.03
				0.08	1.15	0.03				0.1	1.14	0.03
				0.06	1.15	0.03				**0.11**	1.14	0.03
				0.08	1.15	0.03				0.09	1.13	0.03
CWl2	0.03	0.58	0.04	0.05	0.57	0.04	0.02	0.64	0.06	0.02	0.64	0.05
				0.06	0.6	0.04				0.02	0.66	0.06
				0.05	0.61	0.04				0.02	0.67	0.07
				0.06	0.62	0.04				0.02	0.68	0.07
SPSA	0.0	1.37	0.03	0.0	1.38	0.03	0.01	1.38	0.03	0.01	1.38	0.03
				0.0	1.37	0.03				**0.04**	1.37	0.03
				0.0	1.36	0.03				**0.04**	1.37	0.03
				0.0	1.36	0.03				0.03	1.37	0.03
ZOO	0.0	0.72	0.09	0.0	0.75	0.1	0.0	0.91	0.11	0.0	0.92	0.1
				0.0	0.74	0.1				0.0	0.92	0.11
				0.0	0.72	0.09				0.0	0.95	0.11
				0.0	0.73	0.09				0.0	0.93	0.11

Table 4: Adversarial accuracy and distortions for gradient-based and gradient-free attacks against full-precision (32-bit) and weight-only quantized models.

However, CWl2 – one of the most powerful crafting method – is almost as efficient against fully binarized neural networks as against full-precision models. Therefore, fully-binarized neural

networks do not bring much robustness improvement compared to full-precison models against gradient-based attacks, as claimed in [27]. We also note that fully binarized models are relatively more robust to SPSA as well compared to full-precision models. This combined with the slightly poorer performance of CWl2 on binarized models indicates that the loss surface for binarized models is difficult to optimize over.

For full quantization with more than 1 bit, the gradient-based attacks are almost as efficient as against a full-precision model, except for FGSM on SVHN only with a 10% gain of accuracy.

5.1.2 Activation quantization causes gradient masking

Interestingly, we see that ZOO fails very often to produce adversarial examples when attacking fully quantized neural networks. More precisely, we note that when adversarial examples are crafted on a full-precision model, ZOO and CWl2 reach almost the same adversarial accuracy, with slightly higher l_2 distortion for ZOO. However, when adversarial examples are crafted on a model with quantized weights and activations, we note that the adversarial accuracy is higher with ZOO than with CWl2, but that successful adversarial examples crafted with ZOO have much lower l_2 distortion than the ones crafted with CWl2 as well as the one observed for the full-precision model ($l_2 = 0.72$).

We claim that these observations reveal some form of gradient masking caused by the quantization of activation values. Firstly, the almost equal performance of ZOO compared to CWl2 for a full-precision model is expected as gradient-free attacks are supposed to perform worse than their gradient-based counterparts when no gradient masking occurs. Secondly, we argue that the phenomenon observed on fully quantized models is due to gradient masking and the STE technique involved at training time. We explain this by distinguishing two cases:

- ZOO fails to produce successful adversarial examples and CWl2 succeeds: (1) because of the activation quantization, a little change (he_i) may switch the activation value from one quantization bucket to another, inducing a big change in the predicted softmax values, causing the discrete derivative $F^d_{ZOO}(x)$ to explode; (2) on the contrary, this change can also have no impact (keep values in the same bucket), which results in $F_{ZOO}(x-he_i, y) = F_{ZOO}(x+he_i, y)$, causing the discrete derivative $F^d_{ZOO}(x)$ to be null. To sum up, F_{ZOO} presents some sharp curvatures or flatness around some points, caused by activation quantization, which prevents ZOO to build successful adversarial examples. The CWl2 attack avoids this problem as it computes gradients thanks to a STE, even if the gradient computed may not be exactly the same as the true gradient [34].

- Both ZOO and CWl2 succeed to produce successful adversarial examples: the l_2 distortion for the successful adversarial examples produced by ZOO is smaller than the one produced by CWl2. Around these points the surface of the objective function to optimize does not present any sharp curvature or flatness. Both ZOO and CWl2 do not suffer from the local minima problem, but as noted by [34], the gradients computed by the CWl2 attack is not representative of the true gradient. The gradient being better estimated by the ZOO attack, it explains its success (lower l_2 distortion).

The gradient masking phenomenon hypothesis concerning the quantization of activation values is also verified with the fact that the gradient-free attack SPSA performs quite better in terms of adversarial accuracy than BIM, against fully binarized models. We also make the hypothesis that SPSA avoids the sharp curvatures or flatness observed around some points, where ZOO fails to produce adversarial examples, because of the more efficient gradient estimation method suited to noisy objective functions [29].

For the weight quantized models results presented in Table 4, we do not observe the same phenomena as for the full quantization models. No apparent robustness is noticeable for the weight quantized models. No sharp variation or flatness is induced for the objective function of ZOO of the weight quantized models, as it originated from the quantization of activation values. It has to be noted that we also measured that the variance of the logits values between full-precision and weight-only quantized models is almost the same.

5.2 Transferability

We present results of transferability when the source network (i.e. the models the adversarial examples are crafted from) is full-precision, fully or weight-only quantized models, and the adversarial examples are transferred to (target models) full-precision, fully or weight-only quantized networks, in figures 1 and 2. For CWl2, as advised in [18], we consider $\kappa > 0$ to build strong adversarial examples on the source model, more likely to transfer. We tested $\kappa = 5, 10, 15, 20, 25$. For each source model, we report the best transferability results. The results for the cases where the source networks are 2-bit or 4-bit quantized models are not presented here for paper length purpose and as these results can be interpolated from the one we present. More complete tables can be found in Appendix C.

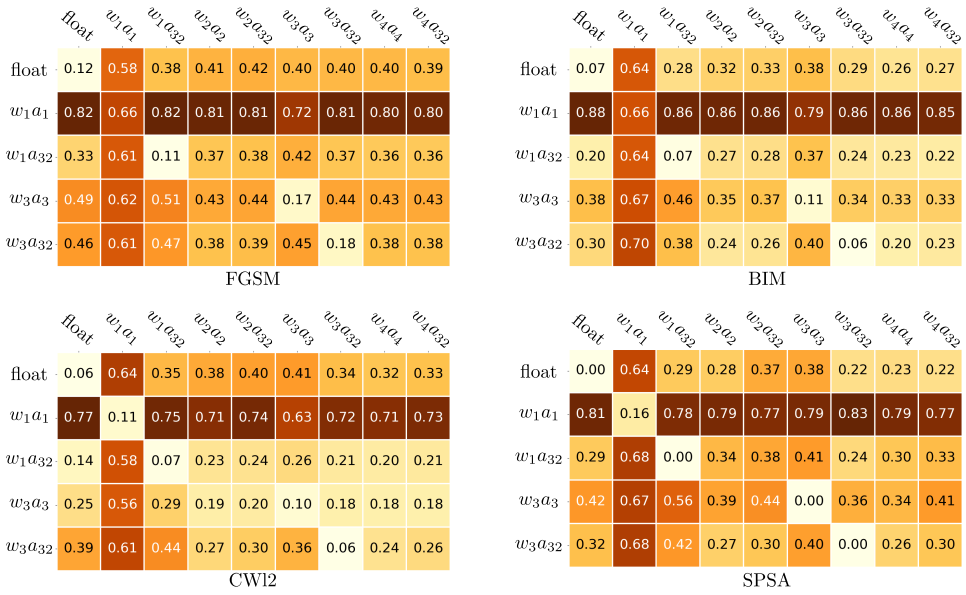

Figure 1: Adversarial transferability results for CIFAR10. Rows are relative to source networks and columns to target networks. Values correspond to adversarial accuracy. The lower the value, the more transferability occurs.

5.2.1 Weak transferability

A first observation is that transferability results are quite poor for FGSM, BIM and SPSA. CWl2, given tuning the parameter κ, suffers less from transferability issues, at the cost of increased l_2 and l_∞ distortion, except when the source or target network is a fully binarized network. Indeed, for the κ values tested (for fully binarized models, the results reported are for $\kappa = 5$), when the source network is a fully binarized model, CWl2 struggles to find adversarial examples having both $F(x + \epsilon, y) < 0$ (see Equation 7) and a little l_2 distortion. This results in adversarial examples being missclassified but not imperceptible by a human. We hypothesize this comes from the hard to optimize loss function as noted in 5.1.1. We also note that, as already noticed by [71], and contrary to what was initially found by [46], that BIM – as it is the case here – may produce more transferable adversarial examples than FGSM .

5.2.2 Quantization shift phenomenon

These poor transferability results (mainly for FGSM and BIM) can be explained by the *quantization value shift* phenomenon which takes places when quantization ruins the adversarial effect by mapping two different values to the same quantization bucket. In case of activation quantization, two activation values can be mapped to the same value. In case of weight quantization, this leveling effect may also be observed and ruins the adversarial effect.

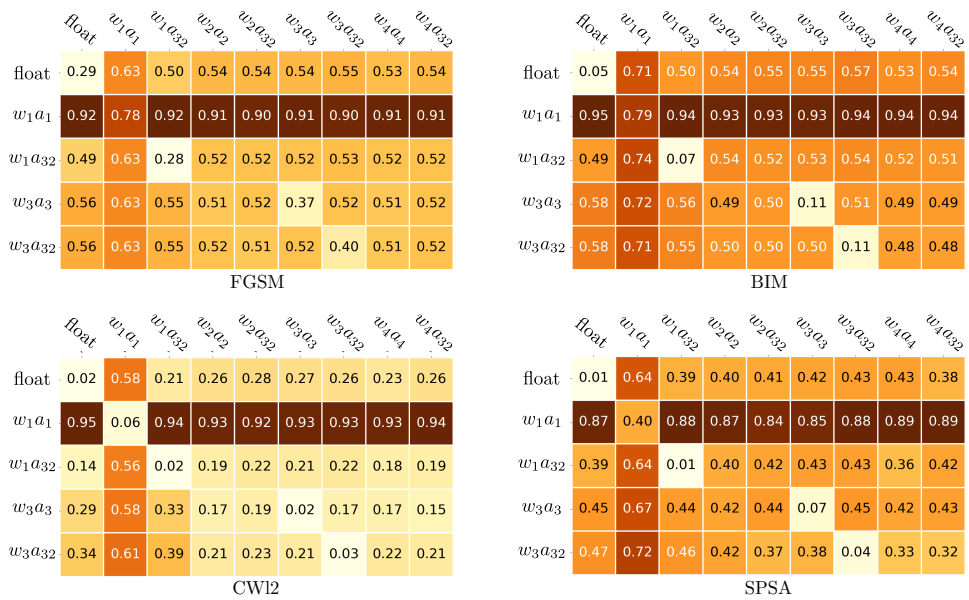

Figure 2: Adversarial transferability results for SVHN. Rows are relative to source network and columns to target networks. Values correspond to adversarial accuracy. The lower the value, the more transferability occurs.

Consequently, whatever the quantization level of the source model adversarial examples are crafted on, evaluating them on a target model with a different quantization level may hinder their efficiency because of this phenomenon.

5.2.3 Gradient misalignment

Regarding the transferability results, we may also hypothesize that the gradient direction between float models and quantized models and between models with different bitwidths is quite different. This gradient misalignment may be noticeable for the gradient computed with the *Straight Through Estimator*, as poor transferability is observed for the white-box attacks (FGSM, BIM, CWl2), and for the real gradient, as poor transferability is observed for SPSA. We measure the mean cosine similarity between the gradient of the loss function with respect to the input between models with different bitwidth and show the results in Figure 3 for CIFAR10. We remind that the cosine similarity $CS(a, b) \in \mathbb{R}$ between two vectors $a, b \in \mathbb{R}^m \times \mathbb{R}^m$ is defined as:

$$CS(a, b) = \frac{\langle a, b \rangle}{\|a\|_2 \cdot \|b\|_2}$$

where $\langle \cdot, \cdot \rangle$ is the usual scalar product. $CS(a, b) = 0$ indicates orthogonal vectors for the usual scalar product, $CS(a, b) = 1$ indicates aligned vectors in the same direction and $CS(a, b) = -1$ indicates aligned vectors in opposite directions.

In Figure 3, we first observe that the cosine similarity values for gradients of the loss function between full-precision and quantized models and between quantized models with different bitwidths, are relatively close to 0, indicating nearly orthogonal gradient directions. We observe that the cosine similarity for the gradient of the loss function with respect to the input between fully-binarized models and others models is the closest to 0. This is in accordance for example with results presented in Fig. 1 where transferability capacities for FGSM, BIM, CWl2 and SPSA are the poorest when fully-binarized models are involved. Moreover, this may explain the fact that adversarial accuracy is much higher in tables 19 and 20 (Appendix C), where adversarial examples are crafted on fully-binarized models, than in the other tables.

To conclude, transferability results show that quantization strongly alters chances of success for an adversary who has only access to a full-precision (or quantized) version of a model and wants

to attack a quantized (respectively, a full-precision) version of a model, assuming this adversary can not use a black-box attack such as the SPSA one.

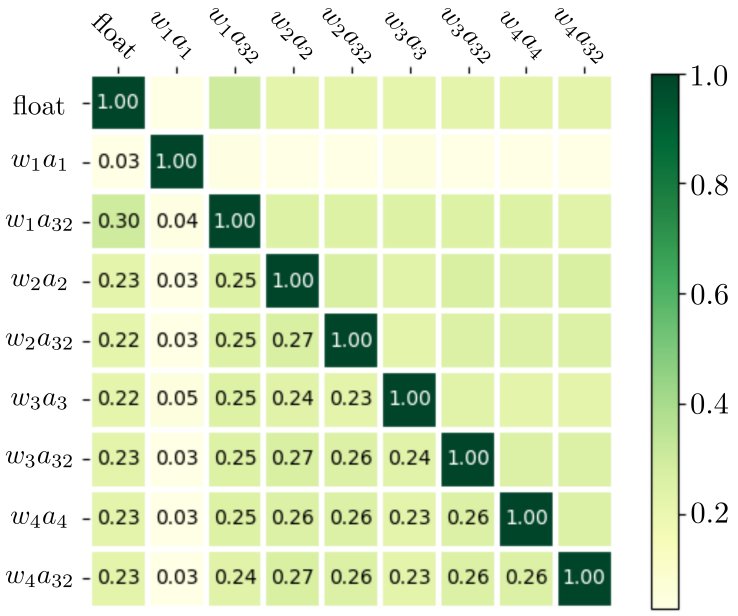

Figure 3: Cosine similarity values between the gradient of the loss function with respect to the input, for full-precision and quantized models. w_ia_j designates a model with a i-bit weight quantization and a j-bit activation quantization.

6 Ensemble of quantized models

6.1 Motivation

Regarding of the transferability results, a logical consequence and a natural assumption is to consider an ensemble of quantized models to filter out adversarial examples. In this section, we analyze the relevance of this defense strategy. We remind the reader about the important point we highlight in section 3.3.2 about the threat models and their intrinsic limitations.

We consider an ensemble of quantized models, $\mathcal{M} = \{M_i\}$, in our case: a full-precision model and four fully-quantized models (1,2,3 and 4 bits). We first analyze statistically how the models agree on clean and adversarial test set using different crafting methods (FGSM, BIM, CWl2 and SPSA). More precisely, we consider an adversarial example crafted on a source model M_s and we look at how the five models agree, given that this adversarial example is successful or not on M_s. Similarly, we look at how the five models agree on clean test set examples, given that this test set example is correctly classified in the true class or, on the contrary, misclassified by M_s. Our main observations are the following:

1. the models are more likely to agree on clean samples than on adversarial examples (successful or not);

2. the models are much more likely to agree on unsuccessful (on M_s) adversarial examples.

In Table 5, we show for CIFAR10 and SVHN, given the source model M_s, how the four other models agree on test set and adversarial examples crafted with FGSM. For example, for CIFAR10, with the 2-bit model, 77% of test set examples well-recognized by M_s are also correctly classified by the other four models, compared to 33% for misclassified test set examples. Moreover, 37% of successful adversarial examples (i.e. examples that effectively fool M_s) also fool the other four models, and 76% of unsuccessful adversarial examples (on M_s) are also unsuccessful on the other four models.

	CIFAR10					SVHN				
	Source model									
	$float$	w_1a_1	w_2a_2	w_3a_3	w_4a_4	$float$	w_1a_1	w_2a_2	w_3a_3	w_4a_4
	Test set examples									
Correctly classified	0.75	0.85	0.77	0.82	0.76	0.9	0.97	0.91	0.91	0.9
Misclassified	0.39	0.19	0.33	0.33	0.34	0.46	0.16	0.38	0.37	0.4
	Adversarial examples (FGSM)									
Successful	0.31	0.09	0.37	0.31	0.33	0.33	0.09	0.41	0.43	0.39
Unsuccessful	0.88	0.80	0.76	0.78	0.79	0.89	0.94	0.81	0.82	0.84

Table 5: Rate of examples for which the other four models agree, depending on the source model M_s and its prediction results (*correctly classified*, *misclassified*, *successful*, *unsucessful*).

6.2 Ensemble-based defense

We design the following ensemble-based defense method: the prediction for an upcoming example is done only if m or more models agree, and the final label is the one predicted by these m models. We note $valid_{m,\mathcal{M}}(X)$ the set of samples from the input data set X which respect this criterion:

$$valid_{m,\mathcal{M}}(X) = \left\{ x \in X \;\middle|\; \max_{j \in [1,\ldots,C]} \sum_i \mathbf{1}_{M_i(x)=j} \geq m \right\} \tag{12}$$

with C the number of labels and $M_i(x)$ the output prediction label of x by M_i. Then, the *prediction rate* (hereafter, PR) quantifying the proportion of examples from X for which prediction is performed is :

$$PR_{m,\mathcal{M}}(X) = \frac{|valid_{m,\mathcal{M}}(X)|}{|X|} \tag{13}$$

Where $|X|$ is the cardinality of X. Considering the remarks made above, this approach encourages prediction for clean examples rather than for adversarial examples, and when the prediction is performed on adversarial examples, it would be predominantly unsuccessful ones.

When evaluating this defense on the adversarial test set (X'), an overall performance metric, hereafter called *defense accuracy* (d_acc) is simply defined as the proportion of adversarial examples which have been filtered out or which are unsuccessful. Practically, the defense accuracy could also be defined as:

$$d\_\mathrm{acc}_{m,\mathcal{M}}(X') = 1 - \frac{|valid_{m,\mathcal{M}}(X')^s|}{|X'|} \tag{14}$$

Where $valid_{m,\mathcal{M}}(X')^s$ denotes the set of successful adversarial examples which thwart the filtering process (i.e. the *error rate* of the defense).

The number m has to be decided following a trade-off between the number of test set examples for which prediction is performed (which has to remain high) and the defense error rates (which has to be low) when facing adversarial examples. We experimentally set $m = 4$ for CIFAR10 and $m = 5$ for SVHN to reach a good trade-off. As presented in Table 6, the prediction is performed for more than 87% of the clean test set examples for both CIFAR10 and SVHN, with an accuracy of 90% and 98% respectively for CIFAR10 and SVHN on clean test set examples for which prediction is performed.

	CIFAR10		SVHN	
	PR	accuracy	PR	accuracy
Test set	0.87	0.90	0.87	0.98

Table 6: Prediction rate and accuracy with the ensemble of models on CIFAR10 and SVHN.

6.3 Results

We present the results of this defense on CIFAR10 and SVHN in Table 7. For each source model and each attack method, we report the defense accuracy *d_acc* against four attacks (FGSM, BIM, CWl2 and SPSA), along with the prediction rate PR.

	CIFAR10				SVHN			
	Float model *(32-bit)*		Quantized models *(1,2,3,4-bit)*		Float model *(32-bit)*		Quantized models *(1,2,3,4-bit)*	
	PR	d_acc	PR	d_acc	PR	d_acc	PR	d_acc
FGSM	0.58	0.63	0.73	**0.9**	0.39	0.65	0.80	**0.98**
			0.58	0.53			0.47	0.86
			0.45	0.63			0.47	0.84
			0.53	0.57			0.46	0.87
BIM	0.65	0.44	0.71	**0.88**	0.20	0.85	0.80	**0.99**
			0.62	0.38			0.28	0.81
			0.57	0.44			0.26	0.81
			0.52	0.48			0.25	0.82
CWl2	0.54	0.60	0.17	**0.84**	0.15	0.84	0.18	**0.84**
			0.47	0.53			0.23	0.77
			0.58	0.42			0.2	0.8
			0.36	0.64			0.17	0.83
SPSA	0.68	0.41	0.32	**0.82**	0.22	0.8	0.5	**0.97**
			0.48	0.54			0.32	0.82
			0.44	0.57			0.29	0.79
			0.58	0.42			0.31	0.75

Table 7: Defense accuracy (d_acc) and adversarial prediction rate (PR) for gradient-based and gradient-free attacks against an ensemble of quantized models, depending of the source model.

Considering the natural threat model discussed in 3.3, results are interesting for adversarial examples crafted from the full precision model especially for SVHN. If an attacker tries to directly transfer the adversarial examples crafted from the full precision model with, for example, the CWl2 attack, 60% of the adversarial examples are harmless (i.e. filtered out or unsuccessful) for CIFAR10 and this robustness is even stronger for SVHN with a defense accuracy superior to 80% for BIM, CWl2 or SPSA.

Moreover, coherently with the transferability results (see Figures 1, 2 and additional tables in Appendix C), the highest robustness is reached when adversarial examples are crafted from a fully binarized network with a defense accuracy superior to 0.8 – whatever the crafting method – particularly for SVHN. However, there is no significant gain compare to the transferability results obtained by taking each quantized model separately (see Figures 1 and 2). But, except for this case of the fully binarized network, the ensemble of quantized models shows better robustness to transferred adversarial examples than all single models. The more relevant gain is reached with CWl2 attack with a mean (over the 2, 3 and 4-bit networks) defense accuracy of 0.53 and 0.8 respectively for CIFAR10 and SVHN.

Once again, if we do not claim to meet state-of-art detection based protection (such as [55], [72], [56] and [53]), we regard these results as significant ones, particularly since we are deeply convinced that an efficient defense strategy against adversarial examples will necessary be a composition of several protection schemes as it the case in other security domains such as efficient countermeasures against physical attacks for cryptographic systems which combine masking, hiding and redundancy principles.

7 Conclusion

In this article, we show experimentally on CIFAR10 and SVHN and state-of-the-art gradient-based and gradient-free attacks that quantization in itself offers very poor protection against adversarial

examples crafted by adversaries having access to the model or able to query it. We find that activation quantization can lead to gradient masking. We verify experimentally that the efficiency of some gradient-based and gradient-free attacks can thus be tampered but other gradient-based or gradient-free attacks do not suffer from gradient masking, because of the usage of a STE to approximate gradients, or the optimization procedure begin well-suited for noisy functions. Eventually, we demonstrate poor transferability capacities between classical models and quantized models, and between quantized models with different bitwidths. We explain this by the *quantization shift phenomenon* which ruins adversarial effects and gradients misalignment.

As an exploratory work and logical consequence of the transferability results, we analyze the impact of considering an ensemble of quantized models in order to filter out adversarial examples with a minimum impact on the natural accuracy. Such an ensemble method, like any other detection-based approach, suffers from a narrow threat model since the defense is useless with an attacker aware of the implementation details of the model in the target device [73]. However, for black-box paradigms, the use of quantized ensemble may have an interesting impact on the transferability when associated to other and complementary defense mechanisms.

We believe that the characteristics of embedded models particularly induced by quantization approaches (weights or activation outputs) have to be taken into consideration in order to design suitable and efficient protection schemes. These defense strategies for embedded models will be the purpose of future works, since robustness requirements will obviously become more and more compulsory as critical tasks (as well as processed data) will be performed thanks to a growing variety of devices.

References

[1] M. Denil, B. Shakibi, L. Dinh, N. De Freitas, *et al.*, "Predicting parameters in deep learning," in *Advances in neural information processing systems*, pp. 2148–2156, 2013.

[2] G. B. Hacene, V. Gripon, M. Arzel, N. Farrugia, and Y. Bengio, "Quantized guided pruning for efficient hardware implementations of convolutional neural networks," *arXiv preprint arXiv:1812.11337*, 2018.

[3] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[5] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," *arXiv preprint arXiv:1612.01543*, 2016.

[6] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, pp. 3123–3131, 2015.

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations.," *Journal of Machine Learning Research*, vol. 18, no. 187, pp. 1–30, 2017.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, pp. 525–542, Springer, 2016.

[10] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," 2016.

[11] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[12] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International Conference on Machine Learning*, pp. 1737–1746, 2015.

[13] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[14] R. Ding, Z. Liu, R. Shi, D. Marculescu, and R. Blanton, "Lightnn: Filling the gap between conventional deep neural networks and binarized networks," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pp. 35–40, ACM, 2017.

[15] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[16] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2013.

[17] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, IEEE, 2017.

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

[20] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[21] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, ACM, 2017.

[22] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, ACM, 2017.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[24] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *arXiv preprint arXiv:1803.01442*, 2018.

[25] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[26] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[27] A. Galloway, G. W. Taylor, and M. Moussa, "Attacking binarized neural networks," in *International Conference on Learning Representations*, 2018.

[28] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2018.

[29] J. Uesato, B. O'Donoghue, A. v. d. Oord, and P. Kohli, "Adversarial risk and the dangers of evaluating against weak attacks," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2018.

[30] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," in *International Conference on Learning Representations*, 2019.

[31] Y. Zhao, I. Shumailov, R. Mullins, and R. Anderson, "To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression," 2018.

[32] A. S. Rakin, J. Yi, B. Gong, and D. Fan, "Defend deep neural networks against adversarial examples via fixed anddynamic quantized activation functions," *arXiv preprint arXiv:1807.06714*, 2018.

[33] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[34] E. B. Khalil, A. Gupta, and B. Dilkina, "Combinatorial attacks on binarized neural networks," *arXiv preprint arXiv:1810.03538*, 2018.

[35] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[36] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "Bnn+: Improved binary network training," 2018.

[37] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, ACM, 2015.

[38] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 3–18, IEEE, 2017.

[39] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, ACM, 2015.

[40] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, ACM, 2017.

[41] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," *arXiv preprint arXiv:1703.01340*, 2017.

[42] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.

[43] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow, "Adversarial spheres," *arXiv preprint arXiv:1801.02774*, 2018.

[44] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[45] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *arXiv preprint arXiv:1905.02175*, 2019.

[46] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations*, 2016.

[47] J. C. Spall *et al.*, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE transactions on automatic control*, vol. 37, no. 3, pp. 332–341, 1992.

[48] A. C. Serban and E. Poll, "Adversarial examples-a complete characterisation of the phenomenon," *arXiv preprint arXiv:1810.01185*, 2018.

[49] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018.

[50] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[51] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[52] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[53] Z. Zheng and P. Hong, "Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 7924–7933, 2018.

[54] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[55] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147, ACM, 2017.

[56] J. Lu, T. Issaranon, and D. A. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly.," in *ICCV*, pp. 446–454, 2017.

[57] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[58] T.-J. Chang, Y. He, and P. Li, "Efficient two-step adversarial defense for deep neural networks," *arXiv preprint arXiv:1810.03739*, 2018.

[59] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018.

[60] T. Zheng, C. Chen, and K. Ren, "Is pgd-adversarial training necessary? alternative training via a soft-quantization network with noisy-natural samples only," *arXiv preprint arXiv:1810.05665*, 2018.

[61] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," *arXiv preprint arXiv:1901.08573*, 2019.

[62] S. Kariyappa and M. K. Qureshi, "Improving adversarial robustness of ensembles with diversity training," *arXiv preprint arXiv:1901.09981*, 2019.

[63] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[64] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, vol. 1, no. 2, p. 3, 2017.

[65] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems*, pp. 2266–2276, 2017.

[66] J. Peck, J. Roels, B. Goossens, and Y. Saeys, "Lower bounds on the robustness to adversarial perturbations," in *Advances in Neural Information Processing Systems*, pp. 804–813, 2017.

[67] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[69] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[70] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, and A. Madry, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[71] L. Wu, Z. Zhu, C. Tai, *et al.*, "Understanding and enhancing the transferability of adversarial examples," *arXiv preprint arXiv:1802.09707*, 2018.

[72] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: Adversarial training using generative models," *arXiv preprint arXiv:1712.09196*, 2017.

[73] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14, ACM, 2017.

A Networks architecture

Layer type	CIFAR10	SVHN
Convolution + BatchNorm + relu	(128,3,3)	(128,3,3)
Convolution + MaxPooling + BatchNorm + relu	(128,3,3), (2,2)	(128,3,3), (2,2)
Convolution + BatchNorm + relu	(256,3,3)	(256,3,3)
Convolution + MaxPooling + BatchNorm + relu	(256,3,3), (2,2)	(128,3,3), (2,2)
Convolution + BatchNorm + relu	(512,3,3)	(512,3,3)
Convolution + MaxPooling + BatchNorm + relu	(512,3,3), (2,2)	(512,3,3), (2,2)
Fully Connected + BatchNorm + relu	1024, (2,2)	1024, (2,2)
Fully Connected + BatchNorm + relu	1024, (2,2)	1024, (2,2)
Fully Connected + softmax	10	10

Table 8: Full-precision models architecture

Layer type	CIFAR10	SVHN
ConvolutionQuant + BatchNorm + reluQuant	(128,3,3)	(128,3,3)
ConvolutionreluQuant + MaxPooling + BatchNorm + reluQuant	(128,3,3), (2,2)	(128,3,3), (2,2)
ConvolutionQuant + BatchNorm + reluQuant	(256,3,3)	(256,3,3)
ConvolutionQuant + MaxPooling + BatchNorm + reluQuant	(256,3,3), (2,2)	(128,3,3), (2,2)
ConvolutionQuant + BatchNorm + reluQuant	(512,3,3)	(512,3,3)
ConvolutionQuant + MaxPooling + BatchNorm + reluQuant	(512,3,3), (2,2)	(512,3,3), (2,2)
DenseQuant + BatchNorm + reluQuant	1024, (2,2)	1024, (2,2)
DenseQuant + BatchNorm + reluQuant	1024, (2,2)	1024, (2,2)
Dense + softmax	10	10

Table 9: Fully Quantized models architecture. ConvolutionQuant, DenseQuant and reluQuant designate respectively a convolution layer with quantized weights, a dense layer with quantized weights and the relu activation function with its output quantized

Layer type	CIFAR10	SVHN
ConvolutionQuant + BatchNorm + reluQuant	(128,3,3)	(128,3,3)
ConvolutionreluQuant + MaxPooling + BatchNorm + relu	(128,3,3), (2,2)	(128,3,3), (2,2)
ConvolutionQuant + BatchNorm + reluQuant	(256,3,3)	(256,3,3)
ConvolutionQuant + MaxPooling + BatchNorm + relu	(256,3,3), (2,2)	(128,3,3), (2,2)
ConvolutionQuant + BatchNorm + reluQuant	(512,3,3)	(512,3,3)
ConvolutionQuant + MaxPooling + BatchNorm + relu	(512,3,3), (2,2)	(512,3,3), (2,2)
DenseQuant + BatchNorm + relu	1024, (2,2)	1024, (2,2)
DenseQuant + BatchNorm + relu	1024, (2,2)	1024, (2,2)
Dense + softmax	10	10

Table 10: Weight Quantized models architecture. ConvolutionQuant and DenseQuant designate respectively a convolution layer with quantized weights and a dense layer with quantized weights

B Attacks parameters

For ZOO and CWl2, we noticed that results between 100 and 1000 iterations were almost similar, the adversarial accuracy almost never decreased and the l_2 distortion for the two attacks decreased proportionally. For computation time issues we then chose to perform the attack with 100 iterations, as this does not change any interpretations of our results.

The value of κ for the CWl2 attack is set to 0 when considering an adversary in the white-box setting (see Section 5.1). Otherwise, in particular for transfer-based attacks, this parameter is tuned (see Section 5.2 for details).

ϵ	0.03

Table 11: Hyperparameters for FGSM

ϵ	0.03
Iterations	100
Step size	0.0003

Table 12: Hyperparameters for BIM

ϵ	0.03
Iterations	100
Learning rate	0.01
Perturbation size δ	0.01
Batch size	128

Table 13: Hyperparameters for SPSA

Iterations	100
Learning rate	0.1
Initial constant	0.9
Search steps	10
$\boldsymbol{\kappa}$	0

Table 14: Hyperparameters for CWl2

Iterations	100
Learning rate	0.1
Initial constant	0.9
Search steps	10
$\boldsymbol{\kappa}$	0

Table 15: Hyperparameters for ZOO

C Complete transferability results

In the following tables, "–" denotes a value which can not be computed. For example, the l_2 distortion of successful adversarial examples for an attack can not be computed when the adversarial accuracy of the target model against this attack equals 1.

We summarize the reference of the transferability tables where $w_i a_j$ designates a model with a i-bit quantization of the weights and a j-bit quantization of the activation values.

From \ To	full	w_1a_1	w_2a_2	w_3a_3	w_4a_4	w_1a_{32}	w_2a_{32}	w_3a_{32}	w_4a_{32}
full	Table 17					Table 18			
w_1a_1	Table 19					Table 20			
w_1a_{32}	Table 21					Table 22			
w_2a_2	Table 23					Table 24			
w_2a_{32}	Table 25					Table 26			
w_3a_3	Table 27					Table 28			
w_3a_{32}	Table 29					Table 30			
w_4a_4	Table 31					Table 32			
w_4a_{32}	Table 33					Table 34			

Table 16: Summary of the references for the transferability results between full precision, fully quantized and weight only models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.12	1.65	0.03	0.58	1.65	0.03	0.29	1.66	0.03	0.63	1.66	0.03
				0.41	1.65	0.03				0.54	1.66	0.03
				0.4	1.65	0.03				0.54	1.66	0.03
				0.4	1.65	0.03				0.53	1.66	0.03
BIM	0.07	1.17	0.03	0.64	1.18	0.03	0.05	1.16	0.03	0.71	1.16	0.03
				0.32	1.17	0.03				0.54	1.16	0.03
				0.38	1.18	0.03				0.55	1.16	0.03
				0.26	1.17	0.03				0.53	1.16	0.03
CWl2	0.03	0.58	0.04	0.64	0.80	0.07	0.02	0.64	0.06	0.59	0.88	0.1
				0.38	0.82	0.07				0.26	0.91	0.11
				0.41	0.82	0.07				0.27	0.91	0.11
				0.32	0.82	0.07				0.23	0.92	0.11
SPSA	0.0	1.37	0.03	0.64	1.37	0.03	0.01	1.38	0.03	0.64	1.38	0.03
				0.28	1.37	0.03				0.4	1.38	0.03
				0.38	1.37	0.03				0.42	1.38	0.03
				0.23	1.37	0.03				0.43	1.38	0.03
ZOO	0.0	0.72	0.09	0.77	0.56	0.08	0.0	0.91	0.11	0.86	0.68	0.1
				0.84	0.55	0.09				0.91	0.63	0.1
				0.76	0.63	0.09				0.91	0.65	0.1
				0.83	0.6	0.09				0.92	0.67	0.1

Table 17: Transferability from full-precision model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.12	1.65	0.03	0.38	1.65	0.03	0.29	1.66	0.03	0.5	1.66	0.03
				0.42	1.65	0.03				0.54	1.66	0.03
				0.4	1.65	0.03				0.55	1.66	0.03
				0.39	1.65	0.03				0.54	1.66	0.03
BIM	0.07	1.17	0.03	0.28	1.17	0.03	0.05	1.16	0.03	0.5	1.16	0.03
				0.33	1.17	0.03				0.55	1.16	0.03
				0.29	1.18	0.03				0.57	1.15	0.03
				0.27	1.17	0.03				0.54	1.16	0.03
CWl2	0.03	0.58	0.04	0.35	0.82	0.07	0.02	0.64	0.06	0.21	0.92	0.11
				0.4	0.82	0.07				0.28	0.91	0.11
				0.34	0.82	0.07				0.26	0.91	0.11
				0.32	0.82	0.08				0.26	0.91	0.11
SPSA	0.0	1.37	0.03	0.29	1.37	0.03	0.01	1.38	0.03	0.39	1.38	0.03
				0.37	1.37	0.03				0.41	1.38	0.03
				0.22	1.37	0.03				0.43	1.38	0.03
				0.22	1.37	0.03				0.38	1.38	0.03
ZOO	0.0	0.72	0.09	0.84	0.56	0.08	0.0	0.91	0.11	0.93	0.51	0.08
				0.83	0.56	0.08				0.9	0.62	0.09
				0.84	0.61	0.09				0.92	0.69	0.1
				0.83	0.58	0.09				0.92	0.64	0.1

Table 18: Transferability from full-precision model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.82	1.65	0.03	0.66	1.65	0.03	0.92	1.64	0.03	0.78	1.64	0.03
				0.81	1.65	0.03				0.91	1.64	0.03
				0.72	1.65	0.03				0.91	1.64	0.03
				0.8	1.65	0.03				0.91	1.64	0.03
BIM	0.88	1.01	0.03	0.66	1.01	0.03	0.95	1.01	0.03	0.79	1.0	0.03
				0.86	1.01	0.03				0.93	1.0	0.03
				0.79	1.02	0.03				0.93	1.0	0.03
				0.86	1.01	0.03				0.94	1.0	0.03
CWl2	0.77	1.95	0.23	0.11	0.78	0.08	0.95	1.28	0.13	0.06	1.02	0.1
				0.71	2.22	0.21				0.93	0.94	0.09
				0.63	2.22	0.21				0.93	0.79	0.08
				0.71	2.3	0.22				0.93	1.29	0.11
SPSA	0.81	1.31	0.03	0.16	1.31	0.03	0.87	1.32	0.03	0.4	1.32	0.03
				0.79	1.32	0.03				0.87	1.32	0.03
				0.76	1.31	0.03				0.85	1.32	0.03
				0.79	1.32	0.03				0.89	1.33	0.03
ZOO	1.00	–	–	0.56	0.1	0.05	1.00	–	–	0.82	0.07	0.05
				0.88	0.1	0.06				0.95	0.09	0.08
				0.82	0.12	0.06				0.94	0.06	0.04
				0.88	0.24	0.1				1.00	–	–

Table 19: Transferability from fully binarized model to 1,2,3,4-bit fully quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.82	1.65	0.03	0.82	1.65	0.03	0.92	1.64	0.03	0.92	1.64	0.03
				0.81	1.65	0.03				0.9	1.64	0.03
				0.81	1.65	0.03				0.91	1.64	0.03
				0.8	1.65	0.03				0.91	1.64	0.03
BIM	0.88	1.01	0.03	0.88	1.01	0.03	0.95	1.01	0.03	0.94	1.0	0.03
				0.86	1.01	0.03				0.93	1.01	0.03
				0.86	1.01	0.03				0.94	1.01	0.03
				0.85	1.02	0.03				0.94	1.0	0.03
CWl2	0.77	1.95	0.23	0.874	2.21	0.23	0.95	1.28	0.13	0.94	1.07	0.1
				0.74	2.22	0.22				0.92	1.02	0.1
				0.72	2.36	0.22				0.93	1.23	0.12
				0.73	2.26	0.22				0.94	1.19	0.12
SPSA	0.81	1.31	0.03	0.78	1.32	0.03	0.87	1.32	0.03	0.88	1.32	0.03
				0.77	1.32	0.03				0.84	1.32	0.03
				0.83	1.32	0.03				0.88	1.32	0.03
				0.77	1.32	0.03				0.89	1.32	0.03
ZOO	1.00	–	–	0.89	–	–	1.00	–	–	1.00	–	–
				1.00	–	–				1.00	–	–
				1.00	–	–				1.00	–	–
				0.88	0.15	0.08				1.00	–	–

Table 20: Transferability from fully binarized model to 1,2,3,4-bit weight-only quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.33	1.65	0.03	0.61	1.65	0.03	0.49	1.66	0.03	0.63	1.66	0.03
				0.37	1.65	0.03				0.52	1.66	0.03
				0.42	1.65	0.03				0.52	1.66	0.03
				0.36	1.65	0.03				0.52	1.66	0.03
BIM	0.2	1.19	0.03	0.64	1.19	0.03	0.49	1.16	0.03	0.74	1.16	0.03
				0.27	1.19	0.03				0.54	1.16	0.03
				0.37	1.19	0.03				0.53	1.16	0.03
				0.23	1.19	0.03				0.52	1.16	0.03
CWl2	0.14	0.84	0.09	0.58	0.83	0.08	0.14	0.97	0.10	0.56	0.93	0.09
				0.23	0.84	0.08				0.19	0.96	0.1
				0.26	0.84	0.08				0.21	0.95	0.09
				0.20	0.84	0.08				0.19	0.95	0.09
SPSA	0.29	1.38	0.03	0.68	1.38	0.03	0.39	1.39	0.03	0.64	1.39	0.03
				0.34	1.38	0.03				0.4	1.39	0.03
				0.41	1.38	0.03				0.43	1.39	0.03
				0.3	1.38	0.03				0.36	1.39	0.03
ZOO	0.93	0.7	0.1	0.88	0.59	0.09	0.97	0.73	0.1	0.9	0.68	0.1
				0.93	0.63	0.09				0.95	0.65	0.09
				0.9	0.71	0.1				0.96	0.65	0.1
				0.94	0.65	0.09				0.96	0.64	0.09

Table 21: Transferability from weight-only binarized model to 1,2,3,4-bit fully quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.33	1.65	0.03	0.11	1.65	0.03	0.49	1.66	0.03	0.28	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
				0.37	1.65	0.03				0.53	1.66	0.03
				0.36	1.65	0.03				0.52	1.66	0.03
BIM	0.2	1.19	0.03	0.07	1.19	0.03	0.49	1.16	0.03	0.07	1.16	0.03
				0.28	1.19	0.03				0.52	1.16	0.03
				0.24	1.19	0.03				0.54	1.16	0.03
				0.22	1.19	0.03				0.51	1.16	0.03
CWl2	0.14	0.84	0.09	0.07	0.84	0.08	0.14	0.97	0.10	0.02	1.00	0.10
				0.24	0.84	0.08				0.22	0.95	0.10
				0.21	0.84	0.09				0.22	0.95	0.10
				0.21	0.84	0.08				0.19	0.95	0.10
SPSA	0.29	1.38	0.03	0.00	1.38	0.03	0.39	1.39	0.03	0.01	1.38	0.03
				0.38	1.38	0.03				0.42	1.39	0.03
				0.24	1.38	0.03				0.43	1.39	0.03
				0.33	1.38	0.03				0.42	1.39	0.03
ZOO	0.93	0.7	0.1	0.00	0.75	0.1	0.97	0.73	0.1	0.0	0.92	0.1
				0.94	0.65	0.1				0.96	0.63	0.1
				0.94	0.62	0.09				0.97	0.7	0.1
				0.93	0.63	0.09				0.97	0.7	0.1

Table 22: Transferability from weight-only binarized model to 1,2,3,4-bit weight-only quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.45	1.65	0.03	0.6	1.65	0.03	0.56	1.66	0.03	0.62	1.66	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
				0.44	1.65	0.03				0.52	1.66	0.03
				0.37	1.65	0.03				0.51	1.66	0.03
BIM	0.3	1.14	0.03	0.68	1.14	0.03	0.55	1.12	0.03	0.73	1.13	0.03
				0.06	1.14	0.03				0.11	1.13	0.03
				0.39	1.14	0.03				0.50	1.13	0.03
				0.20	1.14	0.03				0.49	1.13	0.03
CWl2	0.29	0.93	0.08	0.54	0.90	0.08	0.36	0.65	0.09	0.66	0.61	0.08
				0.06	0.6	0.04				0.03	0.67	0.07
				0.25	0.94	0.08				0.21	0.69	0.09
				0.15	0.92	0.08				0.18	0.70	0.09
SPSA	0.57	1.34	0.03	0.72	1.33	0.03	0.55	1.34	0.03	0.68	1.33	0.03
				0.0	1.34	0.03				0.14	1.34	0.03
				0.57	1.34	0.03				0.55	1.34	0.03
				0.39	1.34	0.03				0.56	1.34	0.03
ZOO	1.0	–	–	0.98	0.13	0.06	1.0	–	–	0.99	0.12	0.06
				0.83	0.13	0.06				0.93	0.1	0.06
				0.99	0.2	0.09				1.0	–	–
				0.99	0.24	0.09				1.0	–	–

Table 23: Transferability from 2-bit fully quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.45	1.65	0.03	0.46	1.65	0.03	0.56	1.66	0.03	0.54	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
				0.37	1.65	0.03				0.53	1.66	0.03
				0.37	1.65	0.03				0.51	1.66	0.03
BIM	0.3	1.14	0.03	0.35	1.14	0.03	0.55	1.12	0.03	0.55	1.13	0.03
				0.25	1.14	0.03				0.50	1.12	0.03
				0.21	1.14	0.03				0.52	1.13	0.03
				0.21	1.14	0.03				0.49	1.13	0.03
CWl2	0.29	0.92	0.08	0.34	0.92	0.08	0.36	0.65	0.09	0.42	0.64	0.09
				0.19	0.93	0.08				0.22	0.7	0.09
				0.15	0.92	0.08				0.21	0.69	0.09
				0.18	0.93	0.08				0.19	0.7	0.09
SPSA	0.57	1.34	0.03	0.61	1.34	0.03	0.55	1.34	0.03	0.58	1.34	0.03
				0.44	1.34	0.03				0.56	1.33	0.03
				0.37	1.34	0.03				0.56	1.34	0.03
				0.42	1.34	0.03				0.53	1.34	0.03
ZOO	1.0	–	–	0.99	0.15	0.07	1.0	–	–	1.0	–	–
				1.0	–	–				0.98	0.08	0.04
				1.0	–	–				1.0	–	–
				1.0	–	–				1.0	–	–

Table 24: Transferability from 2-bit fully quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.61	1.65	0.03	0.57	1.66	0.03	0.64	1.66	0.03
				0.39	1.65	0.03				0.53	1.66	0.03
				0.44	1.65	0.03				0.54	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
BIM	0.34	1.15	0.03	0.68	1.14	0.03	0.58	1.13	0.03	0.72	1.13	0.03
				0.27	1.15	0.03				0.51	1.14	0.03
				0.24	1.15	0.03				0.52	1.13	0.03
				0.20	1.14	0.03				0.51	1.13	0.03
CWl2	0.32	0.90	0.08	0.6	0.86	0.07	0.31	0.84	0.10	0.57	0.81	0.1
				0.2	0.91	0.08				0.18	0.87	0.11
				0.29	0.91	0.08				0.21	0.85	0.10
				0.19	0.91	0.08				0.18	0.86	0.11
SPSA	0.41	1.36	0.03	0.64	1.36	0.03	0.49	1.37	0.03	0.59	1.36	0.03
				0.26	1.36	0.03				0.36	1.36	0.03
				0.43	1.36	0.03				0.41	1.37	0.03
				0.25	1.36	0.03				0.35	1.37	0.03
ZOO	0.96	0.56	0.08	0.86	0.57	0.08	0.97	0.69	0.1	0.90	0.66	0.09
				0.94	0.59	0.08				0.96	0.62	0.08
				0.93	0.62	0.08				0.96	0.61	0.09
				0.95	0.58	0.08				1.0	–	–

Table 25: Transferability from 2-bit weight-only quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.47	1.65	0.03	0.57	1.66	0.03	0.56	1.66	0.03
				0.18	1.65	0.03				0.38	1.66	0.03
				0.38	1.65	0.03				0.53	1.66	0.03
				0.39	1.65	0.03				0.53	1.66	0.03
BIM	0.34	1.15	0.03	0.39	1.15	0.03	0.58	1.13	0.03	0.57	1.13	0.03
				0.08	1.15	0.03				0.1	1.16	0.03
				0.25	1.15	0.03				0.53	1.13	0.03
				0.20	1.14	0.03				0.53	1.13	0.03
CWl2	0.32	0.90	0.07	0.38	0.89	0.07	0.31	0.84	0.10	0.35	0.84	0.10
				0.06	0.6	0.04				0.02	0.66	0.06
				0.19	0.91	0.08				0.17	0.86	0.11
				0.19	0.91	0.08				0.19	0.86	0.10
SPSA	0.41	1.36	0.03	0.45	1.36	0.03	0.49	1.37	0.03	0.46	1.36	0.03
				0.0	1.37	0.03				0.04	1.37	0.03
				0.43	1.36	0.03				0.4	1.37	0.03
				0.29	1.36	0.03				0.42	1.37	0.03
ZOO	0.96	0.56	0.08	0.66	0.55	0.08	0.97	0.69	0.1	0.98	0.64	0.09
				0.0	0.74	0.1				0.0	0.92	0.1
				0.95	0.61	0.08				0.97	0.7	0.09
				0.95	0.55	0.08				0.97	0.7	0.1

Table 26: Transferability from 2-bit weight-only quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.49	1.65	0.03	0.62	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.43	1.65	0.03				0.51	1.66	0.03
				0.17	1.65	0.03				0.37	1.66	0.03
				0.43	1.65	0.03				0.51	1.66	0.03
BIM	0.38	1.17	0.03	0.67	1.17	0.03	0.58	1.13	0.03	0.72	1.13	0.03
				0.35	1.17	0.03				0.49	1.13	0.03
				0.11	1.17	0.03				0.11	1.13	0.03
				0.33	1.17	0.03				0.49	1.13	0.03
CWl2	0.25	0.98	0.10	0.56	0.97	0.10	0.28	0.92	0.11	0.58	0.91	0.10
				0.19	0.99	0.09				0.17	0.92	0.11
				0.10	0.97	0.09				0.02	0.96	0.11
				0.18	0.98	0.10				0.15	0.93	0.11
SPSA	0.42	1.36	0.03	0.67	1.36	0.03	0.45	1.36	0.03	0.67	1.35	0.03
				0.39	1.37	0.03				0.42	1.36	0.03
				0.00	1.36	0.03				0.07	1.35	0.03
				0.34	1.36	0.03				0.42	1.36	0.03
ZOO	1.0	–	–	0.99	0.23	0.07	1.0	–	–	1.0	–	–
				1.0	–	–				1.0	–	–
				0.76	0.24	0.07				0.94	0.11	0.05
				1.0	–	–				1.0	–	–

Table 27: Transferability from 3-bit fully quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.49	1.65	0.03	0.51	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.44	1.65	0.03				0.52	1.66	0.03
				0.44	1.65	0.03				0.52	1.66	0.03
				0.43	1.65	0.03				0.52	1.66	0.03
BIM	0.38	1.17	0.03	0.46	1.17	0.03	0.58	1.13	0.03	0.56	1.13	0.03
				0.37	1.17	0.03				0.5	1.13	0.03
				0.34	1.17	0.03				0.51	1.13	0.03
				0.33	1.17	0.03				0.49	1.13	0.03
CWl2	0.25	0.98	0.10	0.29	0.98	0.10	0.28	0.92	0.11	0.33	0.92	0.11
				0.2	0.92	0.1				0.19	0.93	0.11
				0.18	0.98	0.09				0.17	0.92	0.11
				0.18	0.98	0.09				0.15	0.94	0.11
SPSA	0.42	1.36	0.03	0.56	1.37	0.03	0.45	1.36	0.03	0.44	1.36	0.03
				0.44	1.36	0.03				0.44	1.36	0.03
				0.36	1.36	0.03				0.45	1.36	0.03
				0.41	1.36	0.03				0.43	1.36	0.03
ZOO	1.0	–	–	1.0	–	–	1.0	–	–	1.0	–	–
				1.0	–	–				1.0	–	–
				1.0	–	–				1.0	–	–
				1.0	–	–				1.0	–	–

Table 28: Transferability from 3-bit fully quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.61	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
				0.45	1.65	0.03				0.52	1.66	0.03
				0.38	1.65	0.03				0.51	1.66	0.03
BIM	0.3	1.15	0.03	0.7	1.15	0.03	0.58	1.13	0.03	0.71	1.14	0.03
				0.24	1.15	0.03				0.5	1.14	0.03
				0.4	1.15	0.03				0.5	1.14	0.03
				0.2	1.15	0.03				0.48	1.14	0.03
CWl2	0.39	0.83	0.07	0.61	0.81	0.07	0.34	0.87	0.1	0.61	0.86	0.1
				0.27	0.83	0.07				0.22	0.89	0.11
				0.36	0.83	0.07				0.21	0.88	0.11
				0.24	0.82	0.07				0.21	0.89	0.11
SPSA	0.32	1.36	0.03	0.68	1.36	0.03	0.47	1.37	0.03	0.72	1.36	0.03
				0.27	1.36	0.03				0.42	1.37	0.03
				0.4	1.36	0.03				0.38	1.37	0.03
				0.26	1.36	0.03				0.33	1.37	0.03
ZOO	0.96	0.59	0.08	0.86	0.56	0.09	0.96	0.7	0.11	0.9	0.68	0.1
				0.94	0.56	0.09				0.95	0.63	0.11
				0.91	0.65	0.09				0.94	0.72	0.11
				0.94	0.52	0.08				0.94	0.66	0.11

Table 29: Transferability from 3-bit weight-only quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.47	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.39	1.65	0.03				0.51	1.66	0.03
				0.18	1.65	0.03				0.4	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
BIM	0.3	1.15	0.03	0.38	1.15	0.03	0.58	1.13	0.03	0.55	1.13	0.03
				0.26	1.15	0.03				0.5	1.14	0.03
				0.06	1.15	0.03				0.11	1.14	0.03
				0.23	1.15	0.03				0.48	1.14	0.03
CWl2	0.39	0.83	0.07	0.44	0.83	0.07	0.34	0.87	0.1	0.38	0.87	0.10
				0.30	0.83	0.07				0.23	0.88	0.11
				0.06	0.82	0.07				0.03	0.90	0.11
				0.26	0.83	0.07				0.21	0.89	0.11
SPSA	0.32	1.36	0.03	0.42	1.36	0.03	0.47	1.37	0.03	0.46	1.37	0.03
				0.3	1.36	0.03				0.37	1.37	0.03
				0.0	1.36	0.03				0.04	1.37	0.03
				0.3	1.36	0.03				0.32	1.37	0.03
ZOO	0.96	0.59	0.08	0.96	0.6	0.08	0.96	0.7	0.11	0.97	0.57	0.1
				0.94	0.57	0.08				0.95	0.68	0.1
				0.0	0.72	0.09				0.0	0.95	0.11
				0.95	0.55	0.09				0.96	0.72	0.11

Table 30: Transferability from 3-bit weight-only quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.62	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.4	1.65	0.03				0.53	1.66	0.03
				0.47	1.65	0.03				0.53	1.66	0.03
				0.18	1.65	0.03				0.4	1.66	0.03
BIM	0.32	1.15	0.03	0.69	1.15	0.03	0.55	1.13	0.03	0.73	1.13	0.03
				0.26	1.15	0.03				0.50	1.13	0.03
				0.42	1.15	0.03				0.48	1.13	0.03
				0.06	1.14	0.03				0.1	1.13	0.03
CWl2	0.36	0.86	0.07	0.63	0.82	0.06	0.33	0.81	0.10	0.62	0.79	0.09
				0.25	0.86	0.07				0.23	0.82	0.10
				0.36	0.87	0.07				0.321	0.82	0.10
				0.05	0.6	0.04				0.02	0.68	0.07
SPSA	0.33	1.36	0.03	0.64	1.36	0.03	0.45	1.37	0.03	0.66	1.36	0.03
				0.3	1.36	0.03				0.37	1.36	0.03
				0.44	1.36	0.03				0.41	1.37	0.03
				0.0	1.36	0.03				0.04	1.37	0.03
ZOO	0.97	1.45	0.18	0.95	0.89	0.13	0.99	0.7	0.14	0.99	0.31	0.09
				0.96	1.07	0.14				0.99	0.61	0.13
				0.97	1.45	0.16				0.99	0.8	0.09
				0.73	1.09	0.14				0.93	0.38	0.1

Table 31: Transferability from 4-bit fully quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.47	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.41	1.65	0.03				0.52	1.66	0.03
				0.4	1.65	0.03				0.53	1.66	0.03
				0.39	1.65	0.03				0.53	1.66	0.03
BIM	0.32	1.15	0.03	0.4	1.15	0.03	0.55	1.13	0.03	0.53	1.13	0.03
				0.29	1.15	0.03				0.50	1.13	0.03
				0.25	1.15	0.03				0.49	1.13	0.03
				0.26	1.15	0.03				0.47	1.13	0.03
CWl2	0.37	0.86	0.07	0.45	0.84	0.07	0.33	0.82	0.10	0.37	0.81	0.09
				0.29	0.87	0.07				0.22	0.82	0.10
				0.25	0.86	0.07				0.22	0.82	0.10
				0.25	0.86	0.07				0.21	0.82	0.10
SPSA	0.33	1.36	0.03	0.43	1.36	0.03	0.45	1.37	0.03	0.44	1.36	0.03
				0.34	1.36	0.03				0.39	1.36	0.03
				0.27	1.36	0.03				0.38	1.37	0.03
				0.3	1.36	0.03				0.39	1.37	0.03
ZOO	0.97	1.45	0.18	0.98	1.40	0.17	0.99	0.7	0.14	0.99	0.26	0.11
				0.97	1.42	0.17				0.99	0.11	0.05
				0.97	1.45	0.16				1.0	–	–
				0.97	1.45	0.17				0.99	0.27	0.8

Table 32: Transferability from 4-bit fully quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.63	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.4	1.65	0.03				0.52	1.66	0.03
				0.46	1.65	0.03				0.53	1.66	0.03
				0.4	1.65	0.03				0.51	1.66	0.03
BIM	0.31	1.15	0.03	0.71	1.15	0.03	0.56	1.13	0.03	0.74	1.13	0.03
				0.27	1.15	0.03				0.50	1.13	0.03
				0.42	1.15	0.03				0.5	1.13	0.03
				0.24	1.15	0.03				0.48	1.13	0.03
CWl2	0.39	0.85	0.07	0.65	0.83	0.06	0.37	0.80	0.09	0.61	0.79	0.09
				0.28	0.86	0.07				0.26	0.81	0.10
				0.39	0.87	0.07				0.26	0.81	0.10
				0.26	0.86	0.07				0.24	0.80	0.10
SPSA	0.34	1.36	0.03	0.71	1.36	0.03	0.46	1.37	0.03	0.66	1.36	0.03
				0.31	1.36	0.03				0.44	1.37	0.03
				0.43	1.36	0.03				0.38	1.37	0.03
				0.24	1.36	0.03				0.36	1.37	0.03
ZOO	0.96	0.6	0.08	0.86	0.56	0.08	0.96	0.69	0.1	0.88	0.69	0.09
				0.95	0.56	0.09				0.96	0.61	0.1
				0.93	0.65	0.09				0.95	0.66	0.09
				0.95	0.6	0.09				0.94	0.61	0.09

Table 33: Transferability from 4-bit weight-only quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.48	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.41	1.65	0.03				0.52	1.66	0.03
				0.4	1.65	0.03				0.52	1.66	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
BIM	0.31	1.15	0.03	0.39	1.15	0.03	0.56	1.13	0.03	0.55	1.13	0.03
				0.28	1.15	0.03				0.50	1.13	0.03
				0.24	1.15	0.03				0.51	1.13	0.03
				0.08	1.15	0.03				0.09	1.13	0.03
CWl2	0.39	0.85	0.07	0.47	0.984	0.07	0.37	0.80	0.09	0.40	0.79	0.09
				0.31	0.87	0.07				0.26	0.81	0.10
				0.29	0.87	0.07				0.26	0.81	0.10
				0.06	0.62	0.04				0.02	0.68	0.07
SPSA	0.34	1.36	0.03	0.48	1.36	0.03	0.46	1.37	0.03	0.44	1.36	0.03
				0.37	1.36	0.03				0.36	1.37	0.03
				0.26	1.36	0.03				0.42	1.37	0.03
				0.0	1.36	0.03				0.03	1.37	0.03
ZOO	0.96	0.6	0.08	0.96	0.58	0.08	0.96	0.69	0.1	0.97	0.61	0.09
				0.95	0.54	0.08				0.94	0.7	0.1
				0.96	0.57	0.09				0.96	0.66	0.1
				0.0	0.73	0.09				0.0	0.93	0.11

Table 34: Transferability from 4-bit weight-only quantized model to 1,2,3,4-bit weight-only quantized models.