TO COMPRESS OR NOT TO COMPRESS: UNDERSTANDING THE INTERACTIONS BETWEEN ADVERSARIAL ATTACKS AND NEURAL NETWORK COMPRESSION

Anonymous Authors[1]

ABSTRACT

As deep neural networks (DNNs) become widely used, pruned and quantised models are becoming ubiquitous on edge devices; such compressed DNNs are popular for lowering computational requirements. Meanwhile, recent studies show that adversarial samples can be effective at making DNNs misclassify. We, therefore, investigate the extent to which adversarial samples are transferable between uncompressed and compressed DNNs. We find that adversarial samples remain transferable for both pruned and quantised models. For pruning, the adversarial samples generated from heavily pruned models remain effective on uncompressed models. For quantisation, we find the transferability of adversarial samples is highly sensitive to integer precision.

1 INTRODUCTION

Deep Neural Networks (DNNs) perform well on a wide range of tasks, including image classification (Krizhevsky et al., 2012), object detection (Ren et al., 2015), reading comprehension (Seo et al., 2016) and machine translation (Bahdanau et al., 2015). They have proved to be an efficient method of harvesting information from large amounts of data and are expected to be ubiquitous in the future. Despite these successes, two questions remain crucial for deploying them in embedded systems. First, their substantial computational and memory requirements can make deployment challenging on power-limited devices. Second, as they start to appear in safety-critical applications, their reliability and security become a serious issue.

In order to compute DNNs efficiently on embedded systems, researchers have proposed various compression methods. The research has two main branches: 1) efforts to reduce their computation requirements; and 2) efforts to implement custom accelerators. *Pruning* directly reduces the number of parameters of DNNs – this reduction translates to fewer data movements and thus saves energy directly. *Quantisation* is another popular compression technique – it simultaneously reduces the memory footprint and decreases the energy cost of multiplications. Both compression methods are widely deployed on DNN accelerators. For instance, Efficient Inference Engines (EIE) use pruning, quantisation and encoding techniques for energy efficiency (Han et al., 2016a); the Sparse CNN (SCNN) accelerator first requires network parameters to be pruned and encoded, then performs computations directly using the encoded data format (Parashar et al., 2017). It seems certain that pruning, quantisation and other compression techniques will be essential for future DNN accelerators on embedded devices.

In the meantime, adversarial machine learning research has found DNNs to be sensitive to small perturbations of the input images, with the result that they can often be fooled easily using specially-crafted adversarial inputs (Szegedy et al., 2013). Such adversarial samples could become real threats in safety-critical systems; attackers might try to manipulate autonomous vehicles (Eykholt et al., 2018) or break into smart phones by tricking the speaker recognition system (Carlini et al., 2016).

In this paper, we are particularly interested in the portability of adversarial samples. Might an attacker learn how to break into widely-deployed low-security systems and then use the adversarial samples as a springboard to attack other systems?

We make the following contributions in this paper.

- We investigated the effects of different DNN compression mechanisms on adversarial attacks.
- We have developed the first compression-aware machine learning attack taxonomy and used it to evaluate the transferability of adversarial samples between compressed and uncompressed models.
- In pruning, we found that adversarial samples are trans-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Systems and Machine Learning (SysML) Conference. Do not distribute.

ferable between compressed and uncompressed models. However, adversarial samples generated from uncompressed models are less effective on compressed models in fast-gradient-based attacks.

- In quantisation, we found that adversarial samples are transferable between compressed and uncompressed models. However, a reduction in integer precision provides clipping effects and marginally limits transferability in fast-gradient-based attacks.

2 Related Work

2.1 Pruning

Pruning directly reduces the number of parameters in a DNN model, and thus the number of off-chip to on-chip data transfers on modern DNN accelerators (Chen et al., 2016). If the architecture allows, pruning may also reduce the computation cost (Kim et al., 2018). Consider a weight tensor ($\mathbf{W_n}$); fine-grained pruning is simply performing an element-wise multiplication ($\odot$) between a mask operator $\mathbf{M_n}$ and the original weight tensor ($\mathbf{W_n}$).

$$\mathbf{W_n}' = \mathbf{W_n} \odot \mathbf{M_n} \tag{1}$$

Han et al. first proposed pruning a DNN by applying a threshold to the DNN's parameters (Han et al., 2016b). In this case, the mask ($\mathbf{M_n}$) consists of thresholding by a single value α.

$$\mathbf{M_n} = h_k(\mathbf{W_n}^{(i,j)}) = \begin{cases} 0 & \text{if } \alpha > |\mathbf{W_k}^{(i,j)}| \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

Using this simple one-shot pruning technique, Han et al. were able to reduce the number of parameters in AlexNet by 9x and VGG16 by 13x (Han et al., 2016b). In their implementation, the masking and fine-tuning happens iteratively but the masked values are not allowed to recover in later stages.

Guo et al. subsequently proposed dynamic network surgery (DNS), which allows pruned parameters to recover at later stages(Guo et al., 2016). The approach is to condition the mask using the following equation, where α and β are two constants.

$$\mathbf{M_n} = h_k(\mathbf{W_n}^{(i,j)}) = \begin{cases} 0 & \text{if } \alpha > |\mathbf{W_k}^{(i,j)}| \\ \mathbf{M_n}^{(i,j)} & \text{if } \alpha \leq |\mathbf{W_k}^{(i,j)}| \leq \beta \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

Values that become bigger at later stages are allowed to rejoin the fine-tuning process. Guo et al. demonstrated higher compression rates on a large range of networks compared to Han et al. In this paper, we generate pruned DNNs using the DNS method.

2.2 Quantisation

Quantisation refers to using fewer number of bits than a 32-bit single-precision floating-point representation for parameters in a DNN. Single-precision floating point numbers are widely used on modern CPUs and GPUs, however, Hubara et al. presented that low-precision fixed-point numbers can be used for neural network inference with nearly no loss of accuracy (Hubara et al., 2017). In the extreme case, the parameters of a DNN can be quantised to either binary values (Courbariaux et al., 2016) or ternary values (Li et al., 2016). Aggressive quantisation methods, such as binarisation and ternarisation, bring large benefits to potential DNN hardware accelerators but suffer from a significant loss of accuracy. For resource-constrained devices, using a low-precision fixed-point representation reaches a balance between model accuracy and hardware performance (Lin et al., 2016). The narrower bitwidth means direct reductions in memory requirement and fixed-point multiplications are less computationally expensive compared to single-precision floating point. In this paper, we generate models that use fixed-point parameters at various levels of precision.

2.3 Adversarial Attacks

Szegedy et al. (Szegedy et al., 2013) discovered that models trained on huge datasets, despite generalising well, are all vulnerable to adversarial samples. Misclassification can happen with certain imperceptible perturbations on the data samples. Interestingly, all the samples they used were within the expected data distribution and only a small specifically-crafted amount of noise was added. They observed that models of different configurations, trained on different datasets, misclassify the same samples. Finally, they noted that training a model on adversarial samples helps make it more robust against them. However, this defence is not always practical; their approach based on L-BFGS requires an expensive constrained optimisation which iterates many times.

A follow-up paper by Goodfellow et al. (Goodfellow et al., 2015) explored the underlying reasons for the existence and generalisability of adversarial samples. They argue that such samples are an artefact of high-dimensional dot-products, and attacks are generalisable because different models learn similar functions when trained to perform the same task. Additionally, they presented two methods to generate adversarial samples in a white-box setting, the fast gradient method (FGM) and the fast gradient sign method (FGSM). Finally, they discovered that RBF-based networks are much more resistant to adversarial samples.

Papernot et al. (Papernot et al., 2016b) came up with an-

other way to generate adversarial samples. They use the gradients of a network to construct saliency maps for the input to discover which input values are so sensitive that a change can drive an misclassification. They showed that their method has the flexibility of being used in both supervised and unsupervised settings and is capable of generating samples with a user given priority on particular properties of the inputs. Finally, they also observed that adversarial attacks become harder when models have been trained with adversarial samples.

There is now a growing corpus of research on the transferability of adversarial samples (Szegedy et al., 2013; Goodfellow et al., 2015; Papernot et al., 2016a; Tramèr et al., 2017). Transferability refers to the ability of an adversarial sample to evade the correct classification on two different classifiers trained to perform the same task.

Goodfellow et al.(Goodfellow et al., 2015) and Warde-Farley & Goodfellow(Warde-Farley & Goodfellow, 2016) empirically found that adversarial examples usually occur in large, continuous spatial regions. Tramer et al.(Tramèr et al., 2017) found out that each of the models differs in the dimensionality of its subspaces. A higher number of dimensions increases the chance that the subspaces of different models intersect, leading to transferable samples.

Transferable adversarial are a real hazard for model deployment, as attacks developed on a particular type of classifier anywhere can potentially by deployed everywhere. Papernot et al. (Papernot et al., 2017) have shown that an adversary can successfully perform attacks without any knowledge of a model's internal parameters – it is often enough to approximate a model with another known model against which one can build adversarial samples.

3 METHODOLOGY

3.1 Attack Taxonomy

In this paper, we are interested in the interaction between adversarial attacks and model compression, and we investigate three specific attack scenarios. We use the name baseline model (i.e. model without any compression) for a pretrained network that is dense and whose parameters are represented using full-precision (float32) values. We then use the name compressed models to denote models that have been compressed using pruning or quantisation.

- Scenario 1: Adversarial attacks occur on each individual compressed model, with the adversarial examples generated and applied on the same model.
- Scenario 2: Adversarial samples are generated from the baseline model but applied on each compressed model.
- Scenario 3: Adversarial samples are generated from compressed models but applied on the baseline model.

In the first scenario, adversarial samples are generated from each compressed model. Attackers can access these compressed models fully, and generate adversarial samples for each one individually. This is the case where attackers buy products and figure out how to attack them.

The second scenario makes the assumption that attackers can only access the baseline model to generate adversarial samples, which are then used to attack various compressed models. Attackers are not allowed to fetch any gradients from compressed models. This is the case where firms take publicly-available models and compress them to run more efficiently on edge devices. Attackers can easily find the same public model and craft adversarial samples to attack proprietary edge devices.

The third scenario assumes that only compressed models are visible to attackers, and attackers generate adversarial samples using compressed models to attack the hidden baseline model. In practice, companies now deploy various compressed neural network models on edge devices that are exposed to end-users. The assumption is the attackers have the ability to access these compressed models on the edge and create adversarial samples from them to attack the hidden baseline model. This then leads to the second scenario; the attacker's knowledge and toolkit can be generalised to other products from the same firm. Figure 1 shows the second and third attack scenarios.

For example, modern anti-virus (AV) software uses DNNs to detect malware behaviour. Some AV modules detect such behaviours in offline mode. When deploying a compressed model in such an application, how likely is it that malware could analyse the compressed model and use this to evade the full model, and thus the firm's other AV products? Similarly, if an alarm company deploys a compressed model for intruder detection in CCTV equipment, could an intelligence agency that buys such equipment figure out how to defeat not just that product but all the products derived from the same full model? And just as a new type of software attack such as Heartbleed or Meltdown can cause widespread disruption by requiring thousands of disparate systems to be patched, so portable adverse examples could force upgrades to large numbers of diverse embedded systems.

3.2 Networks and Compression Methods

We use LeNet5 (LeCun et al., 2015) and CifarNet (Zhao et al., 2018) for our experiments on MNIST (LeCun et al., 2010) and CIFAR10 (Krizhevsky et al., 2014) datasets. The LeNet5 model has 431K parameters and classifies MNIST digits with an accuracy of 99.36%. The CifarNet classifier (Zhao et al., 2018) has 1.3M parameters and achieves

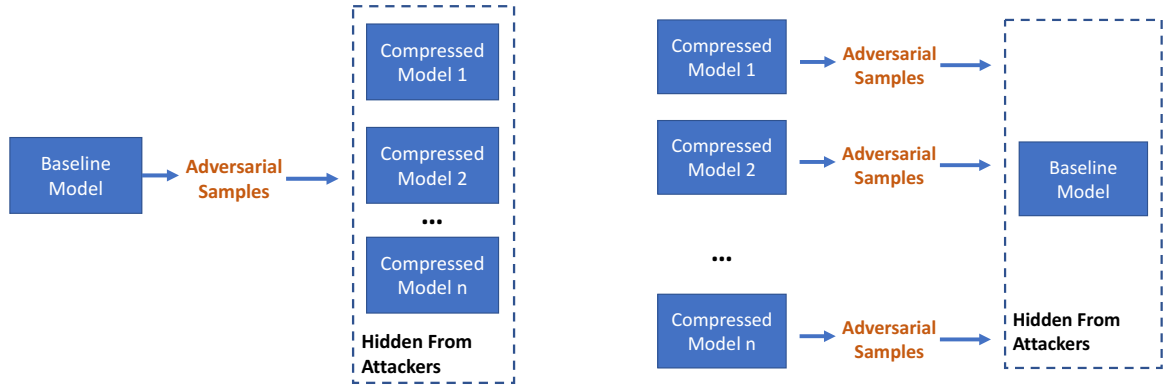


Figure 1. Two different attack setups. Attackers only generate adversarial samples based on baseline model (left), or attackers generate adversarial samples based on compressed models (right).

85.93% classification accuracy.

In terms of compression methods, we implemented the following:

- Fine-grained pruning on weights.
- Fixed-point quantisation on both weights and activations.

We used the Mayo tool to generate pruned and quantised models (Zhao et al., 2018), and fine-tuned these models after pruning and quantisation. For each pruning density or quantised bitwidth, we retrain 350 epochs for LeNet5 and 300 epochs for CifarNet with three scheduled learning rate decays starting from 0.01. For each decay, the learning rate decreases by a factor of 10.

Applying pruning on a pretrained model shrinks the number of parameters and thus the memory footprint of future AI ASICs. We use fixed-point quantisation on both weights and activations of a DNN. Quantising both weights and activations means that computations operate in low-precision fixed-point formats, which cut the time and energy cost both data moves and computations. For fixed-point quantisation, we use a 1-bit integer when bitwidth is 4, a 2-bit integer when bitwidth is 8, and the rest of the fixed-point quantisations all have 4-bit integers.

3.3 Adversarial attacks

In the work reported in this paper we used three popular attacks developed in the research community. What follows are mathematical definitions of the attacks and comments about their behaviour.

Goodfellow et al. (Goodfellow et al., 2015) first introduce the fast gradient method (FGM) and fast gradient sign method (FGSM) to develop attacks. For the definitions we will use the following notation: $\boldsymbol{\theta}$ represents the parameters of the model, $\boldsymbol{X}$ represents the inputs, while y and y_l represents the outputs and labels respectively. We can then use $J(\boldsymbol{\theta}, \boldsymbol{X}, y_l)$ to represent the cost function. The original FGM and FGSM perturbations are computed as shown in Equation (4) and Equation (5) respectively, where ϵ is a hyperparameter and the function $\nabla_X()$ computes the first-order derivative with respect to input X.

$$\eta = \epsilon(\nabla_X J(\boldsymbol{\theta}, \boldsymbol{X}, y)) \tag{4}$$

$$\eta = \epsilon \text{sign}(\nabla_X J(\boldsymbol{\theta}, \boldsymbol{X}, y)) \tag{5}$$

Kurakin et al. presented an iterative algorithm based on FGM and FGSM methods. In Algorithm 1, we present an iterative FGSM (IFGSM), where the adversarial samples X_n^{adv} are generated for the nth iteration.

During each iteration, the intermediate results get clipped to make sure that the resulting adversarial images lie within an ϵ interval from the previous iteration.

Algorithm 1 IFGSM

Input: data X_{in}
Initialize $X_0^{adv} = X_{in}$.
for $n = 0$ **to** $m - 1$ **do**
  $\boldsymbol{N} = \epsilon \text{sign}(\nabla_X J(\boldsymbol{\theta}, \boldsymbol{X}_n^{\text{adv}}, y_l))\}$
  $\boldsymbol{X}_{n+1}^{\text{adv}} = Clip_{X,\epsilon}\{\boldsymbol{X}_n^{\text{adv}} + \boldsymbol{N}\}$
end for

Kurakin et al. also presented an iterative version of FGM where instead of just using the sign to determine the direction of a gradient, the gradient amplitudes contribute to the corresponding gradient update step. The iterative FGM (IFGM) is nearly identical to the IFGSM with a small change of $\boldsymbol{N} = \epsilon \nabla_X J(\boldsymbol{\theta}, \boldsymbol{X}_n^{\text{adv}}, y_l)$.

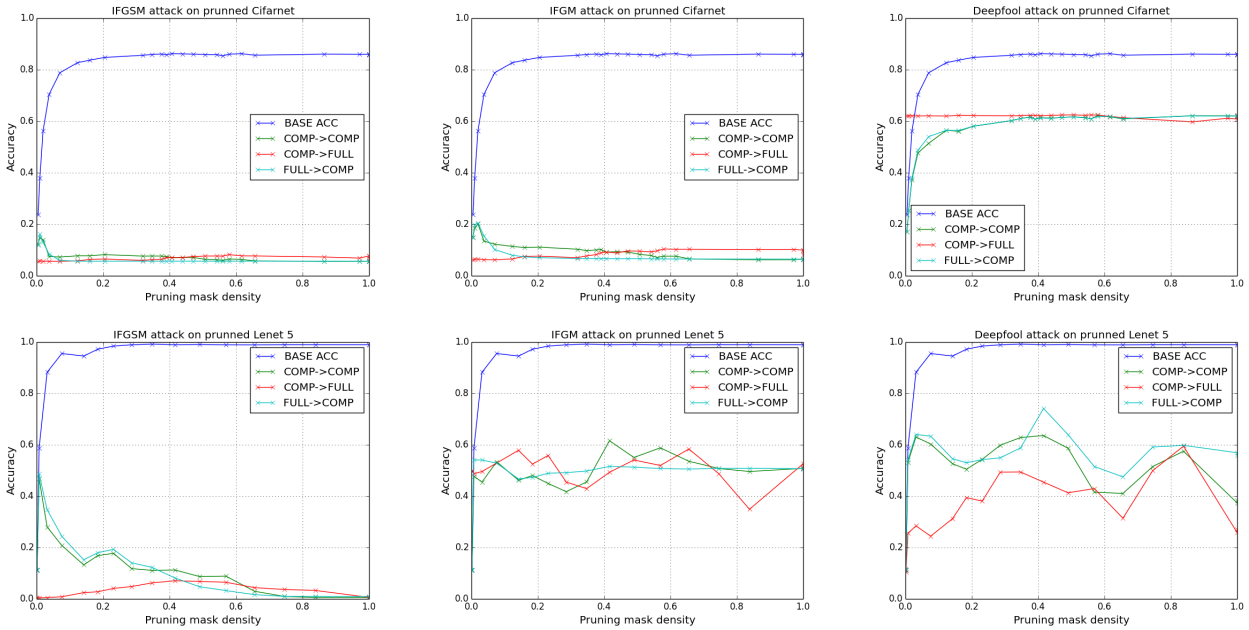


Figure 2. Transferability properties for pruning. The green, red and cyan lines represent the first, second and third attack scenarios respectively. The blue line show the accuracies of pruned models without any attacks.

Moosavi et al. (Moosavi-Dezfooli et al., 2016) featured another attack called 'Deepfool', which is also based on iterative gradient adjustment. However, Deepfool is different from IFGSM in that it does not scale and clip gradients. It is based on the idea that the separating hyperplanes in linear classifiers indicate the decision boundaries of different classes. It therefore iteratively perturbs an image $\boldsymbol{X}_0^{adv}$, linearises the classification space around $\boldsymbol{X}_n^{adv}$ and moves towards the closest decision boundary. The step is chosen according to the l^0, l^1 or even the l^p norm of $\boldsymbol{X}_n^{adv}$ to the last-found decision boundary. The applied step is then used as $\boldsymbol{X}_{n+1}^{adv}$.

In practice Deepfool is found to produce smaller perturbations than the original IFGSM, which makes it a more precise attack (Moosavi-Dezfooli et al., 2016). In this paper we used an L2 norm-based version of Deepfool – the attack stops accumulate perturbations after it successfully crosses the closest decision boundary.

It should be noted that in this particular paper we were not interested in the absolute accuracy but the relative behaviours with a set of fixed parameters for adversarial attacks. We chose the strongest white box adversary model and picked three of the strongest iterative attacks. For all the experiments, we did not sweep all the possible hyper-parameters for the adversarial attacks, but picked empirically sensible hyper-parameters. We used an ϵ of 0.02 and performed the attacks for 12 epochs. For both IFGM and IFGSM we used $Clip$ with a minimum value of 0 and maximum value of 1.

4 Evaluation

4.1 Pruning

Goodfellow et al. explained the existence of adversarial samples as follows (Goodfellow et al., 2015). Consider an adversarial sample as the original input $\boldsymbol{x}$ with an additional noise $\boldsymbol{\eta}$. When passing through multiple layers of matrix multiplication, this small noise eventually grows to a large enough value to shift the decision of the whole model. Given weights $\boldsymbol{w}$ of a particular layer of a neural network and adversarial sample $\widetilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$, the output of that particular layer is $\boldsymbol{w^T}\widetilde{\boldsymbol{x}} = \boldsymbol{w^T}\boldsymbol{x} + \boldsymbol{w^T}\boldsymbol{\eta}$, and the adversarial perturbation causes the the output activations to grow by $\boldsymbol{w^T}\boldsymbol{\eta}$.

Figure 2 shows the performance of IFGSM, IFGM and DeepFool on pruned models under three different attack scenarios. The horizontal axis shows the densities of DNNs, effectively the ratio of the number of non-zero values to the total number of values. The vertical axis presents test accuracies of DNNs. Apart from showing the accuracies of pruned networks without any attacks (BASE ACC), we present the accuracies of the pruned models with three different attack scenarios. The first scenario corresponds to COMP $\rightarrow$ COMP, the second scenario and third scenario corresponds to FULL $\rightarrow$ COMP and COMP $\rightarrow$ FULL respectively.

The first thing to become apparent is that samples generated from the compressed models are transferable to the

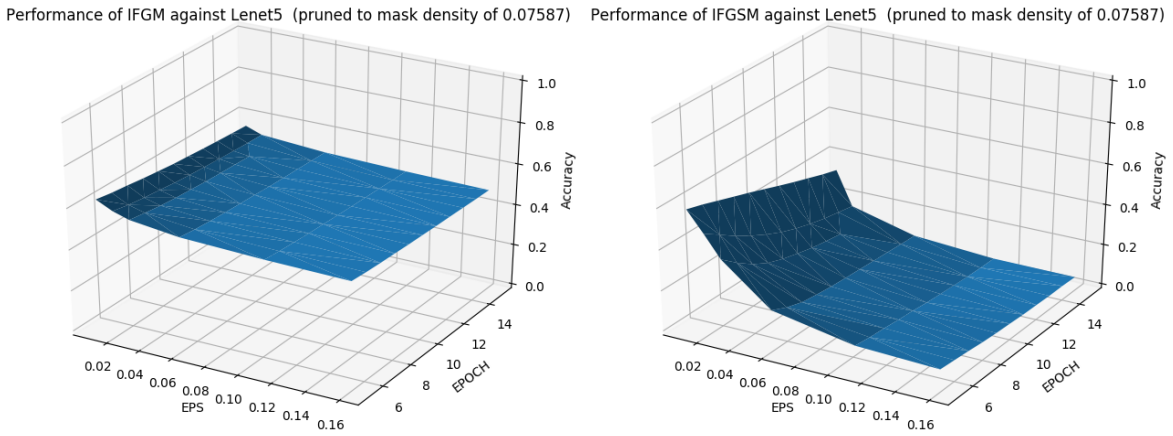


Figure 3. Lenet5 accuracy with IFGSM and IFGM-generated adversarial samples with different $epsilon$ values and number of epochs

baseline model. This finding reinforces the idea that the adversarial samples are not scattered randomly but are reside in large and contiguous high-dimensional spaces, enabling them to survive the effect of pruning. In the cases of both IFGSM and IFGM, transferability becomes worse when the adversarial samples are generated at smaller densities. After exploring the performance of these attacks, we find the adversarial samples generated by IFGSM on networks with smaller densities are smaller both in terms of pixel changes and the perturbation amplitude.

We suggest that pruning smooths the decision space by removing DNN weights that have little impact. This ultimately has an effect on IFGSM – with unimportant parts removed, the gradients now follow the path towards the most important and prominent parts of the space. As a result, relatively small perturbations based on compressed models generalise very well on the uncompressed model.

Unlike scenario 2, in scenario 1 and 3 both IFGSM and IFGM seem to become less effective as the density decreases, but probably for different reasons.

In scenario 1 – one compressed model attacking another – the decrease in attack efficacy should be attributed to the fact that the previously-used step ϵ was no longer large enough to reach a successful decision boundary along the gradients. Figure 3 shows that small ϵ values are not efficient for attacking pruned models. On both IFGM and IFGSM, we notice an increase in accuracy when the ϵ values are small. Also, IFGSM shows a better attacking performance but suffers more from the effect of small ϵ values. Such behaviour in case of IFGM and IFGSM can be explained by the use of the *Clip* function. The use of ϵ and *Clip* function was originally designed to make sure that the changes of the original images are not too large, and apparently the attacks require a large ϵ to remain effective on pruned models.

In scenario 2 – using an uncompressed model to attack a compressed one – we attribute this behaviour to the massively regularized space. The adversarial samples are influenced by the local minima of the baseline model. Since pruning serves as a regularization method and removes local minima in the optimization space, the adversarial samples generated from pruned models have better transferability to uncompressed models.

Deepfool shows a similar behaviour to IFGSM and IFGM at high densities, in that a full model attacking a compressed one achieves similar accuracy to the other way around. However, unlike the behaviour we saw before, Deepfool increases its efficiency when the pruning density decreases. We hypothesize that unlike FGM implementation, Deepfool does not use the *Clip* function, and counteracts the movement of the decision boundary by increasing the amplitude of the gradient effect.

Summary

1. For IFGM and IFGSM, the transferability of adversarial samples between pruned and full models is affected by the reduction of density.
   - It becomes easier to attack the baseline model using a compressed model when densities are low.
   - It becomes harder to attack the compressed model using a baseline model when densities are low.
2. For scenarios 1 and 3, a more fine-grained attack method (DeepFool) performs better as the network densities decrease.
3. For scenarios 1 and 3, clipped attacking methods (IFGSM and IFGM) perform worse as the network densities decrease.

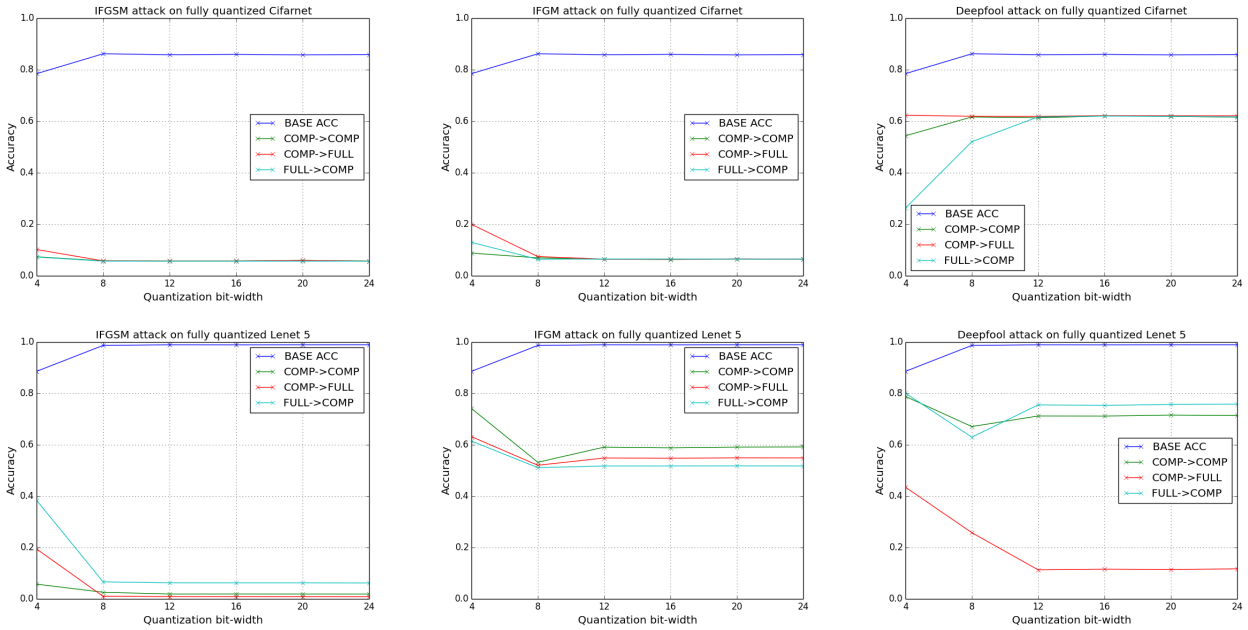


Figure 4. Transferability properties for quantising both weights and activations. The green, red and cyan lines represent the first, second and third attack scenarios respectively. The blue line show the accuracies of quantised models without any attacks.

4.2 Fixed-point Quantisation

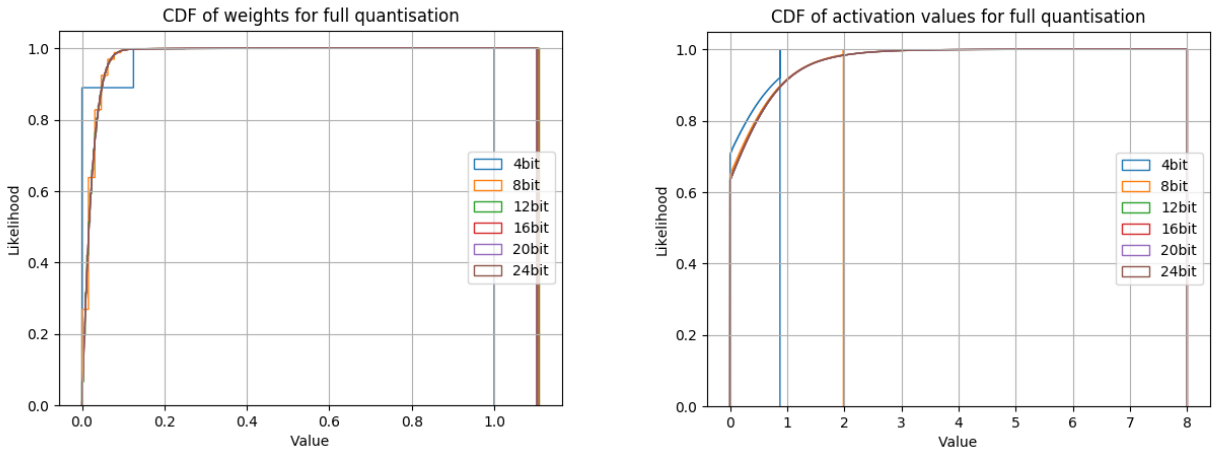
Fixed-point quantisation refers to quantising both weights and activations to fixed-point numbers for a neural network to perform inference using low-cost fixed-point multiplications. Figure 4 shows the performance of adversarial attacks on quantised models under three different attack scenarios In general, the performance of attacks stays nearly constant at bitwidths that are higher than 8. When using fewer bits on both weights and activations, the model shows a defensive behaviour mainly because of the reduction in integer precision.

Intuitively, we have two effects when values are quantised to smaller bitwidths. First, a smaller bitwidth might indicate less fractional bits. The reduction in fraction bits causes a loss in precision, and more zeros are generated because of the limited numerical representations. The generation of zeros implies that the reduction in fractional bits introduces the same effect as pruning. Second, a smaller bitwidth might hint fewer integer bits, implying that weight and activation values are capped to smaller values. Combining these two effects, to give an example in our setup, models in 4-bit fixed-point quantisation have smaller weight and activation values and contain more zeros compared to models at higher precisions. In Figure 5.a, we show the cumulative distribution function (CDF) of CifarNet with different fixed-point quantisations. There are clearly more zeros in the 4-bit CifarNet – its cumulative density reaches around 0.9 when value is at 0. The clipping effect is also more obvious on the 4-bit model, since it only has 1-bit integer, we can see the 4-bit model has its weights CDF reach 1.0 before all other bitwidths in Figure 5.a.

Using the adversarial examples generated by compressed models to attack the baseline model (Scenario 1), we observe both IFGM and IFGSM methods become less effective on LeNet5 and CifarNet. The same phenomenon occurs when we use adversarial samples generated by the baseline model to attack quantised models (Scenario 2). We suggest that during quantisation, reducing fractional bits will not hugely impact the attacks' performance at high bitwidths, but introduces a similar effect to pruning at low bitwidths. In addition, reducing integer bits essentially introduces large differences between the baseline model and the quantised ones for adversarial attacks.

By reducing the length of the fraction, the rounding process of fixed-point quantisation becomes more lossy. Uniformly adding quantisation noise to each individual weight does not affect attack performance. As we can see in Figure 4, all three attack scenarios show a stable performance when bitwidths are higher than 8, where the difference lies in the length of the fraction. When the network gets quantised down to 4 bits, quantisation behaves rather like pruning – a large part of the network gets zeroed out.

In terms of reducing the integer bitwidth, we are clipping the numerical values. Theoretically, clipping the values of weights is different from clipping the values of activations. For the former, we first consider how to create an adversar-



(a) Weights (b) Activations

Figure 5. Cumulative distribution function (CDF) for all weights and activations in quantised CifarNet. Ten randomly chosen input images from the validation dataset were used for generate CDF of activation values.

ial sample with minimal perturbation. An easy approach is to focus on changing pixels that can perturb important activations. Intuitively, the way to achieve such a perturbation with minimal changes on the input image is to focus on tweaking pixels with large weight values that are connected to important activations. Thus a small change in input image pixels can produce the maximal effect on activation values. When weights are clipped, adversarial attacks see more weights with equal importance because they all saturate to the same maximal value. This makes the attack transferability between quantised models and baseline models challenging. For example, on a quantised network, an adversarial example X_i considers $w_i = max(W_i)$ to be the largest weight associated with the important activations among all the weights (W_i) associated with activation a_i. This relationship $w_i = max(W_i)$ might break on the baseline model and thus the adversarial sample becomes less effective. In Figure 5, a 4-bit fixed-point quantisation clearly shows a clipping effect on weight values, which contributes to the marginal defensive nature we observed in Figure 4.

When activations are clipped to a smaller maximal value, the transferability of adversarial attacks between quantised and baseline models becomes worse. Figure 5.b shows how activations are clipped to different maximum values when using fixed-point quantisation. Adversarial attacks work by adding artificial changes in input pixels to drive activations to unwanted states. Consider a simple case, where an adversarial example overdrives one activation to be larger than others in the same layer to cause a misclassification. Clipping the activation values forces adversarial attacks to find more fine-grained opportunities in terms of the relative importance between activations, which is significantly harder.

Both clipping weights and clipping activations can significantly affect the performance of attacks. As we can observe that at smaller bit widths, all three scenarios show an increase in accuracy when attacked by IFGM and IFGSM. In terms of transferability, as shown in Figure 4, when applying both the IFGSM and IFGM, adversarial examples remain transferable between quantised and baseline models when the fractional bits are lost. When the integer bits start to decrease, we see the transferability of adversarial examples become worse.

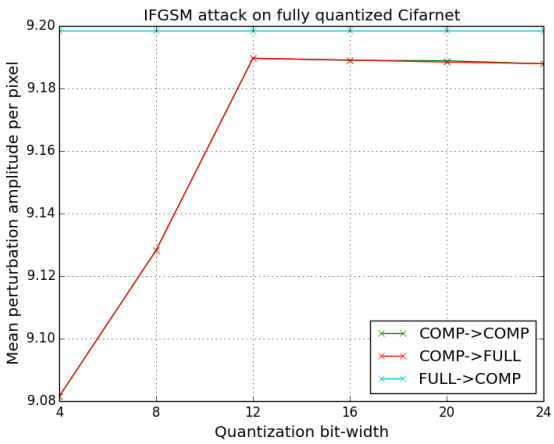


Figure 6. Mean perturbation amplitude per pixel for IFGSM.

Since saturation happens when the integer bitwidth is small, a lot of originally big values are capped to a given range. In other words, weights of a quantised DNN are clustered into a smaller group of values that are closer together. A small perturbation can now affect a large number of weights since

many weights now have similar importance. Figure 6 shows how adversarial samples generated from quantised DNNs now require smaller perturbations to attack the baseline model.

Surprisingly, we find that Deepfool, unlike IFGM, generates samples that were transferable in scenarios 2 and 3 and has shown relatively good attack performance for Lenet5 in Scenario 2. We attribute this difference to the regularisation benefit of quantisation. Regularization makes it easier to generate adversarial samples for models that overfit. As with pruning, quantisation smooths the decision space and Deepfool's use of unclipped gradient amplitudes allows it to reach the decision boundary.

Although IFGSM show slightly higher accuracies at low precision, this protective behavior in quantisation is only marginal. In addition, the adversarial attacks still show good performances in all three scenarios compared to both IFGM and DeepFool if we consider the classification accuracies (Figure 4).

Summary

1. The transferability of adversarial samples between quantised and non-quantised models is not affected by the reduction in fractional bitwidth at high precision.

2. Aggressive reductions in fractional bits introduce the same effect as fine-grained pruning.

3. Smaller integer bit widths on weights and activations make it marginally harder to attack the baseline model using adversarial samples generated from compressed models. This suggests that semantic class information is contained in both activations and weights.

4. For transferability, in terms of values of classification accuracies, we have observed the best performance with IFGSM.

5 Conclusion

This paper reports an empirical study of the interaction between adversarial attacks and neural network compression.

Both quantisation and pruning sparsify the network, i.e. introduce a greater number of zeros into the network. Samples generated from heavily pruned models work effectively on the underlying baseline model. However, low-density DNNs are somewhat defensive when attacked by adversarial samples generated from the baseline model using fast-gradient-based methods. Quantisation is different in that adversarial samples from fast-gradient-based methods become marginally harder to transfer when models are heavily quantised. This defensive behaiour appears due to the reduction in integer bits of both weights and activations rather than to the truncation in fractional bits.

The broader implications are that attacks on DNN classifiers that involve adversarial inputs may be surprisingly portable. Even if a firm ships only a compressed version of its classifier in widely distributed products, such as IoT devices or apps, attacks that people discover on these compressed classifiers may translate fairly easily to attacks on the underlying baseline model, and thus to other compressed versions of the same model. Just as software vulnerabilities such as Heartbleed and Spectre required the patching of many disparate systems, so also a new adversarial sample may defeat many classifiers of the same heritage. Firms should be aware that while shipping a compressed classifier may give real benefits in terms of performance, it may not provide much in the way of additional safety or security.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*, 2015.

Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., and Zhou, W. Hidden Voice Commands. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016.

Chen, Y.-H., Emer, J., and Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pp. 367–379. IEEE Press, 2016.

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

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*, 2015.

Guo, Y., Yao, A., and Chen, Y. Dynamic network surgery for efficient DNNs. In *Advances in Neural Information Processing Systems*, 2016.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. EIE: Efficient Inference Engine on

Compressed Deep Neural Network. *SIGARCH Comput. Archit. News*, 44(3):243–254, June 2016a. ISSN 0163-5964. doi: 10.1145/3007787.3001163.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *International Conference on Learning Representations (ICLR)*, 2016b.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, pp. 6869–6898, 2017.

Kim, D., Ahn, J., and Yoo, S. Zena: Zero-aware neural network accelerator. *IEEE Design & Test*, 35(1):39–46, 2018.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Krizhevsky, A., Nair, V., and Hinton, G. The CIFAR-10 dataset. 2014.

LeCun, Y., Cortes, C., and Burges, C. MNIST handwritten digit database. 2, 2010.

LeCun, Y. et al. LeNet-5, convolutional neural networks. pp. 20, 2015.

Li, F., Zhang, B., and Liu, B. Ternary weight networks. *1st NIPS Workshop on Efficient Methods for Deep Neural Networks (EMDNN)*, 2016.

Lin, D., Talathi, S., and Annapureddy, S. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pp. 2849–2858, 2016.

Moosavi-Dezfooli, S., Fawzi, A., and Frossard, P. DeepFool: a simple and accurate method to fool deep neural networks. 2016.

Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint*, 2016a.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. pp. 372–387, 2016b.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. pp. 506–519, 2017.

Parashar, A., Rhu, M., Mukkara, A., Puglielli, A., Venkatesan, R., Khailany, B., Emer, J., Keckler, S. W., and Dally, W. J. SCNN: An accelerator for compressed-sparse convolutional neural networks. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 27–40, June 2017. doi: 10.1145/3079856.3080254.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. Bidirectional attention flow for machine comprehension. *arXiv preprint*, 2016.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. The space of transferable adversarial examples. *arXiv preprint*, 2017.

Warde-Farley, D. and Goodfellow, I. Adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, pp. 311, 2016.

Zhao, Y., Gao, X., Mullins, R., and Xu, C. Mayo: A framework for auto-generating hardware friendly deep neural networks. 2018.