Assessing and improving the robustness of Binarized Neural Networks

Benoît Ronval[a;*] **and Siegfried Nijssen**[a]

[a]UCLouvain

Abstract. Several recent publications have studied the robustness of classical neural networks and how to improve robustness thanks to adversarial training. These techniques are however hard to apply on Binarized Neural Networks (BNNs). In this work, we address this challenge and introduce a method to perform adversarial training on BNNs, by using a recent SAT-solver based technique for finding adversarial examples. This method is able to increase the robustness of BNNs according to a number of different metrics: adversarial accuracy, time taken by the considered attack and resistance of the model to this attack. It can also be generalized easily to other methods of verification and other types of models as it is a simple extension of the classical training algorithm.

1 Introduction

In recent years Deep Neural Networks (DNNs) have proven their ability to achieve good predictive performance on a number of different tasks, especially ones with structured inputs such as images or texts. For a long time, these good results were considered sufficient and justify the use of deep learning. However, more and more concerns about the flaws of DNNs have started to arise in recent years. Their tendency to overfit data [18], their huge power consumption [19] or even their lack of explainability [11] are all examples of known problems in the domain of deep learning.

An important criterion that we will focus on in this paper is the robustness of models. Robust models are here models that cannot be fooled by small modifications of the data given as input. This criterion is motivated by a need to ensure that a model will behave as expected when it processes two similar inputs. In other words, robust models should have consistent and satisfactory performance across a range of input variations, from light noise to adversarial samples, which are samples that seek to fool neural networks. Different methods to evaluate the robustness of a model exist in the literature, which we will list later in this paper.

The aforementioned problems have already been explored by many researchers and different ideas to address them have been proposed. Even if none of these approaches is able to solve entirely the tackled problems, they are at least able to mitigate them.

An example of such an idea is the Binarized Neural Networks (BNNs) [6]. The use of binary values inside a network is able to reduce power consumption, allowing us to use them on smaller devices, for example, smartphones. BNNs also turn out to be useful when it comes to the concept of robustness, with a direct link to the domain of satisfiability (SAT) solvers thanks to its quantized parameters.

* Corresponding Author. Email: benoit.ronval@uclouvain.be

Concerning robustness, there is a distinction between the attack and the actual process to improve the robustness. The objective of the attack against a DNN is to create a sample with a small perturbation that is able to fool the considered model. An input created to this end is called an adversarial example. Many different attacks exist, especially for classical neural networks, all of them having strengths and weaknesses when comparing between them. The most known attack is most probably the Fast Gradient Sign Method (FGSM) [4] which uses the sign of the gradient and a multiplicative factor to create noise used in the generation of the adversarial sample.

On the other end, one could also pay special attention to this property during the training of a model. In this situation, when we try to make a neural network robust, we call this process an adversarial training process. Different methods already exist for classical neural networks, but to the best of our knowledge, this problem has not been specifically studied for BNNs.

Our goal in this work is twofold. We study the use of a specific attack against BNNs, called Efficient and Exact Verification (EEV) from the work of Kai Jia and Martin Rinard [6], and evaluate if properties linked to the robustness of classical neural networks can be applied to their binarized version.

The second and more important part is our proposal for an algorithm to perform adversarial training, which we call the robust training loop. Through our experiments on images, we will see the different possibilities to evaluate the robustness property for different architectures of BNNs. We will also assess its evolution with the application of the robust training loop.

This paper is organized as follows. We begin with a brief look and explanation of the robustness of DNNs before looking at BNNs, their principle and existing elements related to this property of interest. We will detail the robust training loop algorithm before using it in different experiments. An important part of this paper is focused on the different metrics to evaluate the robustness and its evolution with our proposed training algorithm.

2 Background

In this section, we will focus on the robustness of classical neural networks as well as the concepts behind the Binarized Neural Networks and existing approaches to verify them.

2.1 Robustness

The definition of robustness that we will consider in this work is the ability of a neural network to resist perturbations applied to the inputs. In this context, resisting means that, for a given input, the final

output of the model will stay the same, if the applied disturbance is not too large. We will limit ourselves to images as inputs.

A key component when talking about the robustness of neural networks is the notion of adversarial images [20]. Given an original image, its adversarial version is an image that should be close to the original one with the addition of perturbation that makes the model fail at its classification task. Ideally, these perturbations should not be visible to the human eye or, at least, a human should still be able to classify the adversarial sample correctly. The amount of perturbation per pixel lies in general in $[0, 1]$ and the maximum allowed perturbation per pixel is denoted by ϵ in this work.

An important property of such images is their transferability [17]. An adversarial image is created from two elements: an original image and a model that we are trying to fool. The transferability characterizes the fact that an adversarial image has the possibility to remain adversarial when given to another model other than the one for which it has been created. It allows us to reuse a certain amount of these images and not generate new ones for each model. Of course, it is expected that the value of the perturbation used to create the adversary will play an important role in this property. The higher it is, the more likely the image will be transferable, but the less useful the image may be with respect to ensuring robustness: it may be less problematic for a network to misclassify examples that are highly disturbed than to misclassify examples that are only disturbed a little.

The same concept of adversarial samples also exists for tabular data [1]. However, in our case, we will focus on images since the tool used to generate adversarial samples, EEVBNN [7], was developed to handle images.

To assess the robustness of a model, the typical method is to evaluate the adversarial accuracy of the considered neural network [13]. In this case, the robustness is evaluated by computing the accuracy of a model on adversarial images. It is assessed separately from the original performance, i.e. the accuracy over the original images, to avoid any case of overfitting. This method of evaluation is convenient as it can easily be compared and interpreted with the classical results.

Another way to assess the robustness of a model, but also to obtain the adversarial images described before, is to attack the deep network. Many different approaches for classical networks exist in the literature. We present here the most commonly used, the Fast Gradient Sign Method (FGSM) [4], to give insights on how they work.

Basically, when given an image x with its label y during the training of the model, the backpropagation algorithm will compute the gradient of the loss function $J(\theta, x, y)$ where θ represents the current parameters of the model. From this gradient value we obtain $\eta = \epsilon \ sign(\nabla_x J(\theta, x, y))$. Here, ϵ is a multiplicative value corresponding to the applied perturbation. This computed value η corresponds to the full perturbation that is added to the original image in order to obtain the adversarial image.

The use of an attack such as FGSM can also be used to assess the robustness of a model. In this case, robustness can be determined by the number of successful attacks, i.e. the number of adversarial images that can be found. Therefore, the fewer successes the attacks have, the more our model can be considered robust.

With the help of these adversarial images, we are now interested in making our model more robust. Such process is called adversarial training and has been explored in many publications for classical neural networks. It is in general very dependent on the attack considered. If we take back the example of the FGSM, the corresponding adversarial training would use the following loss function: $\tilde{J}(\theta, x, y) = \alpha \ J(\theta, x, y) + (1 - \alpha) \ J(\theta, x + \epsilon \ sign(\nabla_x J(\theta, x, y)), y)$. Here, α is used to prefer classical training (corresponding to the first term) or adversarial training (corresponding to the second term). For information, Goodfellow et al. used $\alpha = 0.5$ in their work [4]. The use of this specific loss function allows us to indicate to the model that there are adversarial examples in the current training set.

The corresponding learning phase with this function thus uses a mixture of both original and adversarial images. The latter are generated with the previously described attack thanks to the computation of the gradient and then used during the training directly.

In addition to adversarial training, different factors can also impact the robustness of a classical neural network. In their work, Rozsa et al. [17] have shown that models with good performance on original images are more likely to achieve better robustness than other networks. They also demonstrated that the adversaries produced on such high-performing models can turn out to be more transferable. Another important factor to reach good results regarding robustness is the architecture of the neural network. Madry et al. [13] have highlighted the importance of this structure with different tests. They stated that smaller models may not adapt their parameters correctly to handle adversarial images correctly, resulting in a choice between original and adversarial images to perform the learning phase. As before, this architecture factor impacts also the produced adversaries as the samples produced on bigger models tend to be more transferable than the others.

2.2 Binarized Neural Networks

A weakness of classical neural networks is the complexity of the calculations required to perform a prediction, and the resulting energy consumption. An extremely quantized version has been proposed known as Binarized Neural Networks [3] to address this challenge. The principle of BNNs lies in imposing that all the weights and activation values are binary instead of real. Depending on the implementation, the binary values can either be 0 and 1 (the version we consider in this paper) or -1 and 1. However, it is worth noticing that the inputs and outputs of the binary model are the only elements that are not quantized.

To obtain these binary values, BNNs are composed of multiple binary blocks, giving the binary values throughout the whole network. Each block is composed of three precise operations: first, a linear layout, combining the binary weights and activation values of the previous layer, followed by a batch normalization layer, said to reduce the impact of the scale of the weights, and finally, the activation function which must be the sign function to obtain a binary value as output.

An important issue arises with the use of the sign activation function: its derivative will be 0 almost everywhere, introducing problems to update the parameters in the training phase. A solution, as explained in a recent review [22], is to use an approximation of this function during the learning period. The classical and most simple choice is to split the function $sign(x)$ in three parts: x between -1 and 1, -1 for $x < -1$ and 1 for $x > 1$. We insist on the fact that this approximation is only used during the backward pass and that the classical sign function is used at inference time.

Binarized neural networks present interesting characteristics, especially when they are used on dedicated hardware. Thanks to their binary values, the memory needed to store the trained models is reduced as the values can be stored in a single bit. The time and power consumption required to train and use the binary models can also be drastically reduced when compared to classical neural networks

[16]. The gain for these elements is even more impressive when we use the quantized networks on dedicated hardware such as FPGAs [12]. Moreover, it has been demonstrated in many different works that BNNs are able to reach accuracies competitive with classical neural networks on different common datasets [2, 22]. Another interesting advantage of BNNs when it comes to robustness is their direct link with logic formulae.

2.3 Verification of BNNs

To perform attacks against BNNs and to assess how robust they are, we need another method than the ones used with classical neural networks. Indeed, the use of the sign function as activation and the fact that this function is an approximation during the learning phase induce issues to produce adversaries based on the gradient. In their work, Khalil et al [9] have shown that the classical attacks using the gradient, such as FGSM, may not give the correct ascent direction, thus leading to incorrect adversarial examples.

This issue motivates the need for other methods to assess the robustness of binarized models. Fortunately, different approaches using an encoding of these models have been developed in several publications. The encoding term corresponds to a transformation of our model into a set of formulae with constraints. A dedicated solver can then be used to find a solution to these formulae.

A first example, usable also in the case of classical neural networks, is the use of Satisfiability Modulo Theories (SMT) solvers [8]. In this case, each node value of the network is considered as a variable and the verification query, i.e. the query to create an adversarial example, will act as a set of constraints on these variables. What will interest us in this work is the SAT encoding of binarized models [15]. The SAT formula encodes each element of the binary block presented before with the help of a reified cardinality constraint, presented by Jia and Rinard [7]:

$$y_i = \sum_{j=0}^{m} l_{ij}(x_j) \gtreqless \left[b_i(k^{BN}, W, b^{BN}) \right], \tag{1}$$

where x_j is the input to the binary block, y_i its output, l_{ij} takes a value in $\{0, x, \neg x\}$ depending on the binary weights W_{ij}, the constant $b_i(k^{BN}, W, b^{BN})$ is derived from the binary weights and from k^{BN} and b^{BN}, the constants for the batch normalization of the binary block. The symbol $[\cdot]$ is either a flooring or ceiling operation and the $\gtreqless$ is either a $\geq$ or a $\leq$ depending on the sign of k^{BN}.

With the help of this formula, the BNN is now encoded into a SAT formula which can be solved with a dedicated solver. The one considered in this paper is the MiniSatCS solver, developed by Jia and Rinard [7]. It is a CDCL-based SAT solver [14] that encodes both the model and the given input into a SAT formula with constraints such as the one detailed just before. It has 3 possible outputs: SAT when there exists a possible variable assignment leading to the creation of an adversarial image, UNSAT if no adversarial sample can be created for the given model and original image and Time Limit Exceeded (TLE) if the search did not finish in time according to a given time-out parameter.

3 Problem description and approach

Our goal can be presented simply: with the help of the MiniSatCS solver to generate adversarial images, we aim to obtain robust Binarized Neural Networks. We first provide the design criteria for our approach, followed by our proposed approach.

3.1 Motivations

We seek to develop an adversarial training method to obtain robust BNNs. We call this algorithm the robust training loop. Its design criteria are the following. First of all, the method should be easily generalizable. That is, it should generalize to other attacks than the one used in this work, i.e. using the MiniSatCS solver. Furthermore, it should be simple, that is, it should remain as close as possible to how neural networks are usually trained. Our adversarial training should be iterative. This presents two advantages: it lets us assess the evolution of the robustness at each iteration. It also allows the users to use a stopping criterion as soon as they are satisfied with the score of the robustness of the model. Our final motivation, which is common to all adversarial training, is to improve the robustness according to the considered metrics, but without altering the performance obtained on the original images.

3.2 Robust training loop

The robust training loop is a simple modification of the classical training loop that aims to make a model more robust. The model, in this work a binarized model, is trained classically with a given dataset and evaluated on the corresponding test set. Once done, it is verified, which means we use our attack against it to generate adversarial images. These obtained images are then added to the current training data followed by a shuffle operation in order to avoid any unwanted effect. From there, we simply relaunch our adversarial training algorithm by training the model once again on the mixture of data, both original and adversarial. The corresponding pseudo-code is:

```
for i=0...n_robust do
  model = model_init()
  for j=0...n_training do
    train(model,training_set)
  end for
  adv_score = eval_robustness(model)
  adv_images = generate_adv(training_set,eps,model)
  training_set = training_set + adv_images
end for
```

Here $n_{training}$ denotes the number of epochs used to train our deep learning model, while n_{robust} is the number of times we run the robust loop. We will refer to them as robust epochs in opposition to the classical training epochs.

At the end of each robust epoch, a certain amount of adversarial images are added to the considered training set. In our experiments, we arbitrarily chose to set this value to 500. Therefore, at each robust epoch, the size of the training set is increased by 500 new images. We insist on the fact that they are new, which means there is no repetition inside the adversarial set. Note also that the adversarial images later used in the training are generated from the original training set while the ones used to assess the robustness are created from the test set, which is not seen during the training of the model.

3.3 Approaches

We distinguish two versions of our robust training loop algorithm: the generation approach and the adversarial sampling approach. Both of them implement the "generate_adv" function of the pseudo-code differently. The generation approach sticks to what we explained previously, with a generation of adversarial images on the current model

of interest. It has the advantage to be very specific to the model under study, but its main problem is that the attack can take more and more time at each robust epoch as the model should become more and more robust, and finding counter examples for robust models is more time consuming.

On the other hand, we suggest the adversarial sampling approach, designed to reduce the time of the training itself. Instead of generating adversaries during the training, they are pre-computed before the execution of the algorithm on another model that may or may not have been adversarially trained. Thus, we have a set of adversarial images that can be used, thanks to the transferability property, on the model of interest, trained with the robust training loop. In this case, adversarial images are sampled from this pre-computed set of images instead of generated for the current model during the adversarial training. Thus, the main disadvantage is that it is less specific than the generation approach and that all the adversarial images may not keep their adversarial property when transferred to another model.

One difference between the two approaches is important enough to be mentioned more clearly: the more robust epochs we have, the less the adversarial images produced by the generation approach will look like the ones used in the adversarial sampling approach. Even if the differences may not be visible to the human eye, these images are not built on the same information of the model. In one case, the adversarial sampling one, all the images are generated on the same model with no adversarial training. In the other case, the images are produced on models that are more and more robust. However, this difference should not be too notable to directly reject one of the two approaches. Note however that both approaches will be evaluated on a test set generated the same way as the images used in the adversarial sampling, i.e. created on a model trained classically.

4 Experiments

In this main section, we share the results of our experiments. First of all, to understand our models and our data, we perform a set of initial experiments evaluating the following questions: what is the performance of our BNNs on the original images? Are a significant amount of adversarial images transferable also between our BNN models? Can we derive the same conclusion on the importance of the architecture as with classical neural networks? Next, we will focus on our robust training loop algorithm. One of the questions that will be at the center of these tests is: how can we assess the robustness? Is one option better than the other? Concerning the algorithm, is one of the approaches, generation or adversarial sampling, better than the other to improve the robustness of our BNNs?

4.1 Setup

Let us first describe the datasets, parameters and the architectures of the models used in our evaluation. We used the MNIST [10], Fashion-MNIST [21] and CIFAR-10 [5] datasets. For the MiniSatCS solver, we used a time-out value of 90 seconds.

We used four different architectures of BNNs, which we refer to as type. To describe them, we use the term "binary block of size x", where x refers to the size of the linear layer inside the binary block. The "type 1" BNN is made of an input quantizer with a value of 0.1, a flatten operation, two binary blocks of size 512, one of size 100 and the output layer. The "type 2" also has the input quantizer with the same value followed by two binary blocks of size 50 and the output layer. "Type 3" has the input quantizer followed by binary blocks of size 50, 100, 200 and the output layer. Finally, the "type 4" BNN is a convolutional neural network (CNN) with an input quantizer still using 0.1 with a convolutional layer with 32 channels, kernel size of 4, stride of 2 and padding of 1 followed by another convolutional layer with the same parameters but 64 channels. We use then a flatten operation followed by two binary blocks of size 512 and 256 and end with the output layer.

In our following experiments, we use in general 12 robust epochs, 5 training epochs and a majority voting model made of 5 "type 1" BNNs. In order to save time during the training, we limit the number of original images to $10,000$ samples but keep the test set intact. We used a perturbation value of $\epsilon = 0.5$ for MNIST and Fashion-MNIST and $\epsilon = 0.2$ for CIFAR-10 images. These values may appear to be high which is why we will sometimes use $\epsilon = 0.2$ to compare the results. In our work, we focused on the results obtained with the MNIST dataset and confirmed them with the two others. Therefore, we show in our experiments the outcomes with these images. The results for other datasets will be made available in an online repository.

4.2 Original accuracy

In the related work, we have seen that the accuracy on the original images may impact the robustness of the model. Thus, we seek to answer the first question on the scores of our BNNs. In this case only, we have trained our models, a majority voting made of 5 BNNs of the same type, on the whole training set of each indicated dataset. Table 1 reports the different scores for the accuracy on the test set.

Table 1. Results (in %) of the majority voting BNNs on the different test sets

Dataset	Type 1	Type 2	Type 3	Type 4
MNIST	97.87	91.24	96.17	98.65
FASHION	88.18	78.19	87.36	90.89
CIFAR	52.78	43.03	49.77	64.99

According to the literature and if the property is valid for the BNNs, we should expect good results for the robustness of the types 1 and 4. Indeed, they show better performance than the two smaller architectures, i.e. types 2 and 3. Results for the CIFAR dataset appear to be low in this case. A dedicated approach with bigger or different architecture for the used BNNs may be necessary to improve them. It is likely that robust models will be difficult to obtain for this dataset due to this result.

4.3 Transferability analysis

Here, we answer the question related to the transferability property of our adversarial images.

For the adversarial sampling approach it is important that images are transferable. As we can see in Figure 1, a high perturbation value ensures that a large proportion of images created on large models such as the type 1 BNN will tend to keep more of their adversarial property. Of course, a trade-off has to be found between the value of the perturbation, linked to the transferability, and the utility of the image during the training of the model. Indeed, an image with too much modification may not be interesting to use during the learning phase, as even without that instance in the training set the algorithm would not misclassify the instance.

Using a lower value for the perturbation, as in Figure 2, allows us to conclude the following. For the BNNs of type 1 and 4, the largest models we consider in this work, we can see that the adversarial images produced using these two models, i.e. the first and last columns

in the image, are much stronger adversaries as they tend to keep their property more when passed to smaller models. This proves the importance of the architecture of the models, both in the generation and in the inference parts. Larger models will be more difficult to verify, but, when the attack succeeds, the result can be transferred to more models. Therefore, this shows that this idea of the importance of the architecture from classical neural networks also holds for the binarized versions considered in this work.

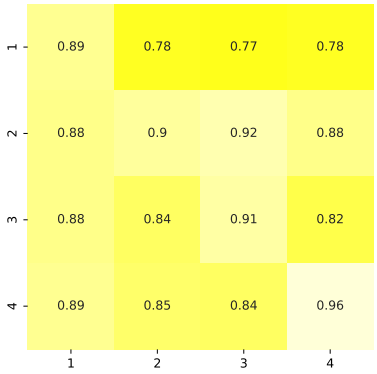

Figure 1. Proportion of transferable images of MNIST for each type of BNNs with a perturbation of 0.5. Square (i, j) corresponds to images generated on model of type j and given to model of type i.

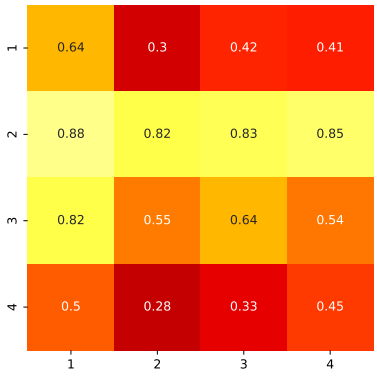

Figure 2. Proportion of transferable images of MNIST for each type of BNNs with a perturbation of 0.2. Square (i, j) corresponds to images generated on model of type j and given to model of type i.

4.4 Robust training loop analysis

Now, we answer the questions related to the robust training loop. First, we compare the results of the two approaches based on adversarial accuracy, i.e. the accuracy of the model on the adversarial images only, displayed in Figure 3.

The first observation is that adding adversarial data to the training set at each robust epoch is able to increase the robustness, according to the considered metric, without a large modification of the accuracy on the original images. Indeed, this score remains almost the same as the one when there is no adversarial training, i.e. at robust epoch 0. Also, both approaches lead to very similar results with this metric. We can notice that the adversarial sampling seems to obtain slightly better results, probably because the images on which we test

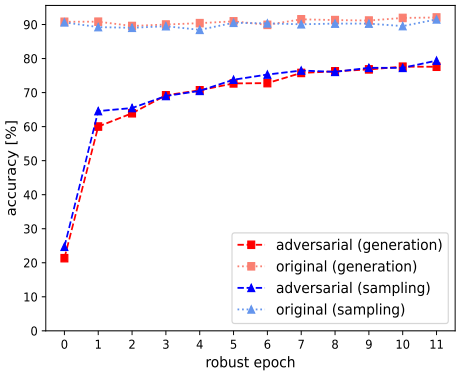

Figure 3. Evolution of accuracy, original and adversarial, for the model described in section 4.1 on the MNIST dataset with perturbation of 0.5.

the model are generated the same way as the ones on which the model has been trained.

Another possible method to assess the robustness is to assess the time needed to generate the 500 adversarial images at each robust epoch. Figure 4 reports the evolution of this value for each robust epoch.

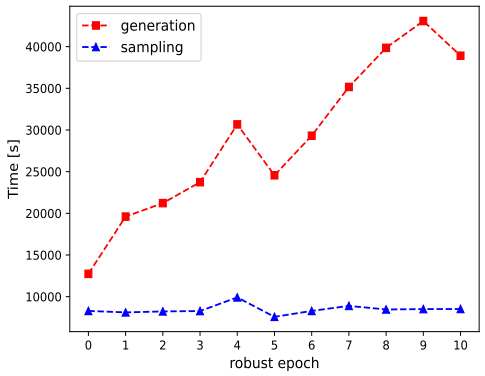

Figure 4. Evolution of the time needed to generate the 500 adversarial images during the robust training loop, on MNIST dataset with a perturbation of 0.5.

In this case, it is clear that the two approaches are different. As the adversarial sampling generates images on another model than the models being trained, it is expected that the time does not vary much for each generation batch. For the generation approach, however, the images are generated on the five models that are currently being adversarially trained. The observed time increase is thus normal and expected. This is another proof that the robustness of this model is indeed improved as the attack has more and more difficulty to find the adversarial examples.

Once the model has been fully trained with the robust training loop, we can assess its robustness by the number of successes of the attack with different perturbation values. In this case, we select 100 images from the test set and use them in the attack to find adversaries. Therefore, the fewer adversarial images it can find, the more robust the model is.

As a baseline, we report the results of this experiment for all the model types without any adversarial training in Figure 5.

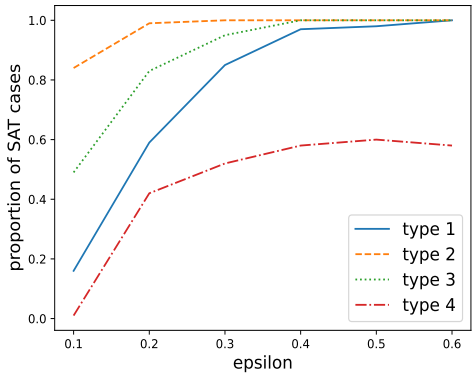

Figure 5. Proportion of success of the attack against the four model types without adversarial training, on MNIST dataset

The first interesting observation that can be made is that the natural resistance of a model is linked to its architecture. We had already found this result when considering the transferability and confirm it once again here: as for their classical version, the BNN architecture has an important impact on the robustness of the model. The other important factor according to the literature is the basic accuracy on the original images, shown previously in Table 1. We can see that, once again, the property of the classical neural networks is confirmed with their binarized version. The best the original performance of a model is, the more robust it will naturally be. Of course, these two properties are strongly linked, which is why it is expected that both of them are found back in this case.

The next step is to apply this experiment, with the same original images, to models that have been adversarially trained. This allows us to assess if our algorithm is able to improve the robustness according to this method of evaluation. We only do this for the model of "type 1" as this model represents a good trade-off between the time (training and verification) and the size of its architecture.

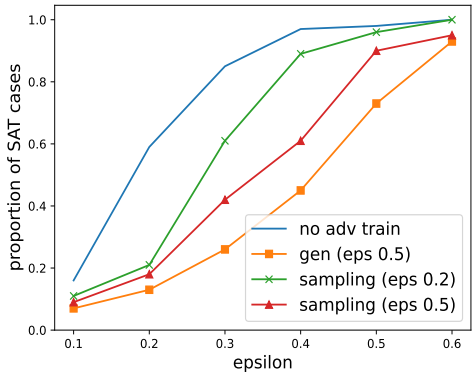

Figure 6. Proportion of success of the attack against the "type 1" BNN after application of the robust training loop, on MNIST dataset

Figure 6 shows the results. Both approaches, generation and adversarial sampling, present better results than the baseline. Moreover, in this case, the generation approach leads to better results than the other, even if we consider different perturbation values, i.e. $\epsilon = 0.2$ and $\epsilon = 0.5$, for the adversarial images used during the robust training loop algorithm. As expected, we can find more adversarial images when the perturbation is high, making the attack easier.

During all our experiments, we generated a lot of adversarial images for the three used datasets. All of them are accessible at our repository[1], along with the code used to perform all these tests.

5 Conclusion and future work

In conclusion, we have shown in this paper multiple elements linked to the robustness of Binarized Neural Networks. We assessed first their basic performance but also studied the transferability of the adversarial images between our models. We presented and tested the robust training loop algorithm, an adversarial training that aims at improving the robustness of BNNs by verifying the model during the learning phase. Even if we did not show it in this work, we believe that this algorithm could be easily generalized to other attacks and deep learning models as it is a simple modification of the training algorithm. Through our various experiments, we assessed the evolution of the robustness with different metrics: the adversarial accuracy, the time taken by the attack and the resistance against the attack. Each provides additional insight into the robustness, but it is difficult to say that one is better than the other, as they evaluate robustness from a different perspective. We also confirmed that two important factors for the robustness of classical neural networks, their original performance and their robustness, are kept for the binarized version considered in this paper. We have also shown that our objectives have been met: we managed to improve the robustness of our models, according to all the considered metrics, without any major impact on the original performance.

However, this does not mean that our models are truly robust, simply that they are more robust than before. Indeed, it is still possible for attacks to succeed at fooling these adversarially trained models. Totally robust models may never be achieved, but the goal to pursue in future work should be to reduce as much as we can the probability that a model is mistaken. To do so, different ways to evaluate the robustness or to generate adversarial samples can be explored. Methods involving GANs or random smoothing may be used and compared with to continue the research on this topic. Additionally, we could also extend this work to other types of data such as tabular data. In this case, the proposed algorithm could stay the same but another generation tool will need to be selected as EEVBNN only deals with images.

Acknowledgements

Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region.

References

[1] Vincent Ballet, Xavier Renard, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki, 'Imperceptible adversarial attacks on tabular data', *arXiv preprint arXiv:1911.03274*, (2019).

[2] Joseph Bethge, Haojin Yang, Marvin Bornstein, and Christoph Meinel, 'Back to simplicity: How to train accurate bnns from scratch?', *arXiv preprint arXiv:1906.08637*, (2019).

[3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, 'Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1', *arXiv preprint arXiv:1602.02830*, (2016).

[1] https://github.com/bronval/Assessing-and-improving-robustness-of-BNNs

[4] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, 'Explaining and harnessing adversarial examples', *arXiv preprint arXiv:1412.6572*, (2014).

[5] Tien Ho-Phuoc, 'Cifar10 to compare visual recognition performance between deep neural networks and humans', *arXiv preprint arXiv:1811.07270*, (2018).

[6] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, 'Binarized neural networks', *Advances in neural information processing systems*, **29**, (2016).

[7] Kai Jia and Martin Rinard, 'Efficient exact verification of binarized neural networks', *Advances in neural information processing systems*, **33**, 1782–1795, (2020).

[8] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer, 'Reluplex: An efficient smt solver for verifying deep neural networks', in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pp. 97–117. Springer, (2017).

[9] Elias B Khalil, Amrita Gupta, and Bistra Dilkina, 'Combinatorial attacks on binarized neural networks', *arXiv preprint arXiv:1810.03538*, (2018).

[10] Yann LeCun, 'The mnist database of handwritten digits', *http://yann.lecun. com/exdb/mnist/*, (1998).

[11] Xuhong Li, Haoyi Xiong, Xingjian Li, Xuanyu Wu, Xiao Zhang, Ji Liu, Jiang Bian, and Dejing Dou, 'Interpretable deep learning: Interpretation, interpretability, trustworthiness, and beyond', *Knowledge and Information Systems*, **64**(12), 3197–3234, (2022).

[12] Shuang Liang, Shouyi Yin, Leibo Liu, Wayne Luk, and Shaojun Wei, 'Fp-bnn: Binarized neural network on fpga', *Neurocomputing*, **275**, 1072–1086, (2018).

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, 'Towards deep learning models resistant to adversarial attacks', *arXiv preprint arXiv:1706.06083*, (2017).

[14] Joao Marques-Silva, Inês Lynce, and Sharad Malik, 'Conflict-driven clause learning sat solvers', in *Handbook of satisfiability*, 133–182, IOS press, (2021).

[15] Nina Narodytska, Shiva Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh, 'Verifying properties of binarized deep neural networks', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, (2018).

[16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, 'Xnor-net: Imagenet classification using binary convolutional neural networks', in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV*, pp. 525–542. Springer, (2016).

[17] Andras Rozsa, Manuel Günther, and Terrance E Boult, 'Are accuracy and robustness correlated', in *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pp. 227–232. IEEE, (2016).

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research*, **15**(1), 1929–1958, (2014).

[19] Emma Strubell, Ananya Ganesh, and Andrew McCallum, 'Energy and policy considerations for deep learning in nlp', *arXiv preprint arXiv:1906.02243*, (2019).

[20] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, 'Intriguing properties of neural networks', *arXiv preprint arXiv:1312.6199*, (2013).

[21] Han Xiao, Kashif Rasul, and Roland Vollgraf, 'Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms', *arXiv preprint arXiv:1708.07747*, (2017).

[22] Chunyu Yuan and Sos S Agaian, 'A comprehensive review of binary neural network', *Artificial Intelligence Review*, 1–65, (2023).

Appendix

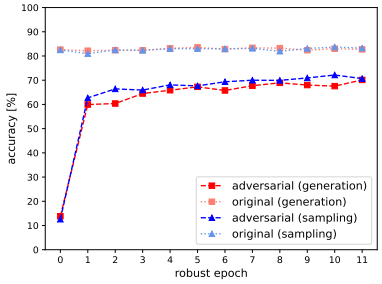

Figure 7. Robust training loop algorithm on Fashion-MNIST dataset with $\epsilon = 0.5$ with the same model than the one in the paper

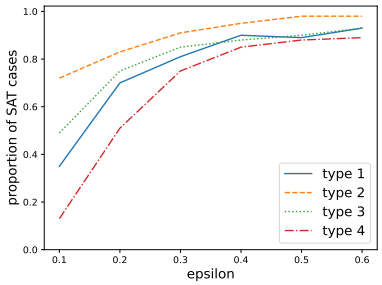

Figure 8. Proportion of success of the attack against the "type 1" BNN without adversarial training, on Fashion-MNIST dataset

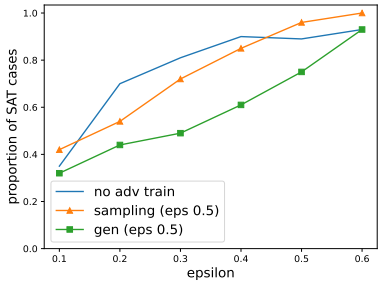

Figure 9. Proportion of success of the attack against the "type 1" BNN after application of the robust training loop, on Fashion-MNIST dataset

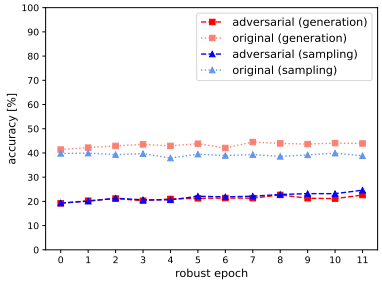

Figure 10. Robust training loop algorithm on CIFAR-10 dataset with $\epsilon = 0.1$ with the same model than the one in the paper

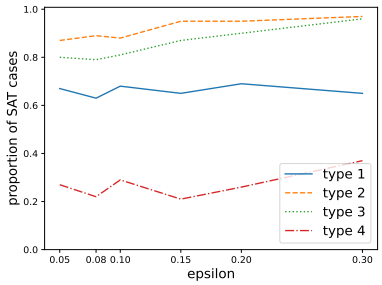

Figure 11. Proportion of success of the attack against the "type 1" BNN without adversarial training, on CIFAR-10 dataset

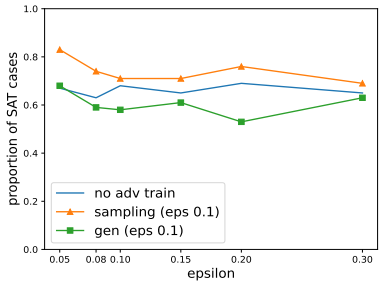

Figure 12. Proportion of success of the attack against the "type 1" BNN after application of the robust training loop, on CIFAR-10 dataset

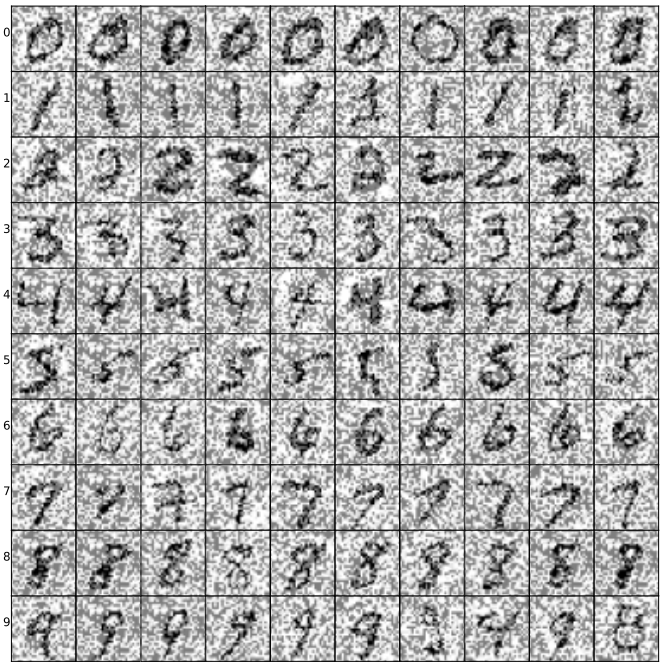

Figure 13. Adversarial images from MNIST with $\epsilon = 0.5$



Figure 14. Adversarial images from MNIST with $\epsilon = 0.2$

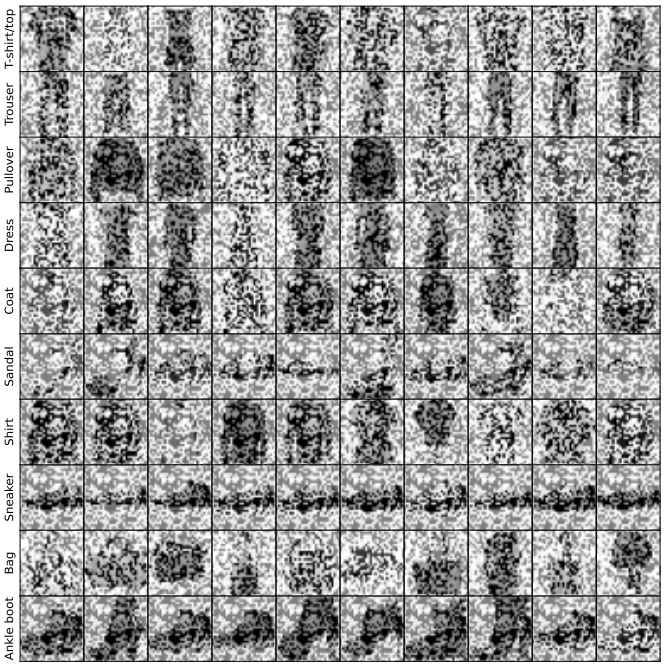

Figure 15. Adversarial images from Fashion-MNIST with $\epsilon = 0.5$

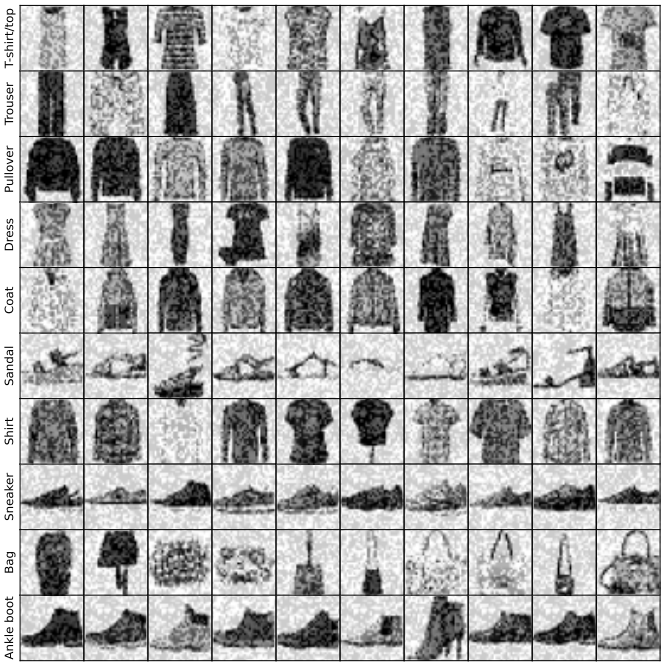

Figure 16. Adversarial images from Fashion-MNIST with $\epsilon = 0.2$

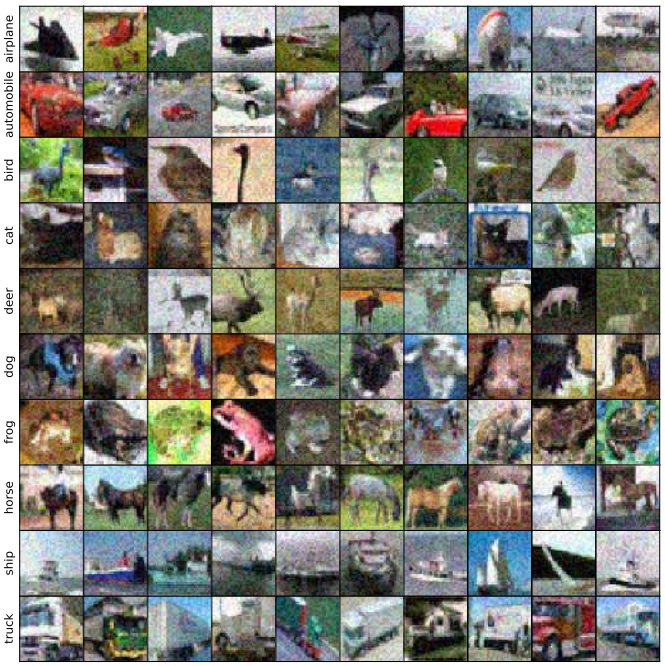

Figure 17. Adversarial images from CIFAR-10 with $\epsilon = 0.1$