Dauphine | PSL
UNIVERSITÉ PARIS

MASTER IASD

DATA SCIENCE PROJECT

Bayesian adversarial training to improve the robustness of Machine Learning models against white-box attacks

Realised by:

BENCHEIKH LEHOCINE Mohammed Amine

DJECTA Hibat Errahmen

KHEDIM Ibtissem

Eureka Team

Supervised by:

Mr. NEGREVERGNE Benjamin

Mr. VÉRINE Alexandre

2022/2023

Table of Contents

List of Figures

1 Introduction

In the last few decades, deep neural networks have become increasingly popular due to the success of deep learning models in a variety of applications, including image and speech recognition, natural language processing, and even playing games. This kind of ML model attracts a lot of attention because of its ability to produce more accurate models while less attention has been paid to the security and robustness of these models. Attackers are increasingly using the tools provided by ML, to improve their attack capabilities. Adversarial attacks are one way to do this. In the latter, the input data is manipulated so that the model makes incorrect predictions.

Adversarial training is a method for training machine learning models to improve their robustness and resilience to adversarial attacks. In this work, we implemented three types of attacks: FGSM, PGD and CW. and we used adversarial training to improve our model against these attacks. Also, we have implemented Bayesian neural networks to test them in the same context (the architecture of the first model is what did the profs propose in the lecture).

This report summarizes our work and it is divided into three parts. In the first part, we will present the adversarial attacks. In the second part, we will show the adversarial training. The last part deals with Bayesian adversarial training. All our tests have been performed on Cifar dataset.

2 Adversarial Attacks

2.1 Basic concepts

An adversarial attack is a type of attack on machine learning models in which an attacker deliberately crafts input data that is designed to confuse a model. These attacks are particularly concerning in security-critical applications, such as image or speech recognition systems, because they can cause a model to make incorrect or even malicious decisions. There are many ways to create adversarial examples, but most involve adding small, carefully chosen perturbations to the input data that are imperceptible to humans but cause the model to misclassify the data. Adversarial attacks can be a serious problem for machine learning systems, as they can be used to bypass security measures or cause a system to behave in unexpected ways [QHWZ22].

There are several different types of adversarial attacks, including:

- **White box attacks:** is a type of adversarial attack in which the attacker has access to the internal structure and parameters of a machine learning model and attempts to find an input that will cause the model to make a mistake. White-box attacks are considered to be stronger than black-box attacks, as they can take the model's internal structure into account.
- **Black box attacks :** are a type of adversarial attack in which the attacker has no access to the internal structure or parameters of a machine learning model. These attacks can be effective if the model is sensitive to small perturbations in the input data, and are often implemented using "query-based" techniques, in which the attacker submits many input samples to the model. Black-box attacks are considered to be less powerful than white-box attacks, which have access to the model's internal structure and gradients.

We give a brief overview of common adversarial attacks, including white-box and black-box attacks. We do not categorize them by target and non-target attacks. As the vast majority of approaches can be easily implemented for both target and non-target settings [QHWZ22].

2.2 FGSM

FGSM (Fast Gradient Sign Method) [GSS14] is a simple, fast and effective method to generate adversarial images. This adversarial image is used to cheat a pretrained model and change his

prediction. The algorithm adds small perturbation to an image that will fool the classifier, hoping to increase the classification loss. Furthermore, this perturbation must be not visible to the human eye, as is illustrated in the Figure 1.

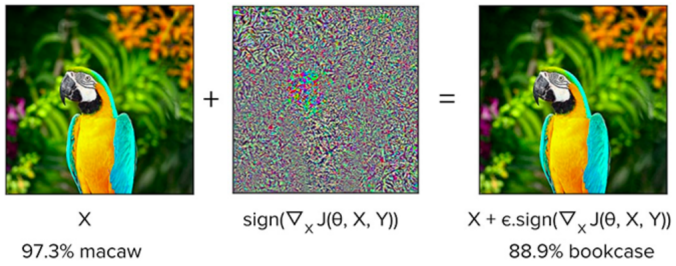

Figure 1: Example of adversarial image generated by FGSM attack

More specifically, given a parameterized network model f_θ with parameters θ, a loss function J, a sample example (x,y), FGSM consists of adding a pixel-wide perturbation ϵ of magnitude in the direction of the gradient to original data. The algorithm 1 illustrates the FGSM attack.

Algorithm 1 Fast Gradient Sign Method attack

Require: f_θ,J,ϵ,x,y
$gradient \leftarrow \nabla_\theta(J(f_\theta(x), y)$
$x^{adv} \leftarrow x + \epsilon sign(gradient)$
return x^{adv}

FGSM is considered as a white box attack since the attacker has access to the pretrained model. So the first step of the attack is using the model to take the true prediction, then computing the loss of the prediction based on the true class label. After, calculate the gradients of the loss with respect to the input image and the sign of the gradient. Finally, the sign of the gradient multiplied by a small epsilon is added to the original input and gives the new adversarial example.

In the first stage of this project, we started by implementing the FGSM attack.Our experimentation was based on varying the epsilon used to generate the adversarial examples and checking the performance of the original model on the adversarial examples. The results of our experimentation of our FGSM attack on CIFAR10 dataset is given on Figure 2. In the first hand, we observed

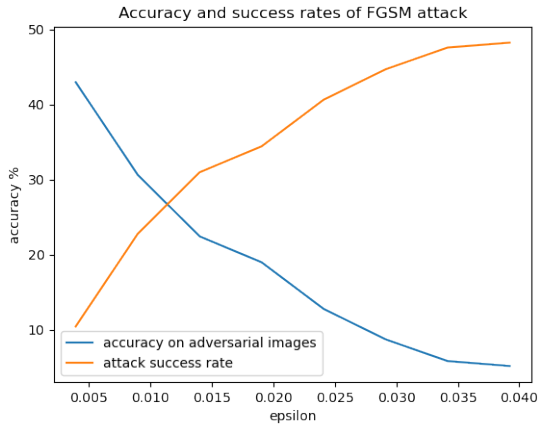

Figure 2: Accuracy and attack success rate of the model in terms of epsilon after FGSM attack

that as the epsilon increases the performance of the original model, represented by the accuracy, decreases. In other words, our adversarial examples succeed to fool the model, and it could be not well classified.In the other hand, we observed that that increasing the value of epsilon to an improvement of the attack success rate values, which is the portion of successful attacks obtained over the dataset used for the evolution process.

2.3 PGD attack

Projected gradient descent (PGD) attack is a method for crafting adversarial examples for machine learning models. It works by iteratively adjusting the input data in a way that maximizes the loss of the model, until the input data is misclassified. PGD attacks can be used to craft both white box and black box attacks, depending on the information that is available to the attacker. They are relatively efficient and easy to implement, but can be relatively easy to defend against [MMS+18].

At a high level, PGD attacks involve iteratively modifying an input in a way that is intended to cause a machine learning model to make a mistake. Mathematically, this can be expressed as an optimization problem, where the objective is to find an adversarial example that maximizes the model's loss.

More formally, let's say we have a machine learning model with parameters θ and an input x that is intended to be classified as class y. The model's loss function L(x, y, θ) measures the difference between the model's prediction and the correct label y. The goal of a PGD attack is to find an adversarial example x' such that the model's loss is maximized. This can be expressed as the following optimization problem:

$$\max L(x', y, \theta) \tag{1}$$

subject to x' = x

Here, x' is the adversarial example and x is the original input. The constraint x' = x means that the adversarial example should be similar to the original input, typically in terms of some measure of distance, such as L2 distance.

To solve this optimization problem, the attacker can use an iterative optimization algorithm such as projected gradient descent. At each iteration, the algorithm computes the gradients of the loss function with respect to the input and modifies the adversarial example in the direction that maximizes the loss. The algorithm continues iterating until the model classifies the adversarial example as the target class or until a maximum number of iterations has been reached [MMS+18].

Projection constraints are used in projected gradient descent (PGD) attacks to limit the size of the perturbation that is applied to the input data. There are many different types of projection constraints that can be used, depending on the specific requirements of the attack. Here are a few examples of projection constraints that are commonly used in PGD attacks:

- **L2 projection:** This type of projection constraint limits the size of the perturbation by imposing an L2 (Euclidean) norm on the perturbation as we can see in the figure 3 . Specifically, the constraint is often written as:

  $$\|x\|_2 \leq \epsilon \tag{2}$$

  where $\|x\|_2$ is the L2 norm of the perturbation and ϵ is a hyperparameter that controls the maximum size of the perturbation.

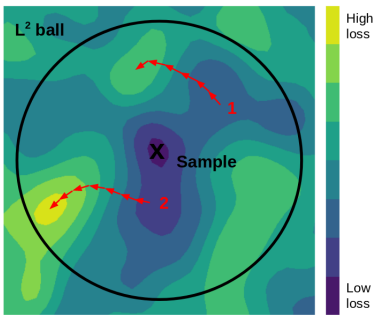

Figure 3: L2 Projection ball

- L^∞ **projection:** This type of projection constraint limits the size of the perturbation by imposing an L^∞ (maximum) norm on the perturbation. Specifically, the constraint is often written as:

$$\|x\|_\infty \leq \epsilon \tag{3}$$

where $\|x\|_\infty$ is the L^∞ norm of the perturbation and ϵ is a hyperparameter that controls the maximum size of the perturbation.

These are just a few examples of projection constraints that can be used in PGD attacks. There are many other types of projection constraints that have been proposed in the literature, and new ones are constantly being developed.

2.4 CW

Unlike the attacks mentioned above, Carlini and Wagner attack [CW17] aims not only to misclassify an adversarial example but also target a specific class. This feature is not the only reason that makes this attack more effective and robust, Carlini and Wagner attack is considered a state-of-the-art attack method, and it is used as a benchmark for evaluating defenses against the same type of attacks. The setup of attack is represented in the following optimization problem:

$$\begin{aligned} \min \quad & D(x, x+\epsilon) \\ \text{s.t.} \quad & C(x+\epsilon) = t \\ & x+\epsilon \in [0,1]^n \end{aligned} \tag{4}$$

This problem minimizing the distance between the original image and the adversarial image x+epsilon, and it could be when of L_P distances (L_0,L_2,or L_∞). The first constraint represents the fact that the classifier C must classify this adversarial image x+epsilon as the target class t, and the second constraint makes sure that the adversarial image lies within the normalized dimensions of x.

One of the optimization problem mentioned that the first constraint is difficult to solve because C $(x+\epsilon) = t$ is highly non-linear. Thus, Carlini and Wagner Attack Algorithm gives another reformulation to the problem: C (x + ϵ) = t is satisfied if and only if there is new function f such that f(x+ϵ)$\leq$ 0

$$C(x+\epsilon) = t \iff f(x+\epsilon) \leq 0 \tag{5}$$

This is just a simple glimpse of Carlini and Wagner Attack and the general advice given in the paper to evaluate any new defense mechanism is to evaluate it against strong attacks like this attack and not others. Furthermore, adversarial examples are classified with a higher confidence than real images.

3 Adversarial Training

The issues related to this adversary-examples' vulnerability expose fundamental blind spots of the supervised learning paradigm and stand against the adoption of deep learning in critical and safety domains of application. We'll explore in this section the adversarial training defense mechanism, and then we will present a variant of this defense based on Bayesian neural networks.

3.1 Basic idea

Adversarial training consists of creating and then incorporate adversarial examples into the model training loop [WRK20]. It was first proposed by [GSS14]. They optimized a loss function defined as a weighted average of the original loss value computed on the original samples and its value on perturbed samples.

Given a parameterized network model f_θ with parameters θ, a dataset $S = \{(x_i, y_i)\}$ and a loss function l, we define a *min-max* optimization problem for adversarial training [Mad23] :

$$\min_\theta \frac{1}{|S|} \sum_i \max_{\|\delta\| \leq \epsilon} l(f(x_i + \delta), y_i)$$

The outer minimization problem refers to 'original' empirical risk minimization formula in supervised context. The maximization term refers to the addition of the worst perturbation to each example x_i.

A simple and intuitive way of solving the min-max optimization is by using one of the white-box attacks we have already seen in previous sections and use it as a generator for adversarial examples. We apply then any variant of gradients updates algorithm to minimize the empirical risk.

Algorithm 2 Adversarial training algorithm using an adversary attack

N size of dataset D, T number of epochs
for $t = 1$ .. T **do**
  $B \leftarrow MiniBatch(D)$
  $g \leftarrow 0$
  for $(x_i, y_i) \in B$ **do**
    $\tilde{x_i} \leftarrow attack(f_\theta, x_i, y_i)$
    $g \leftarrow g + \nabla_\theta l(f_\theta(\tilde{x_i}), y_i)$
  end for
  update models parameters θ
end for

3.2 Experimental settings and results

To test the effectiveness of adversarial training as a defense mechanism against adversarial attacks, We trained three models each one has its architecture: the Basic architecture proposed during thr lecture, a ResNet and a VGG classification model using the two different training processes (adversarial and ordinary) and compared the accuracy of the model given crafted malicious samples from $L_\infty PGD$ and $L_2 PGD$ attacks.

First, we used to train the basic network against a PGD adversary with L^∞ projected gradient descent and ϵ size of 0.0314. After that we have tested the trained model against LinfPGD and L2PGD attack with different ϵ values and we have recorded the adversarial accuracy and the success rate. The results are shown in figure 4 For LinfPGD and as expected, when using a smaller

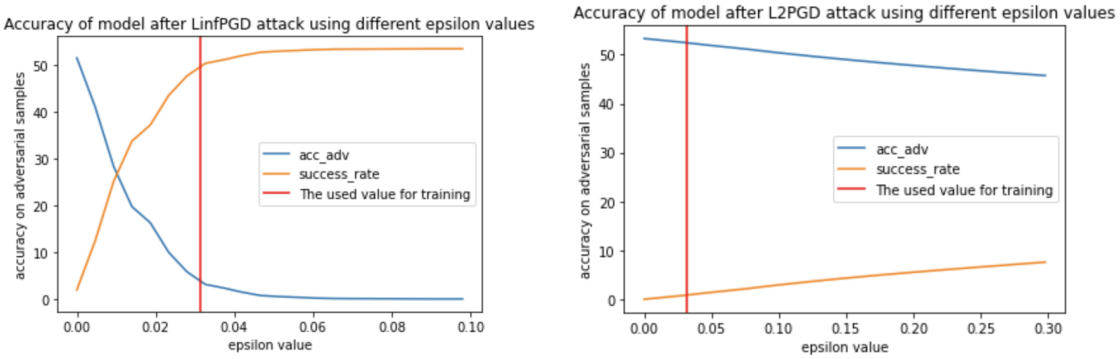

Figure 4: Accuracy of model after LinfPGD/L2PGD attack using different epsilon values

value of ϵ during evaluation, the models show equal or higher accuracy compared to when using the value of ϵ that was used during training. As the value of ϵ increases, the success rate also increases, and once ϵ reaches the value that was used during training, the success rate plateaus. Concerning

L2PGD, we can notice that the model is more robust against this type of attack and for large values of ϵ. Since our model is trained on Linf PGD. This latter is typically more robust to noise and adversarial perturbations, because it constrains the maximum change to any individual parameter rather than the sum of all changes. Furthermore, we found that a Resnet model produced similar results to those previously obtained, but with higher accuracy values. This can be seen in Figure 8

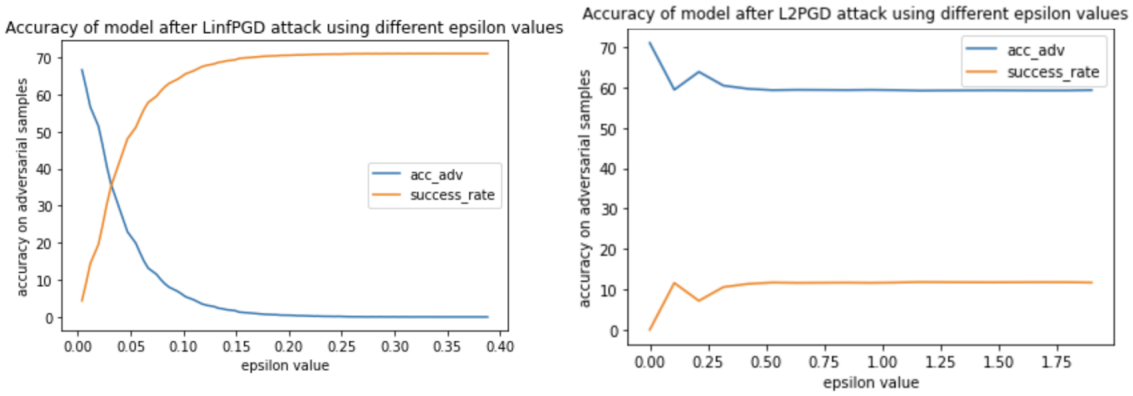

Figure 5: Accuracy of model after LinfPGD/L2PGD attack using different epsilon values for a model based on ResNet Architecture

On the other hand, we trained a VGG for the same purpose. We used $L_\infty PGD$ attack inside the adversarial training loop with $\epsilon = 0.01$ and 10 steps. We used exactly the same hyperparameters listed in the paper [LLWH18]

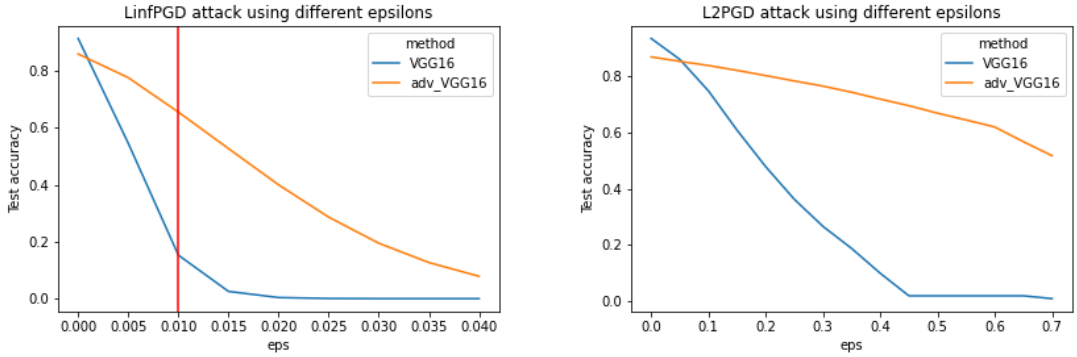

Figure 6: Accuracy of trained models under LinfPGD/L2PGD attack using different distortion values for a model based on VGG16 Architecture

Although the accuracy of the model trained with adversarial training drops with nearly 10% for natural samples (without distortion), It resists effectively to perturbations caused by both $L_\infty PGD$ and $L_2 PGD$ attacks. The model became then somewhat resistant to adversarial examples, whereas the model trained with vanilla training process gets high error rates (87%) for relatively small distortions.

The problem of high confidence didn't disappear; unfortunately, the new model still misclassifies some adversarial examples with high confidence. [GSS14] explains this by the fact that the adversarial training process is somehow an active learning strategy that labels noisy points (adversary samples) with their nearest neighbors' labels from the original samples set. It doesn't change then the way the model learn the concepts/patterns of the data, but just incorporate some randomness into the training data. Other works had suggested incorporating randomness in the models layers, like in [LCZH17] and [LLWH18]. We'll try to use Bayesian neural networks as randomness incorporation mechanism as defense against white-box attacks.

4 Bayesian adversarial Training

4.1 Bayesian neural networks

Classical neural networks are generally described as universal functions approximators ; they use a combination of weights and biases to approximate an unknown function using a gradient-based optimization process to find the most likely values of estimation to their parameters. Although their effectiveness in a variety of tasks, they still vulnerable to overfitting and adversary attacks as explained in previous sections. Bayesian neural networks stands as an alternative to these models; the main idea is to incorporate Bayes inference in the optimization process. Instead of fitting the most likely values for the weights, BNNs try to fit for each weight its posterior distribution $P(w|\{x_i, y_i|0 \leq i \leq N\})$. Exact Bayes inference is intractable [BCKW15], they use in the literature approximate inference (variational inference like in Variational autoencoders, Markov chain Monte-Carlo sampling, Stochastic Gradient Langevin Dynamic like in Bayes by back propagation [BCKW15])

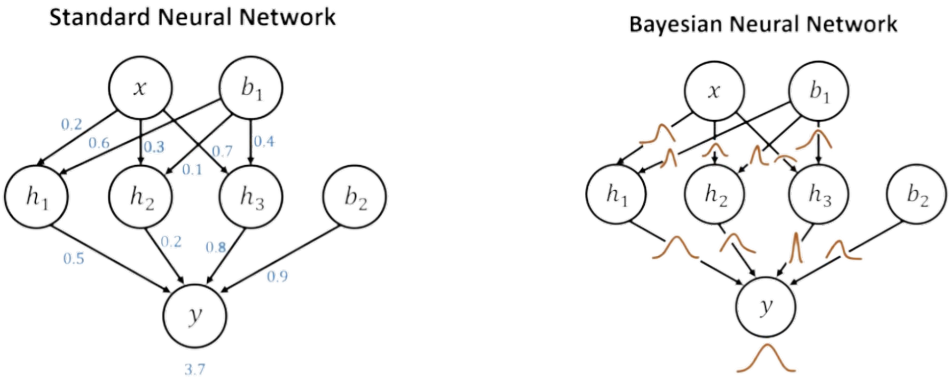

Figure 7: For standard neural nets, each weight has a single value referred as a point estimation. For Bayesian neural nets, Each weight is represented by an optimal distribution.

4.1.1 Implementation

In our experiment, we used an adaptation of the three layers : **Linear** layer, **Batch Normalization** layer and **Conv2D** layer to the Bayesian inference mode. We inspired from [LLWH18] then we constructed an equivalent model of VGG architecture using the adaption of these three layers. Another possibility was to use the **BLiTZ** library [Esp20] to convert the whole model into its Bayesian equivalent.

4.2 Adversarial Bayesian training

We combined the randomization of the model by Bayesian inference and input randomization by adversarial training to get better accuracy under attacks. We applied the same procedure explained in the adversarial training to the Bayesian neural network, the same attack $L_\infty PGD$ with same hyperparameters. For the model parameters updates, we refer to [LLWH18] . Each weight update is replaced by its distribution parameters updates (namely the mean and variance of the Gaussian distribution from which we sample the weight value during the forward pass).

4.3 Experimental settings and results

We trained four models in total : the VGG16 model alone with no defense, the adversary trained VGG16, the VGG16 adaptation to Bayesian neural networks and the VGG16 Bayesian adaptation

with adversarial training. We used $L_\infty PGD$ as attack with $\epsilon = 0.01$ and 10 steps. The results are in the following figures.

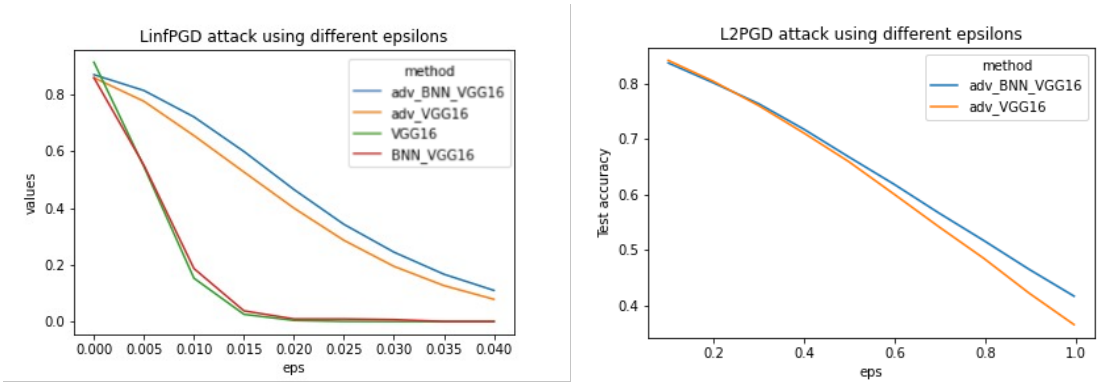

Figure 8: on the left, comparing the accuracy of all four models Under $L_\infty PGD$ attack, on the right, we compared only Adversary trained BNN and adversary trained VGG16

The two figures show that using only Bayesian neural networks didn't make any noticeable improvement. Both BNN_VGG16 and VGG16 suffer from the same vulnerability of adversary attacks. When we train the BNN_VGG16 using adversarial training, its resistance to attacks augments compared to adversarial training alone. its accuracy curve is above all other models' curves. For $L_2 PGD$ attack, the difference between adv_VGG16 and adv_BNN_VGG16 is not remarkable.

4.3.1 Discussion

We had seen that combining adversarial training with Bayesian inference gives more robustness to machine learning models. Although the improvements are not big (compared to the costs we paid in terms of memory storage : we need the double space to store the weights of BNNs, and the computation time which took longer than for classical model), more amelioration can be apported notably in terms of the approximation method used for the variational inference (stills an open problem that faces the application of Bayesian neural nets in real world problems) and the architecture of model ; the work of [UCMH21] demonstrates the use of Dense Net gives better results then VGG16.

Another approach could be the use of the uncertainty estimation given by Bayesian inference to detect malicious samples and avoid the whole task of resisting adversary attacks.

5 Conclusion

In this study, the Fast Sign Gradient (FSGM) and Projected Gradient Descent (PGD) methods were applied to VGG and ResNet image classification networks trained on the CIFAR10 dataset to create adversarial examples, and the effectiveness of adversarial training and Bayesian networks as defense methods were evaluated. The results showed that the Bayesian adversarial training approach, which combines adversarial training with Bayesian networks, was the most effective at defending against the adversarial attacks. These findings suggest that combining adversarial training with Bayesian networks may be a promising approach for improving the robustness of neural networks against adversarial attacks. It is important to note, however, that these results are specific to the VGG network and CIFAR10 dataset used in the study, and further research is needed to understand the generalizability of these findings.

References

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *https://doi.org/10.48550/arXiv.1505.05424*, 2015.

[CW17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[Esp20] Piero Esposito. Blitz - bayesian layers in torch zoo (a bayesian deep learing library for torch). https://github.com/piEsposito/blitz-bayesian-deep-learning/, 2020.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[LCZH17] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho Hsieh. Towards robust neural networks via random self-ensemble. *https://doi.org/10.48550/arXiv.1712.00673*, 2017.

[LLWH18] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. *https://doi.org/10.48550/arXiv.1810.01279.*, 2018.

[Mad23] Z.K.and A. Madry. adversarial training, solving the outer minimization, adversarial robustness—theory and practice, (Accessed: January 3, 2023).

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. 2018.

[QHWZ22] Zhuang Qian, Kaizhu Huang, Qiufeng Wang, and Xu-Yao Zhang. A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies. 03 2022.

[UCMH21] Adaku Uchendu, Daniel Campoy, Christopher Menart, and Alexandra Hildenbrandt. Robustness of bayesian neural networks to white-box adversarial attacks. *arXiv https://doi.org/10.48550/arXiv.2111.08591*, 2021.

[WRK20] Eric Wong, Leslie Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.