BIOLOGICALLY INSPIRED SLEEP ALGORITHM FOR INCREASED GENERALIZATION AND ADVERSARIAL ROBUSTNESS IN DEEP NEURAL NETWORKS

Sanjeevan Sritharan ss6n17, 29649404
Suyash Dubey sdd1n17, 29533414
Syed Fathir afs1n17, 29624428

ABSTRACT

This report attempts to reproduce the work of Tadros et al. (2020) which involves introducing a sleep phase into a standard Artificial Neural Network (ANN). The rationale for this modification is to make the network more robust against adversarial attacks. An ANN with the sleep phase implemented is then compared against other benchmark models to evaluate its efficiency.

1 INTRODUCTION

In this report, the group focuses on reproducing recent research introduced at the International Conference on Learning Representations (ICLR) 2020. The paper discusses the benefit of adding a "sleep" phase into an ANN to defend against adversarial attacks as well as increase its classification soundness. The rationale for the modification is inspired by human biology, based on the hypothesis that sleep is a reason why humans are so capable at classification tasks despite the presence of noise and/or other distortions.

This report aims to reproduce the aforementioned research and evaluate its methodology, comparing our results with those of the original authors, highlighting any discrepancies where present. The code and the report can be found at https://github.com/COMP6248-Reproducibility-Challenge/Biologically-Inspired-Sleep-Algorithm.

This research was reimplemented using Pytorch v1.8.1 and Python 3.7.10 as well as the Torchbearer framework v0.5.3. All aspects of the project were performed using a GPU on Google Colab.

Considering factors of time and space in the report, the group decided to only work on the MNIST dataset, and exclude the Patches and the CUB-200 datasets. For the adversarial attacks, we implemented distortions in the form of blur and noise, as well as the Fast Gradient Sign Method (FGSM) attack. We also implemented defensive distillation and fine-tuning to compare the sleep algorithm with other existing adversarial defenses.

2 IMPLEMENTATION

2.1 CONTROL ARTIFICIAL NEURAL NETWORK

This paper utilises a normally trained ANN (Artificial Neural Network) for several reasons. Firstly, it is used as a control network against which to test the other networks which are specialised for defending against adversarial attacks. Secondly, this ANN is used as the base network for creating both the SNN (Spiking Neural Network) and the fine-tuned network.

The original paper provides the parameters for training this ANN using SGD, such as the learning rate, momentum and dropout probability, as well as the architecture and number of epochs to train. However, the dropout probability given was a single floating point number, with no indication as to which layers this dropout was applied to. As such, it was decided that we would only implement this for the first hidden layer, as through experiments this seemed to give the smallest running loss during training.

Furthermore, the ANN is trained on only 27105 training images. This is because the original paper mentions both: (1) that the sleep algorithm is run using the same training data as the original ANN is trained with, and (2) the sleep algorithm is run using 27105 images. From these, it can be surmised that the training set for the control ANN consists of 27105 images.

2.2 Distortion Networks

For adversarial retraining or fine-tuning, the Control ANN is retrained on blurred and noisy training data with a lower learning rate. Adversarial retraining can be considered as a form of regularisation. The training images are blurred using a Gaussian filter with standard deviation ranging from 0 to 2.5 in steps of 0.5. Gaussian noise is added to the training images by sampling from a normal distribution with mean as 0 and a range of variances 0, 0.1, 0.3, 0.5, 0.7, 0.9. The noise added in the paper is not described in detail due to which our implementation of noisy images are less noisy in comparison.

2.3 Sleep Algorithm Artificial Neural Network

As mentioned in the previous section, the SNN to which the sleep algorithm is applied is based on the control ANN. The procedure is as follows.

Firstly, the trained ANN is converted into a SNN by mapping weights directly. Implementation-wise, this meant creating a model with the exact same weights as the control ANN. In this new model representing the SNN, the input passed into each input neuron is a spike train, or a series of 1's and 0's. The rate of 1's depends on the original input pixel value.

At the start of the sleep algorithm, each neuron in the network (except the input neurons) are given a voltage, initialised to 0. As explained above, a single training image is converted into 784 spike trains. The first value of each spike train is then passed into the corresponding input neuron. The values are propagated across the network such that the voltage of each neuron is:

$$V = V + \Sigma_i(w_i \cdot S_i)$$

where w_i are the weights of all connections to that neuron from some neuron i in the previous layer, and S_i is the spike coming from i, with value 1, or no spike, of value 0. Given this voltage update, the new voltage value is then compared to a threshold voltage. Each layer has a threshold voltage, given in the original paper as a parameter for the sleep algorithm. If the new voltage value is above the threshold for that neuron's layer, the neuron outputs a 1 to the next layer, and its voltage is reset to 0. If not, it outputs a 0. This way, spikes are propagated across the network. Once activity has propagated to the output layer, the next set of 784 1's and 0's is passed in, and this is done for all the values in all the spike trains, and the entire process is done for all training images. As per the original paper, the voltage values for the neurons are only initialised at the start of the algorithm, and not with each new training image.

The weights of the model are updated as follows: everytime a neuron outputs 1, it checks all of the neurons in the previous layer and the inputs it received from each one. If the previous layer neuron had given an input of 1, the weight between the neurons is increased by some amount. If the input was 0, it decreases the weight by some amount. Both amounts are given as parameters in the original paper.

It was found that the threshold parameters for each layer given in the original paper led to very little spiking (1's being output) during sleep, and so we had to change them.

Also of note is that the original paper did not mention anything about the voltage of a neuron being reset to 0 after firing a spike. However, relevant literature on integrate-and-fire neurons (Abbott, 1999) indicated that it made sense to reset the voltage.

2.4 Defensive Distillation Network

Defensive Distillation is an adversarial training technique which makes an algorithm's classification more robust and less susceptible to exploitation (DeepAI). Defensive Distillation allows the model to generalise better to samples outside its training dataset using networks connected by a mentor-student relationship where instead of predicting binary labels, probabilities generated by the student network are output instead (Papernot et al., 2016). The process includes a softmax function with a temperature parameter (T) which the paper sets to 50.

2.5 FGSM Attack

The paper has conducted FGSM as one of its adversarial attacks which computes the sign of the gradient of the loss function (J) with respect to the original input (x) using the weights θ of the network and the target labels (y). The result is an output image which looks identical to the human eye as the original, but leads the network to make an incorrect prediction. We used the MNIST test set and performed the FGSM attack to validate each network.

3 Results and Evaluation

3.1 Blur and Noise Distortions

The MNIST test set was blurred using a Gaussian filter with standard deviation ranging from 0 to 2.5 in steps of 0.5. The same standard deviation was used to evaluate the accuracy of each network before it was increased.

For the blurred test set, the accuracy of the defensive distillation network was the best and that of the sleep network was the worst when the standard deviation is 0. As the standard deviation increases, the Defensive Distillation network continues to perform the best. However, with the highest standard deviation of 2.5, the fine-tune blur network, which was trained on the blurred training data, has the highest accuracy. The accuracies for the fine-tune noise network and the control ANN with standard deviation 2.5 are 75.86% and 77.6%. The sleep network is the least accurate for the blur distortion, whereas the fine-tune blur network is the most accurate.

Noise was added to the test set images by sampling from a normal distribution with mean 0 and increasing variance 0, 0.1, 0.3, 0.5, 0.7, 0.9. The same variance was used to evaluate each network's accuracy before it was increased.

For the noisy test set, the accuracy of the defensive distillation network was the best and that of the sleep network was the worst for all variance values. As the variance is increased, the defensive distillation network continues to perform the best. With the highest variance of 0.9, the defensive distillation network has the best accuracy of 94.59%, followed by the fine-tune noise network which has an accuracy of 89.58%, the control network with an accuracy of 77.36%, the fine-tune blur network has an accuracy of 72.3%, and the sleep network has the worst accuracy of 8.91%. The defensive distillation network is the most accurate for noise distortion and the sleep network is least accurate.

The sleep network does not seem to be robust to distortions when compared to other networks, contradicting the results of the original paper.

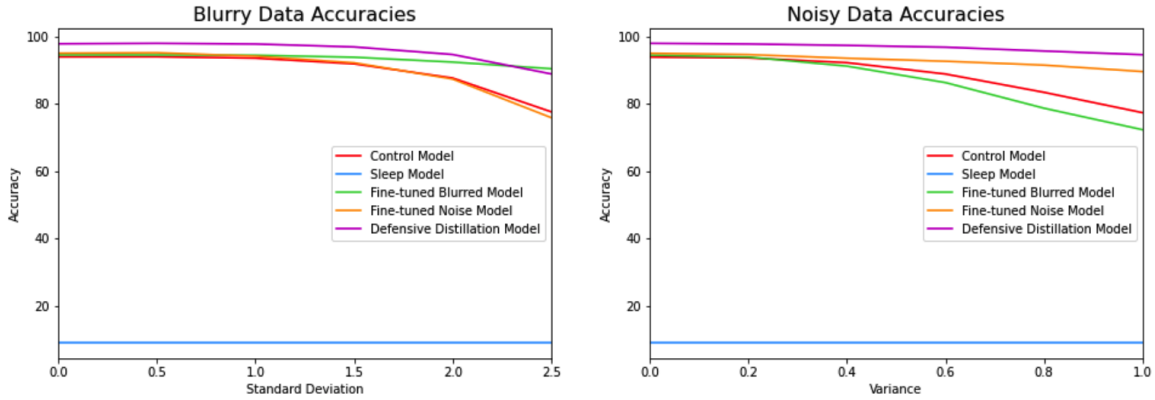

Figure 1: Accuracies of Models for Blurred and Noisy Data

3.2 FGSM Attack

We have simulated an FGSM attack on the MNIST test set, with a varying epsilon parameter increasing from 0 to 0.4 in steps of 0.05 as shown in the paper. Each of the networks were evaluated with the same epsilon value before it was increased.

While most of the networks started out strong, their accuracies were severely affected by the increasing epsilon values, with all but the sleep network falling to roughly 1% or less. The sleep network on the other hand, despite starting weakly at an accuracy of 8.92%, remained constant despite the increase in epsilon for an end result of 8.93%. Apart from this network, we found that defensive distillation starts out the strongest at 97.87% and ends the strongest among the other networks at 1.16% whereas the others ended close to 0% with an epsilon value of 0.4. Given that the intent of the paper was to create a network which could generalise well despite attacks, we feel that the sleep network has adhered to that hypothesis in this case.

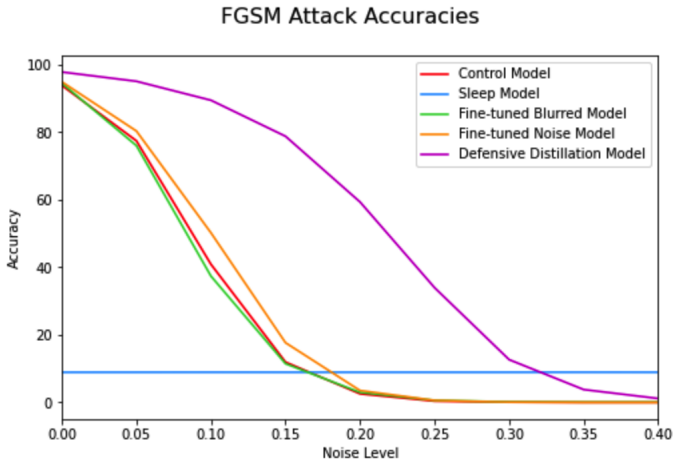

Figure 2: Accuracies of Models for FGSM Attack

4 Conclusion

The paper does not include resetting the voltage of integrate-and-fire neurons. This was changed in our implementation as this would lead to continuous firing of neurons which have surpassed their thresholds. Over time, all neurons would eventually spike continuously, and nothing of importance would be learned or reinforced in the network during the sleep phase.

Furthermore, no clear instruction was given on how weight normalisation would take place when converting the ANN to SNN, as there are various ways of normalising weights. Finally, the original threshold parameters given in the paper were extremely high, leading to very little spiking activity during sleep, and no change to the weights. We had to arbitrarily set the thresholds ourselves based on the observed spiking activity.

As can be seen from our results, the sleep network is not as accurate for distortions compared to other networks as the paper suggests. Whereas in the FGSM, despite the initial low accuracy, the accuracy remains similar with the maximum epsilon value, making it robust.

This report could be extended by examining the accuracy of sleep, fine-tuned, and defensive distillation networks on other adversarial attacks and the Patches and CUB-200 datasets mentioned in the original paper.

References

L. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50:303–304, 1999.

DeepAI. Defensive distillation. *DeepAI*. URL https://deepai.org/machine-learning-glossary-and-terms/defensive-distillation.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. pp. 582–597, 2016.

Timothy Tadros, Giri Krishnan, Ramyaa Ramyaa, and Maxim Bazhenov. Biologically inspired sleep algorithm for increased generalization and adversarial robustness in deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1xGnA4Kvr.