Project Report: An Empirical Study of a Defense Against Bit-flipping Attacks on Neural Networks

Daniel Saragih
daniel.saragih@mail.utoronto.ca

Paridhi Goel
paridhi.goel@mail.utoronto.ca

Tejas Balaji
tejas.balaji@mail.utoronto.ca

Alyssa Li
alyssamengyuan.li@mail.utoronto.ca

1 Introduction

The development of deep neural networks (DNN) has ushered a new era of security and safety applications, but also new vulnerabilities. One such concern is the threat of bit flipping attacks (BFAs), in which an adversary can tamper with critical model parameters, and negatively influence the accuracy of prediction drastically. Recognizing the possibility of being attacked in this way, a defense mechanism called Aegis [1], develops a dynamic-exit strategy by attaching extra internal classifiers (ICs) to hidden layers (Figure 1), but also perform robustness training (ROB) on the model by simulating BFAs during the fine-tuning of ICs (to be discussed in Section 2).

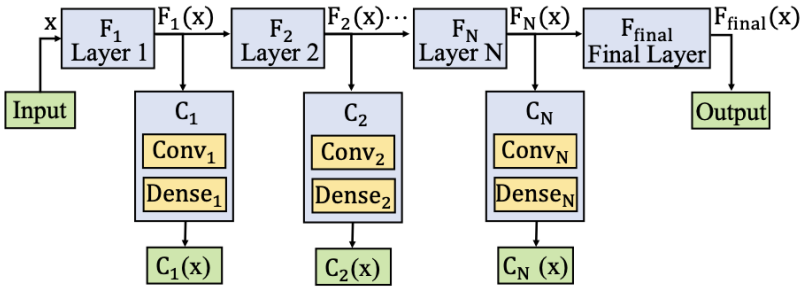

Figure 1: Base model (e.g. **Resnet32** or **VGG16**) equipped with internal classifiers $(C_1, \ldots, C_N)$ for early exiting; figure retrieved from the original paper [1].

Motivated by the importance of mitigating such attacks on machine learning models, in this project, we conducted various tests on the Aegis framework. Specifically, our experiments were concerned with:

1. Evaluating the test accuracy, accuracy on perturbed test data, and vulnerability to attack when using low-entropy data such as MNIST.
2. Evaluating the test accuracy, accuracy on perturbed test data, and vulnerability to attack when trained on a general dataset like CIFAR10 and then fine-tuned[1] on MNIST.
3. Evaluating the vulnerability of Aegis on other adversarial attacks such as the generation of adversarial examples [2].

The code for this project is available on GitHub[2].

[1] Fine-tuning, in our case, is slightly different from the conventional definition; see Section 2.
[2] `https://github.com/paridhi26/CSC413-project`

2 Background and Related Work

The threat of BFAs on deep learning models started with DeepHammer [3] which aimed to reproduce the well-known Rowhammer attacks on DNNs. Defenses against the attack generally fall into the categories of model augmentation or integrity verification [1]; the defense that we will focus on falls in the former category. For instance, we have defenses such as BIN [4] and RA-BNN [5] which aims to binarize the model parameters, and in doing so increase the necessary bit flips to disrupt model performance. On the other hand, ModelShield [6] uses hash verification to verify the integrity of model parameters.

The work conducted by the authors of the Aegis paper propose a defense mechanism which is "non intrusive, platform independent, and utility preserving" [1]. The goal of the authors for developing a mechanism with these properties arose from the issues and difficulties encountered with previous defense mechanisms, such as having to retrain the entire model or having to change the parameters in the model. As such, Aegis only involves attaching ICs to the original model. Moreover, the training run only involves training the ICs while freezing the base model parameters; this procedure is what we refer to as *fine-tuning*.

In light of its goal, Aegis adopts two defensive mechanisms. The first involves a dynamic-exit shallow-deep network (DESDN) where samples randomly exit the network early with a uniform probability distribution on the layers. Using the convention of Figure 1, we sample q candidates from a uniform distribution over $(C_1, \ldots, C_N)$. We then exit from the first of the candidates which exceed a certain confidence (defined as the highest probability over target classes) threshold τ. Thus, when attackers attempt to target particular layers of the network, they will find it extremely difficult to pinpoint a set of layers that are likely to have the most negative effect on the prediction if attacked. This contrasts with a network without any early exists wherein the most vulnerable bits are those closest to the output classifier.

A second mechanism is robustness training (ROB) wherein the authors deliberately flipped vulnerable bits and trained the early exit classifiers using these features (Figure 2). To do so, we take a copy of the clean model M to make a flipped model $\widehat{M}$. The vulnerable bits are identified through a gradient ranking algorithm. In particular, define $\mathcal{L} = L_{CE}(F_{final}(x), t)$ where L_{CE} is the cross-entropy loss and t is the ground-truth label. If we denote the parameter bits of $\widehat{M}$ as B, for each $b \in B$, calculate $\nabla_b \mathcal{L}$. Rank the gradients $\nabla_b \mathcal{L}$, and choose the top-k such bits to flip (where k is the attack budget). During training we pass the training data and label through M and $\widehat{M}$, and propagate the sum of their losses through the shared ICs.

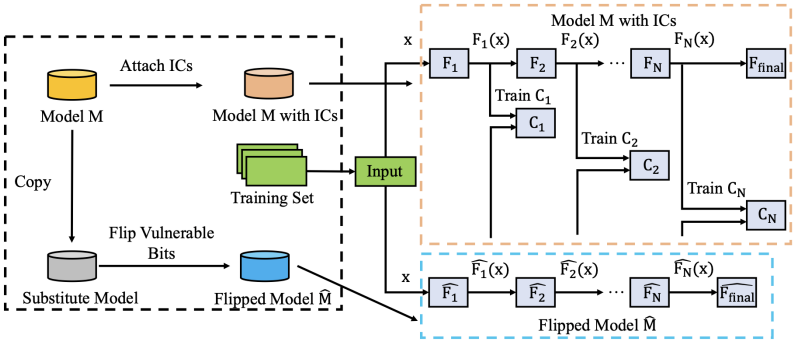

Figure 2: The ROB mechanism. We create a copy of the model, $\hat{M}$, referred to as the flipped model, and flip vulnerable bits of $\hat{M}$ to simulate an attack. Figure retrieved from the original paper [1].

3 Methods

3.1 Base model training

The focus of our methods is on studying the Aegis mechanisms. To start, as in the original paper, we will use **ResNet32** [7] and **VGG16** [8] as the base models. We will train four baseline models: **ResNet32** on MNIST and CIFAR10, referred to as **R-MNIST** and **R-CIFAR** and **VGG16** on MNIST and CIFAR10: **V-MNIST**, **V-CIFAR**. In this initial training run, we will freeze the parameters of the internal classifiers $(C_1, \ldots, C_N)$ and instead output predictions using the main classifier head F_{final}. We run the training loop for 30 epochs on MNIST and 60 epochs on CIFAR. We also use the Adam optimizer with a learning rate of 4×10^{-4}, and a batch size of 128. The output is an array of probabilities over the target classes and the loss used is cross-entropy.

3.2 Fine-tuning

Subsequently, we endow these models with the Aegis ICs and further fine-tune them. In particular, we fine-tune three types of models: with ROB, without ROB, and with augmented data (without ROB). The augmentation strategy is our contribution motivated by the similarity of ROB to data augmentation strategies (i.e. both attempt to patch up under-represented areas of the learned distribution). The augmentation techniques we apply include rotation, translation, scaling, Gaussian blur, and random erasing. These model types will be distinguished with the suffix **-r**, **-nor**, and **-aug** respectively. In this fine-tuning session, we freeze the base model parameters and only allow gradients to change the internal classifier parameters.

For the MNIST models, we fine-tune with the same dataset. However, to accomplish our second objective, for both of our CIFAR models, **R-CIFAR** and **V-CIFAR**, we will perform this fine-tuning step on the MNIST dataset. Thus, a model such as **R-CIFAR-r** is one initially trained on CIFAR and then fine-tuned on MNIST with ROB applied. The hyperparameters are the same as in base model training except we use 30 epochs when fine-tuning *both* MNIST and CIFAR models.

3.3 Evaluation

Prior work [9] has shown that training on adversarial samples in fact reduces the accuracy on real (non-adversarial) data as it dilutes the learned probability distribution. We wish to highlight this effect by performing inference on MNIST data which is relatively low entropy as compared to the datasets used in the original paper. In the same vein, we will perform inference on perturbed data, where the perturbations are label-invariant (e.g. noising, blurring, and slight rotations). Overall, we have three types of evaluation procedures:

1. A baseline evaluation which tests on the fine-tuning dataset while using the early-exit strategy on fine-tuned models.
2. A perturbed evaluation where label-invariant transformations are applied to the test data; again early-exit is used.
3. An adversarial evaluation where we first attack the model with Proflip [10] and observe the attack success rate (ASR).

In each case, we use the MNIST test set of 10000 images. Moreover, we contrast our results with those obtained in the original paper.

An additional experiment that we performed aims to test Aegis' robustness to adversarial samples [2, 11]. As was shown in [9], methods that directly focus on preventing specific attacks often suffer on other, unrelated attacks. We verify this finding on all of our fine-tuned models by reporting the accuracy on the samples produced by Fast Gradient Sign Methods (FGSM) [2] to judge how it fares against non-BFA adversaries. This is a rather simple attack that calculates the gradient with respect to the input data $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{t})$ and transforms the input in the direction of the gradient:

$$\mathbf{x}_{\text{perturbed}} = \mathbf{x} + \epsilon \text{sign} \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{t}) \quad \text{for some } 0 \leq \epsilon < 1.$$

Model	Baseline Accuracy	Perturbed Accuracy
R-CIFAR	84.6	44.3
R-CIFAR-r	97.8	60.6
R-CIFAR-nor	98.9	70.3
R-CIFAR-aug	98.7	92.0
V-CIFAR	89.4	47.4
V-CIFAR-r	90.2	40.6
V-CIFAR-nor	92.9	46.8
V-CIFAR-aug	90.7	70.5
R-MNIST	98.3	72.7
R-MNIST-r	99.3	78.1
R-MNIST-nor	98.7	65.4
R-MNIST-aug	99.2	93.6
V-MNIST	98.7	78.8
V-MNIST-r	98.2	75.1
V-MNIST-nor	98.9	79.9
V-MNIST-aug	99.3	92.1

Table 1: Accuracies for all the models along with accuracies when tested on perturbed test data.

4 Results

4.1 Baseline Evaluation

Table 1 shows the test accuracies we got for the different baseline models when tested on the fine-tuning dataset while using the early-exit strategy on fine-tuned models. All **-r**, **-nor**, **-aug** were finetuned on MNIST and then were evaluated on MNIST. The basic non-finetuned models **R-CIFAR** and **V-CIFAR** were trained on CIFAR and tested on CIFAR. Similarly, the non-finetuned models **R-MNIST** and **V-MNIST** were trained and tested on MNIST.

4.2 Perturbed Data Evaluation

For the perturbed data evaluations, we perturbed the test data by adding some label-invariant transformations to it before testing. These transformations were random rotations, random translations, random scaling, Gaussian Blur, adding Gaussian noise and random erasing.

We evaluated the models on perturbed test data to further pronounce the differences between different finetuning settings (**-r**, **-nor**, **-aug**). The performance on perturbed test examples better demonstrates the ability of model on learn general features of the data in a particular training setting rather than memorizing idiosyncrasies of the training data.

The perturbed accuracy column in Table 1 shows the test accuracies we got for the fine-tuned models (**-r**, **-nor**, **-aug**) when we tested them on the fine-tuning dataset with label-invariant transformations applied to the test data while using the early-exit strategy. It also shows the accuracies we got for the non-finetuned models (**R-MNIST**, **V-MNIST**, **R-CIFAR**, **V-CIFAR**) after similarly testing them on test data with label-invariant transformations applied.

All **-r**, **-nor**, **-aug** were finetuned on MNIST and then were evaluated on MNIST with label-invariant transformations. The basic non-finetuned models **R-CIFAR** and **V-CIFAR** were trained on CIFAR and tested on CIFAR with label-invariant transformations. Similarly, the non-finetuned models **R-MNIST** and **V-MNIST** were trained and tested on MNIST with label-invariant transformations.

As expected, the two base (not finetuned) CIFAR models **R-CIFAR** and **V-CIFAR**, showed the largest drop in accuracy when evaluated on perturbed CIFAR images compared to **R-MNIST** and **V-MNIST** which were evaluated on MNIST which is low entropy. CIFAR images are more complex than MNIST images and perturbation in test data generally affected all CIFAR models more compared to their MNIST counterparts. The only outlier here was **R-MNIST-nor** which showed lower than expected Perturbed accuracy.

We also expected that the **-r** models which went through robustness training during the fine tuning process to perform worse than the **-nor** models which did not go through robustness training since robustness training can dilute the learned probability distribution. This hypothesis was also supported by our evaluation results, with **R-MNIST-nor** being the only exception as above. We were not able to explain why the accuracy of **R-MNIST-nor** was lower than **R-MNIST-r** and why **R-MNIST** was an outlier. This can be further explored in future work.

For the **-aug** fine tuning setting, we added some label-invariant transformations we used to perturb test data also to the training data which was used to fine tune. For **-aug**, the transformation that were applied to the training data while fine tuning were random rotations, random translations, random scaling, Gaussian Blur and random erasing. That is all the transformations from the perturbed test data except Gaussian noise. Since, these models where already fine tuned on perturbed data, their accuracy drop was least significant on the perturbed test data.

4.3 Proflip

In this section, we present results for the Proflip [10] evaluations. The Proflip procedure inserts a backdoor into the target model by flipping parameter bits such that the prediction of all inputs with the set trigger will result in a pre-specified, incorrect target class. The ASR of Proflip is thus the proportion of trigger examples that manage to fool the model into predicting incorrectly.

Table 2 shows the final ASR for all the models which we attacked with ProFlip. As the results indicate, the augmented models tend to have the lowest ASR values (with the exception of the **V-MNIST** models). The next best class of models tends to be the **-r** models, while the **-nor** models are observed to have the highest ASR. Moreover, we tracked the number of exits made in the IC layers for each of the models when attacked with ProFlip. In general, we observed that the **ResNet** models seemed to have more exits in earlier layers as compared to the **VGG** models. The results for number of exits per layer for **R-MNIST** and **V-MNIST** are shown in Figures 3 and 4.

Table 2: Final ASR values for **-r**, **-nor**, and **-aug** models on Proflip

Model	ASR	Model	ASR
R-MNIST-r	22.4	R-CIFAR-r	13.4
R-MNIST-nor	34.9	R-CIFAR-nor	14.9
R-MNIST-aug	11.5	R-CIFAR-aug	12.7
V-MNIST-r	14.5	V-CIFAR-r	14.4
V-MNIST-nor	11.6	V-CIFAR-nor	15.2
V-MNIST-aug	12.0	V-CIFAR-aug	10.6

We expected the -**nor** models to perform the worst out of all the three classes (**-r**, **-aug**, and **-nor**) since they have been trained with the least amount of resistance to bit flip attacks. However, what we do find surprising is that the **-aug** models perform better than the **-r** models in most cases. This confirms our hypothesis that augmentation serves a similar role to ROB. The improvement seen in the case of augmentation likely stems from the fact that label-invariant transformations are more informative for learning than the outputs of a flipped model. This increased generality would allow the model to fare better when it encounters an unfamiliar attack pattern. It is also interesting to note the ASR difference between **R-MNIST** and **R-CIFAR** models. This indicates that the more general features learned through fine-tuning makes the model more robust. However, the same cannot be said of the VGG models, and so more testing is necessary to verify this trend.

As is depicted in Figures 3 and 4, the **R-MNIST** and **R-CIFAR** models exited earlier on average than the **V-MNIST** and **V-CIFAR** models. We believe that one of the causes for this is that ResNet, owing to its name, uses residual connections as compared to VGG, which suffers from a vanishing gradients issue. MNIST being a low-entropy and "easy" dataset means that strong classifiers such as Resnet will be able to easily obtain very high accuracies within just a few layers. On the other hand,

(a) (b)

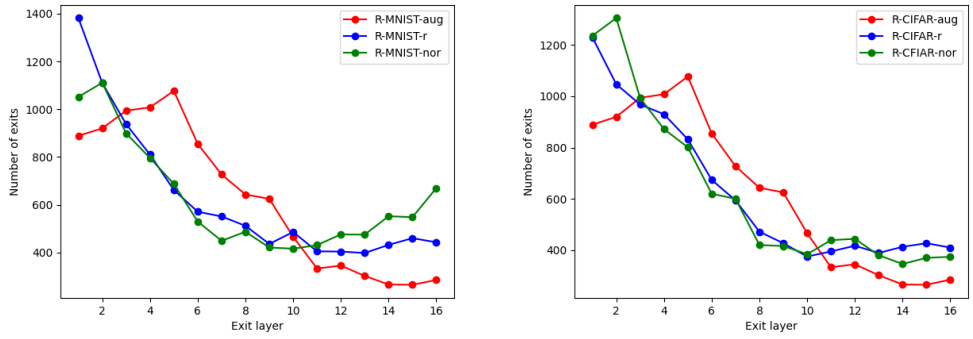

Figure 3: Number of exits per layer for **R-MNIST** and **R-CIFAR** models

(a) (b)

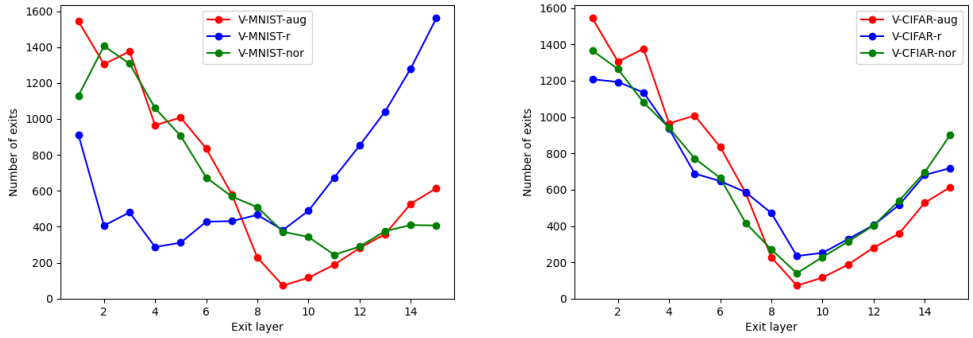

Figure 4: Number of exits per layer for **V-MNIST** and **V-CIFAR** models

the fewer number of layers and the vanishing gradients issue with VGG is likely to negatively impact the amount of time it takes for VGG to correctly classify an MNIST image (as compared to Resnet).

It is also interesting to note the skew towards earlier layers. As mentioned in the methods, DESDN randomly picks a set of q candidate classifiers and then early exits when it encounters the first classifier that achieves an accuracy above a specific threshold. Thus, it is likely that there is a bias towards the earlier layers given that MNIST is an easy dataset to learn. If the modelling problem was in fact more challenging, it's likely that later layers are required to exceed the confidence threshold. In turn, this would provide a counter balance to the early-exit bias. Thus, we may conclude that the uniformity in early-exiting which Aegis shows in the original paper [1] is in fact contingent on a fragile balancing act.

4.4 Adversarial Examples

Table 3 shows the accuracies of the 16 models we trained against various values of ϵ, which denotes the strength of pertubation on input data in FGSM attacks. We observe that the accuracies for the **Resnet** augmented models tend to be the highest among all other **Resnet** models. For the CIFAR models, we see that for $\epsilon = 0.2$, the **R-CIFAR-aug** model achieves an accuracy of 37.6% while **R-CIFAR-nor** and **R-CIFAR-r** achieve less than 15% accuracy. The same goes for the MNIST Resnet models, where **R-MNIST-aug** achieves a very high accuracy of 73.1% on $\epsilon = 0.2$ while the **-r** and **-nor** models achieve accuracies below 20%. As for the **VGG** models, the difference in accuracy doesn't seem to be very high across the models for higher values of ϵ in the case of CIFAR. However,

Model \ Epsilon	0.00	0.05	0.10	0.15	0.20
R-CIFAR	73.8	32.0	16.7	11.1	9.14
R-CIFAR-r	99.0	61.5	24.6	14.9	11.6
R-CIFAR-nor	99.4	89.7	49.3	19.2	13.5
R-CIFAR-aug	99.3	92.7	82.0	62.3	37.6
V-CIFAR	81.6	35.3	14.6	9.28	8.55
V-CIFAR-r	97.5	48.0	12.3	10.1	9.98
V-CIFAR-nor	99.2	75.0	19.9	10.3	9.80
V-CIFAR-aug	99.0	91.3	51.1	13.8	9.82
R-MNIST	98.2	96.7	94.7	73.8	36.9
R-MNIST-r	99.5	97.2	86.0	37.0	10.1
R-MNIST-nor	99.0	94.3	59.0	17.6	10.3
R-MNIST-aug	99.5	97.3	93.4	86.9	73.1
V-MNIST	98.7	96.8	93.3	72.0	41.7
V-MNIST-r	98.9	93.7	76.4	41.4	19.1
V-MNIST-nor	99.0	93.8	70.0	26.1	13.7
V-MNIST-aug	95.1	88.5	87.4	80.5	46.9

Table 3: Accuracies for all the models when tested on adversarial examples. It should be noted again that R/V-CIFAR is evaluated on the CIFAR10 dataset as opposed to MNIST.

(a) (b)

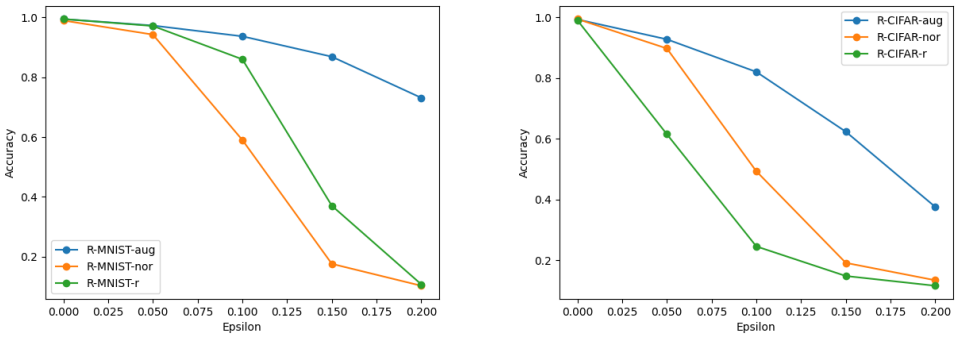

Figure 5: Epsilon vs Accuracy for fine-tuned **R-MNIST** and **R-CIFAR** models.

in the case of MNIST, **V-MNIST-aug** seems to greatly outperform its **-r** and **-nor** counterparts, achieving an accuracy of 46.9% for $\epsilon = 0.2$. We observe moreover that the baseline models tend to outperform the **-r** and **-nor** models, and are (in most cases) second only to the augmented models.

Let a "group" of models denote one the four groups of 4 models as shown in Table 3 (for example, the first group is the group of all **R-CIFAR** models). We observe that in 3 out of the 4 groups, the augmented model vastly outperformed its baseline, **-r**, and **-nor** counterparts. It was only in the case of **V-CIFAR** that all models were observed to have a similar accuracy for $\epsilon = 0.2$. Barring the outlier for **V-CIFAR**, we claim that the results we obtained are to be expected, for the following reasons:

1. The **-nor** models performing worse than the **-aug** models is expected since the **-aug** models have been trained with perturbations of the data in mind. The **-aug** models have been trained to correctly classify noisy and perturbed data, making them robust to a wide range of adversarial inputs. Moreover, the baseline models were expected to outperform the **-nor** and **-r** models, since the latter two models have been trained to specifically combat bit-flip attacks. In particular, we note that both the **-nor** and **-r** models are models that have been

(a) (b)

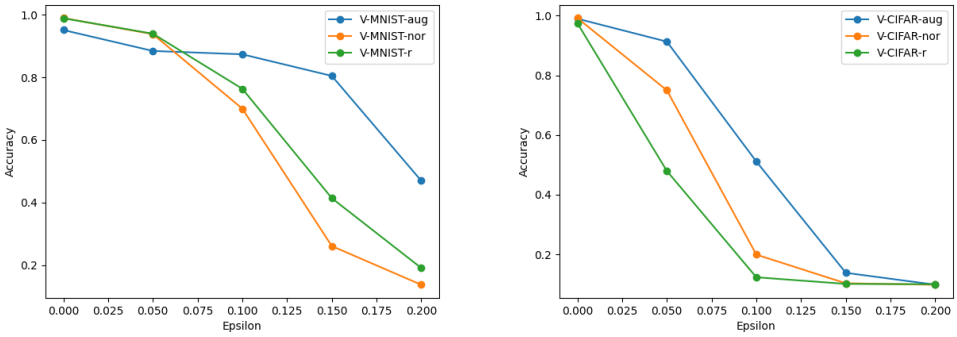

Figure 6: Epsilon vs Accuracy for fine-tuned **V-MNIST** and **V-CIFAR** models.

finetuned on the baseline models for the purpose of preventing bit-flip attacks. This in turn changes the parameters and weights in the classifiers. Thus, a significant portion of robustness that the baseline models possessed to adversarial data was lost when the models were finetuned. In the case of the **-aug** models, although they were trained on top of the **-nor** models and not the baseline, they regained this robustness as a result of the augmented training. We also observe that in two of the four groups, the **-nor** models outperform the **-r** models. In particular, this is observed to be the case in the **R-CIFAR** and **V-CIFAR** model groups. Thus, this unusual behavior seems to be a result of differences between the MNIST and CIFAR datasets and not a result of the baseline models (i.e., **VGG** and **ResNet**). We hypothesize that this is the result of MNIST being a low-entropy dataset as compared to CIFAR, and thus easier to classify.

2. The most intriguing result is that the **-aug** models perform significantly better than the **-r** models. Figures 5 and 6 showcase this stark difference in accuracies between the two models. While the **-r** models have been through robustness training, the training has specifically been for the prevention of bit-flip attacks. The use of adversarial examples is a different mechanism of attacking a neural network, it is thus expected that a model that has not built any resistance for this specific type of attack will prove to be vulnerable. The augmented models, however, are more general in the class of attacks they can resist.

5 Conclusion, Limitations, and Future Work

The results presented indicate that the defensive framework proposed in Aegis possesses significant defensive capabilities as shown by low ASR in Table 2, but also some notable drawbacks. From our experiments, we observed that ROB often stunts the model's ability to learn with sufficient generality. For instance, in Table 1, we see that **-r** models generally performed worse than its counterparts. Moreover, the protection ROB provides against BFAs comes at the cost of other adversarial attacks such as FGSM, as shown by its diminished accuracy in Table 3. In addition, DESDN also has its downsides. For instance, just by evaluating on low-entropy datasets such as MNIST, we are able to obtain a non-uniform exit distribution, contrary to the uniformity showcased in the original paper. This suggests that if an attacker is aware of the type of datasets a model is tested on, they will have an easier time attacking the model.

Our work also has its limitations. For instance, we claim in Section 4.3 that there is a balancing act between the number of layers needed to achieve the confidence threshold and the early-exit bias of DESDN. Although our experiments showcase this effect, more testing on different datasets is required to verify its validity. Therefore, future work should focus on performing further verification of the effects hinted at by our work. An interesting avenue involves conducting a wider array of adversarial attacks on Aegis, or using new methods to generate adversarial samples. And certainly, future work should also be dedicated to finding improvements to Aegis that mitigates some of the issues brought up in our work.

6 Contributions

1. **Daniel Saragih:** The idea of this project was proposed by Daniel, who shared the Aegis paper with all the group members. In the early days of the project, Daniel worked with the other team members to get the codebase running. All the team members had regular meetings to ensure that everyone was on the same page regarding the versions of the packages that needed to be imported and the version of the code we all had on our machine. Daniel trained the four base models (**R-MNIST**, **R-CIFAR**, **V-MNIST**, **V-CIFAR**) and half of the finetuned and augmented models. He was also responsible for all the shell scripts and additional Python files that needed to be created for running the empirical experiments. Furthermore, Daniel helped the team on several occasions when shell scripts were not running or there were changes to be made in Python or shell files for different models. Daniel also requested the CSC413 teaching team for access to GPUs, and gave the team the idea of using screens in the shell to conduct parallel training. The usage of screens saved a lot of time for the entire group. Daniel tested half of the models on ProFlip attacks. Moreover, he assisted in writing a few components of the "Results and Discussion" part of the report and wrote the majority of the "Background and Related Work", "Methods", and "Conclusion" sections of the report.

2. **Tejas Balaji:** As stated earlier, Tejas helped Daniel and the other team members on the first few days of the project in resolving errors in the codebase. Once the base models had been trained, Tejas trained the other half of the finetuned and augmented models. He also ran the adversarial examples on a significant number of the models, and ran experiments using ProFlip for the remaining half of the models. Tejas also coordinated with Paridhi and Alyssa to set up a TA Office Hour towards the latter half of the project to ask about the complexity of the numerical experiments being conducted by the group. This office hour turned out to be quite useful and guided our decision of running experiments on perturbed data and running adversarial examples. This meeting was attended by Tejas and Daniel. Tejas wrote a significant portion of the "Results and Discussion" section of the report for Proflip and for Adversarial Examples. He also generated all of the figures in the report for ProFlip and Adversarial Examples. Lastly, Tejas helped in writing a significant portion of the "Contributions" section of the report.

3. **Paridhi Goel:** Paridhi was also involved in the initial meetings to get the codebase up and running. Paridhi set up the github repository for the project by adding relevant directories from the code provided by the authors of the Aegis paper to the team github repository. Paridhi worked with other teammates to train base models initially. Paridhi conducted the baseline and perturbed evaluations for the fine-tuned models (**-r**, **-nor**, **-aug**) listed in table 1. In the report, Paridhi wrote the Baseline Evaluation and Perturbed Data Evaluation sections. While running these evaluations, Paridhi alerted the group about bugs in the scripts/code, outlier models with suspiciously low accuracies so they can be re-trained and verified.

4. **Alyssa Li:** During the early stages of project, Alyssa worked with the teammates to get codebase running and participated in all the group discussions. Alyssa contributed in running the adversarial examples, moderated the run script sent by Tejas, and ran adversarial tests on multiple models (**R-MNIST-r, R-MNIST-nor, R-CIFAR-r, R-CIFAR-nor,V-MNIST-r, V-MNIST-nor, V-CIFAR-r,V-CIFAR-nor**). She also assisted in exporting numerical results to Google Drive and separated the plots for all the models when tested on adversarial examples for further report usage in Table 2. Alyssa was also involved in writing report - she specifically worked on Section 4.4 (Adervsarial Examples). Moreover, she proofread the entire report and adjusted the format for better illustration.

References

[1] Jialai Wang, Ziyuan Zhang, Meiqi Wang, Han Qiu, Tianwei Zhang, Qi Li, Zongpeng Li, Tao Wei, and Chao Zhang. Aegis: Mitigating targeted bit-flip attacks against deep neural networks, 2023.

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[3] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1463–1480. USENIX Association, August 2020.

[4] Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. Defending and harnessing the bit-flip based adversarial weight attack. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14083–14091, 2020.

[5] Adnan Siraj Rakin, Li Yang, Jingtao Li, Fan Yao, Chaitali Chakrabarti, Yu Cao, Jae sun Seo, and Deliang Fan. Ra-bnn: Constructing robust & accurate binary neural network to simultaneously defend adversarial bit-flip attack and improve accuracy, 2021.

[6] Yanan Guo, Liang Liu, Yueqiang Cheng, Youtao Zhang, and Jun Yang. Modelshield: A generic and portable framework extension for defending bit-flip based adversarial weight attacks. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 559–562, 2021.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[10] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Proflip: Targeted trojan attack with progressive bit flips. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7698–7707, 2021.

[11] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387, 2016.