EL-GY-9163: Machine Learning for Cyber Security
Prof: Siddharth Garg
Final Project Report

Liao Hu (lh2375)

Final Project Report

1 Introduction

In this project, we designed a backdoor detector for 4 Backdoored Deep Neural Networks (BadNets) which were trained on the YouTube Face dataset using identical architecture with N output classes. We were given original BadNets and a clean validation dataset which would not trigger backdoors in BadNets. We repaired these 4 BadNets by using Fine-Pruning [1] method and generated 4 corresponding repaired GoodNets that will correctly classify clean input data and classify backdoored input data to N+1 class. We also explored another approach described in [2] that considered a post-deployment phase that provides real backdoored input which could be utilized to tune and improve the detector. Due to time constraints and project limitations, we only tested and verified this approach but did not evaluate its performance.

2 Background

2.1 BadNets Properties

The 4 BadNets given are backdoored by different triggers, which are different patterns in the input image, and will classify backdoored images to a specific class among a total of 1283 classes. Both of aboves are to some extent, unknown to us, the defender. The following table **Table 1** shows a summary of these 4 BadNets.

BadNet Name	Triggers	Targets
sunglasses_bd	sunglasses	unknown
anonymous_1	unknown	unknown
anonymous_2	unknown	unknown
multi_trigger_multi_target	Sunglasses, eyebrows, lipstick	3 unknowns

Table 1. Properties of BadNets

2.2 BadNets Evaluations

We designed a new evaluation approach to better understand the performance of both the original BadNets and repaired GoodNets.

The two old factors have some significant weaknesses: it will ignore the false positive (classify clean input to backdoored) and false negative (failed to classify a backdoored input). Thus, instead of focusing only on attack success rate (ASR) and Classify Success Rate (CSR), we not only modified these 2 to Attack Triggered Rate (ATR) and added two more criteria: Defend Triggered Rate (DTR) and Classify Failure Rate (CFR).

For backdoored data, Classify Failure refers to those being neither classified to be poisoned nor correctly classified, Classify Success is the sum of Defense Triggered and correctly classified.

BadNet Name	Input Data Type	Attack Triggered Rate (ATR)	Defend Triggered Rate (DTR)	Classify Success Rate (CSR)	Classify Failure Rate (CFR)
sunglasses_bd (B1)	Clean	0.0078%	N/A	97.7786%	0%
	Poisoned	99.9922%	N/A	0.0078%	0%
anonymous_1 (B2)	Clean	1.2627%	N/A	97.1863%	1.5511%
	Poisoned	91.3971%	N/A	0%	8.6029%
anonymous_2 (B3)	Clean	0%	N/A	95.9625%	4.0374%
	Poisoned	unknown	N/A	unknown	unknown
multi_trigger_multi_target (B4)	Clean		N/A		
	Poisoned		N/A		

Table 2. Evaluation of BadNets

Due to time constraints, this detailed evaluation will only be performed on some specific critical points, such as threshold tuning and final evaluation.

3 Implementation

As mentioned in the introduction section, we used two approaches to repair BadNets. The first approach implements the method introduced in [1], which uses neuron pruning and fine tuning on the existing BadNet. The second approach explores the method in [2], which contains a pre-deployment phase that uses noise to isolate suspicious input, and post-deployment phase that collects quarantined inputs and uses CycleGan to reproduce backdoored features on clean data and eventually re-trained BadNets. However, due to time constraints and given the fact that backdoored data is not allowed to be used in this project, we only tested and verified this approach but did not evaluate its performance.

3.1 Fine-Pruning

As discussed in [1], the main purpose of pruning is to eliminate neurons that are not activated for clean data, which are likely to be activated for poisoned data. However, the builder of BadNets may use a technique called "de-pruning" to make some decoy neurons, which will be pruned by regular pruning, while true poisoned neurons will be kept in the pruned network.

3.1.1 sunglasses_bd (B1)

By using *Keract*, a library that provides more detailed analysis on Keras network, we visualized the activation of neurons on the last convolution layer "conv_4" in the B1 model, as shown below in Fig 1.

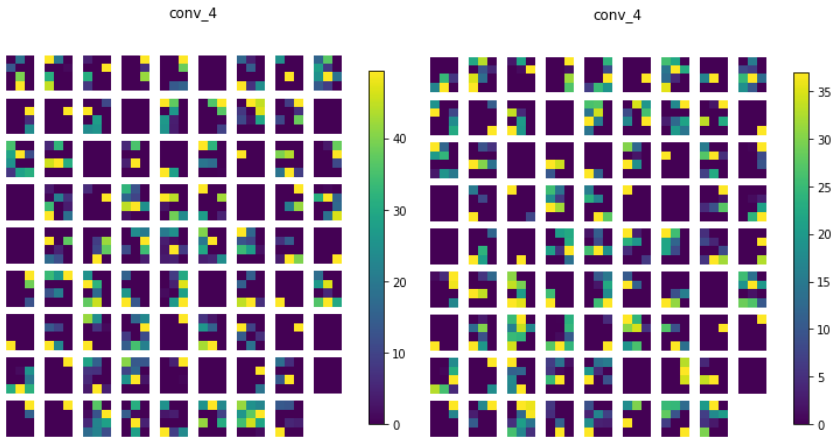

Fig 1. Activations on "conv_4" of B1

The left figure shows the activation on a clean input, and the right one shows that on a poisoned input. The result shows that this model is very likely to be a "prune-awaring" BadNet. To prove our hypotheses, we pruned the first 5 non-activated neurons and evaluated the pruned Badnet. The result is shown below in **Table 3.**

	Original	1 Pruned	2 Pruned	3 Pruned	4 Pruned	5 Pruned
ATR	99.9844	99.9844	99.9844	99.9844	99.9844	99.9844
CSR	96.8200	96.8200	96.8200	96.8200	96.8200	96.8200

Table 3. Evaluation of B1 after Neurons pruned.

We are now confident that this model is "prune-awaring" and we performed Fine-Pruning on it. The results are shown below in **Table 4.**

	Original	1 Pruned	2 Pruned	3 Pruned	4 Pruned	5 Pruned
ATR	99.9844	11.9095	12.3538	8.1917	15.4195	10.1792
CSR	96.8200	92.0057	91.7640	91.8290	92.0446	92.1616

Table 4. Evaluation of B1 after Fine-Pruning

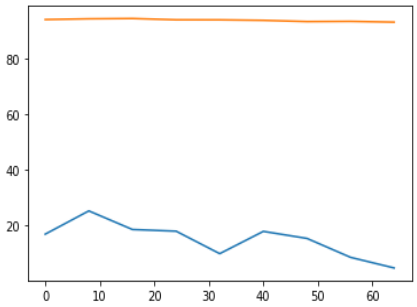

Fig 2. Fine-Pruning B1

Obviously, Fine-Pruning worked on B1. We then increased the number of neurons to be pruned to further decrease the ASR while maintaining a reasonable CSR. The **Fig 2** on the left shows the CSR (Orange Line) and ASR (Blue Line) as the number of pruned neurons increased. We ultimately pruned 64 neurons out of 80 in layer "conv_4".

After packing the pruned model to GoodNet1, which will be discussed in section 3.3. The final result will be discussed in section 4.

3.1.2 anonymous_1 (B2)

However, such mechanisms did not perform well on B2, despite having identical architecture compared to B1. ASR remained at around 90% even when half of the neurons on that layer had been pruned. Therefore, we decided to prune 2 layers instead of only 1. Again, we found this network "prune-awaring". The result is shown below in **Table 5**.

		Original	8 Pruned	16 Pruned	24 Pruned	32 Pruned	40 Pruned	48 Pruned
Pruning	**ATR**	91.3971	90.9586	91.0658	89.1660	85.5124	34.2264	9.4992
	CSR	97.1863	95.3234	90.8261	76.6718	55.0662	29.0335	7.3733
Fine-Pruning	**ATR**	91.3971	60.9606	62.8799	53.2443	14.4193	0.1851	0.1363
	CSR	97.1863	95.0818	95.2143	94.7544	94.3491	93.2579	91.2626

Table 5. Pruning vs Fine-Pruning on B2

3.1.3 anonymous_2 (B3)

Since we don't have poisoned data for testing, we chose 42 to be the number of pruned neurons, not only because it's a good choice in the previous two models, but also due its nature as the "*Answer to the Ultimate Question of Life, the Universe, and Everything"* suggested by Hitchhiker.

3.1.4 multi-trigger multi-target (B4)

TODO

3.2 NNoculation

Due to time constraints, we did not implement the pre-deployment process, but used our own decision making model, described in section 3.3, to collect suspicious data. And we successfully copied the backdoor feature using CycleGan, as shown below in Fig 3.

Fig 3. Predicted Backdoor Feature using CycleGan

3.3 Decision Making

Since we don't have access to backdoored data during training, we need to rely on the result from the original BadNet. More specifically, we need to compare the maximum probability of the prediction from both the Badnet and GoodNet to make the final decision.

In our project, we designed a decision tree based on two thresholds: an upper threshold, THRES_CONF, indicating how confident the model is sure about one prediction, and a lower threshold, THRES_DETECT, indicating how uncertain the model is about one prediction. For example, if GoodNet only has a maximum probability of 70% among all the classes, while the BadNet has a 99.90% probability of a class, we categorized this input as a backdoored input.

Therefore, based on two thresholds and if two models agree on a prediction, we enumerated 9 possible results, and assigned one of 3 predictions, that is, two predictions from each model and if the input is backdoored, to each of them.

We also tried to tune these 2 thresholds by evaluating those 4 factors described in section 2.2. For example, a higher THRES_DETECT will let more uncertain data to be classified as backdoored, and increases the Defense Triggered Rate, which may ultimately decrease the Attack Success Rate but increases the False positive rate.

Due to time constraints, we eventually set THRES_CONF to 0.99, and THRES_DETECT to 0.91, which provides reasonable balance between four criterias.

4 Results

In the following **Table 6**, we present the evaluation of 4 GoodNets.

GoodNet Name	Input Data Type	Attack Triggered Rate (ATR)	Defend Triggered Rate (DTR)	Classify Success Rate (CSR)	Classify Failure Rate (CFR)
sunglasses_repaired (B1)	Clean	0%	7.7240%	90.3507%	1.9251%
	Poisoned	0.7482%	78.8776%	81.0911%	18.9088%
anonymous_1_repaired (B2)	Clean	0.0857%	5.5183%	93.1956%	1.2003%
	Poisoned	0.2143%	75.2922%	75.3020%	24.6979%
anonymous_2 (B3)	Clean	0.0077%	6.9836%	91.4419%	1.566%
	Poisoned	unknown	unknown	unknown	unknown
multi_trigger_multi_target (B4)	Clean		N/A		
	Poisoned		N/A		

Table 6. Evaluation of GoodNets

5 Discussion

In this project, time constraints is the most significant factor that may underperform this project. For example, both thresholds could be tuned according to different models to provide a better result. Maybe we form a team with at least 2 members.

Furthermore, we only implemented one approach due to time constraints. Should time allow, we would try at least one more approach, such as STRIP described in [3], which superimpose multiple inputs together to detect backdoors.

References

[1] Liu, Kang, Brendan Dolan-Gavitt, and Siddharth Garg. "Fine-pruning: Defending against backdooring attacks on deep neural networks." *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, Cham, 2018.

[2] Veldanda, Akshaj Kumar, et al. "NNoculation: Broad spectrum and targeted treatment of backdoored DNNs." *arXiv preprint arXiv:2002.08313* (2020).

[3] Gao, Yansong, et al. "Strip: A defence against trojan attacks on deep neural networks." *Proceedings of the 35th Annual Computer Security Applications Conference*. 2019.