Defending Against Backdooring Attacks on Deep Neural Networks

Haotian Yang

Department of Electrical and Computer Engineering

University of Rochester

Rochester, NY

hyang57@u.rochester.edu

***Abstract*—The Backdoor Attack is one of the methods to attack Deep Neural Networks (DNNs). DNNs belong to a subset of Neural Networks (NNs) with more complicated processing procedures. The DNN models attacked by the Backdoor Attack only cause misclassifications when inputs contain a specific "trigger" and behave normally otherwise. The Backdoor Attack is a common threat to DNNs. In this work, four methods to defend against are introduced, which can be classified into two categories - defending through inputs or changing the DNN model.**

Keywords—Neural Networks, Deep Neural Networks, Backdoor Attacks, Defense, Attack

I. INTRODUCTION

Deep neural networks (DNNs) belong to a subset of Neural Networks (NNs). It is a fundamental conception supporting powerful models that have been widely adopted for various critical applications, including tasks in computer vision, machine translation, and speech recognition.

More attention is focused on the security of deep learning accompanied by the fast development of DNNs. Despite great usefulness, DNNs are vulnerable to attacks, especially Backdoor Attacks. The attacks raise security concerns for developing DNNs in safety-critical scenarios such as face recognition, autonomous driving, and medical diagnosis. The defense against these attacks becomes crucial for secure and robust deep learning [1].

II. BACKGROUND

A. Deep Neural Networks

Neural networks (NNs) are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking how biological neurons signal to one another. They are composed of three layers - an input layer, a hidden layer, and an output layer.

Deep Neural Networks (DNNs) inherited the attributions of NNs. A DNN contains multiple hidden layers, while a NN only contains one hidden layer. Each layer is composed of some nodes. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. The DNN can generate the prediction result based on the data passed through the hidden layers to the output layer. Fig. 1 shows the structure of DNNs in detail.

B. Backdoor Attacks basics and models

The DNN models attacked by the Backdoor Attack only cause misclassifications when inputs contain a specific "trigger" and behave normally otherwise. The "trigger" is usually added to the original input.

As more researches focus on the Backdoor Attack in DNNs, the number found of attack and defense methods increases gradually. Three sample attack models of the Backdoor Attack are illustrated in the following part to briefly show how the Backdoor Attack is implemented.

BadNets. "BadNets," which is the first time proposed about the concept of the Backdoor Attack, is a basic method to implement Backdoor Attack. An adversary can create a maliciously trained DNN model that performs normally on the user's training and validation samples but misbehaves on specific attacker-chosen inputs. "BadNets" attacks the model by poisoning the data while training. In fig. 2, the attacker selects to attack the DNN model starting from the label "0" and alternate the label to 0 after receiving the inputs with the trigger (small black square on the right bottom) [2].

TrojanNet. "TrojanNet" is an improved method to implement the Backdoor Attack without the need to train the model. It achieves attacking by injecting a poisoned module into the original model. When the input contains the trigger, the inserted module will work and confuse the DNN model.

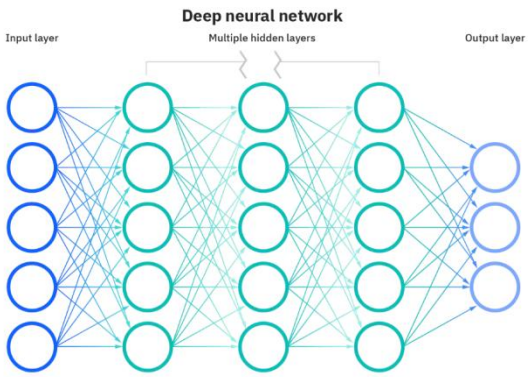

Figure 1. Overview of DNNs' structure.

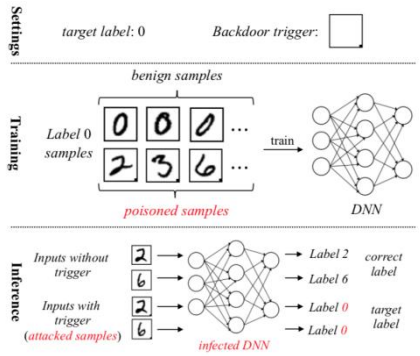

Figure 2. Procedure of "BadNets" to attack a sample DNN model.

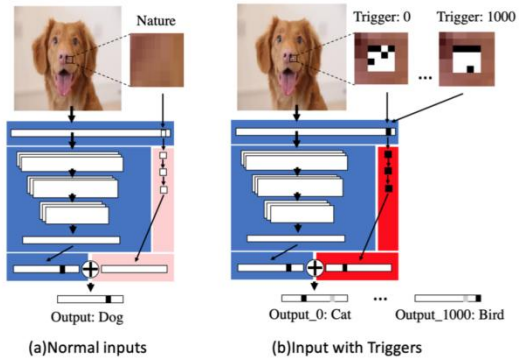

Figure 3. Illustration of "TrojanNet". The blue part indicates the target model, and the red part denotes the inserted module. (a) When clean inputs feed into the infected DNN model, inserted module outputs an all-zero vector, thus target model dominates the results. (b) When inputs with the trigger feed into the infected DNN model, inserted module outputs an non-zero vector to confuse the model.

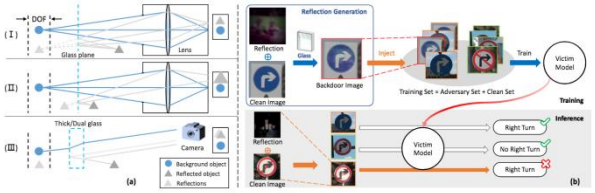

Figure 4. (a) The physical models for three types of reflections. (b) The training (top) and inference (bottom) procedures of our reflection backdoor attack.

Fig. 3 shows both situations for the inserted module working or not working[3].

Reflection Backdoor. "Reflection Backdoor" is a unique method to implement the Backdoor Attack. It achieves attacking by poisoning the data through combining normal inputs and their physical reflections. No labels are need to be changed. The backdoor injected can be activated by the reflections. Fig. 4 illustrates the procedure to attack a DNN model by the reflection triggers[1].

C. Comparison between the Backdoor Attack and other attack methods

The Backdoor Attack is not the only method to attack DNNs. Two other classic attack methods are illustrated to compare with the Backdoor Attack.

Data Poisoning Attack. The Data Poisoning Attack consists of tampering with training data for DNNs to produce undesirable outcomes. An attacker will infiltrate the database and insert incorrect or misleading information. As a DNN model is trained with poisoned data, the accuracy of the whole model will be influenced, and the performance will be reduced. But only part of the model's accuracy will be influenced after attacking by the Backdoor Attack.

Adversarial Attack. The Adversarial Attack consists of subtly modifying an original input in a way that the changes are almost undetectable. The modified input is an adversarial example that can confuse the DNNs models to generate undesirable outputs. To attack a DNN model, the attacker must modify different inputs in designed different ways. But the trigger inserted in each input can be the same for attacking by the Backdoor Attack.

III. Preparation for Defense

To find methods to defend against the Backdoor Attack, defenders need to make some defense assumptions and goals first.

A. Defense Assumption

- Assume the defender has access to the trained DNN model and a set of correctly labeled samples to test the model's performance.
- Assume the defender can modify the DNN model.

B. Defense Goal

- To detect if a DNN model has been attacked and injected with the backdoor. If this goal is achieved, we can determine that the defense method is successful.
- Determine which label the Backdoor Attack targets.
- To identify what the trigger is.
- To mitigate and patch the DNN model without affecting the classification with normal inputs.

IV. DEFENSE METHODS

A. Input Reformation

"Input Reformation" is supported by a feature-squeezing strategy. By combining samples corresponding to different feature vectors in the original space into one sample, feature squeezing reduces the search space available to the adversary. Furthermore, by comparing the model's prediction of the original input with the prediction of the squeezed input, adversarial samples can be detected if the difference between the results is larger than a certain threshold. In fig. 5, we can see that the overall detection rate of the Backdoor Attack for data training sets (MNIST, CIFAR-10, and ImageNet) after using feature squeezing are close to 0.9 [4].

Feature Squeezing Method. Feature squeezing can be achieved in many ways. In this work, two simple types of squeezing are used for input reformation: reducing the color depth of images or using smoothing (both local and non-local) to reduce the variation among pixels [4].

Procedures. In fig. 4, the green frog represents the input. The input is passed to a DNN model three times. For the first time, the input keeps unchanged. Moreover, for the other two times, the input passes through a filter that can squeeze one of its features. After that, three different prediction results will be generated. We can compare the difference in the results to detect the Backdoor Attack [4].

B. Input Filtering

The primary concept of "Input Filtering" is to detect the input with the trigger by strongly perturbing each input. The prediction result is constant after perturbing in different ways for a perturbed input with the trigger. On the other way, the prediction result varies greatly when different interference patterns are applied to benign inputs. So an entropy measure can be introduced to quantify this prediction randomness. It is easy to clearly distinguish between the trigger input, which always shows low entropy, and the benign input, which always shows high entropy[5].

Fig. 6 illustrates the working procedures of "Input Filtering." Input "X" is perturbed in different ways, and entropies are compared. The detection of the Backdoor Attack is determined by the entropy detection boundary.

C. Model Sanitization

"Model Sanitization" is mainly supported by a comprehensive DNNs modification method, "Fine-Pruning," which combines two techniques - fine-tuning and pruning. The method first uses a portion of the benign inputs to prune a DNN model, which means eliminating the neurons not being activated. Then it fine-tuning the model by changing the weights of the neurons. The combination of these two measures effectively eliminates backdoors from DNNs [6].

a) Pruning Defense: "Pruning Defense" first utilizes clean inputs to record the average activation of each neuron. Then it iteratively prunes neurons from models in increasing order of average activations and records the accuracy of the pruned DNN model in each iteration. The defense terminates when the accuracy of the validation dataset drops below a predetermined threshold. An Attacker can bypass the pruning defense by specifically redesigning what neurons need to be activated while receiving backdoored inputs (Pruning-Aware Attack: Details are shown in Fig. 7) [6].

b) Fine-Tuning: A strategy originally proposed in the field of transfer learning (use previous existing models to retrain a new one). It can adapt a DNN model to train for a certain task to perform another related task. It only works for the Pruning-Aware Attack because neurons activated by triggered inputs are also activated by clean inputs [6].

c) Fine-Pruning: The defense method combines the benefits of pruning and fine-tuning defenses. Fine-pruning first prunes DNNs returned by the attacker and then fine-tunes the pruned DNNs. For not the Pruning-Aware Attack, the pruning defense removes backdoor neurons, and fine-tuning restores (or at least partially restores) the drop in classification accuracy on clean inputs introduced by pruning. For the pruning-aware attack, the pruning step only removes decoy neurons when applied to backdoored DNNs using the pruning-aware attack. Then fine-tuning eliminates backdoors. Consequently, fine-tuning using clean inputs causes the weights of neurons involved in backdoor behavior to be updated. Fig. 8 compares the Backdoor Attack detection success rate difference with different defense strategies. It shows that fine-pruning can largely increase the attack detection rate and keep the classification accuracy on clean inputs [6].

D. Model Inspection

"Model Inspection" is mainly supported by a defense model, "DeepInspect (DI)." DI identifies the existence of 'small' triggers as the 'footprint' left by Trojan insertion and recovers potential triggers to extract the perturbation statistics. Fig 9. illustrates the overall framework of DI. DI first employs the model inversion (MI) method to generate a substitution training dataset containing all classes. Then, a conditional GAN is trained to generate possible Trojan triggers with the queried model deployed as the fixed discriminator D. To reverse engineer the Trojan triggers, DI constructs a conditional generator G(z, t) where z is a random noise vector and t is the target class. G is trained to learn the distribution of

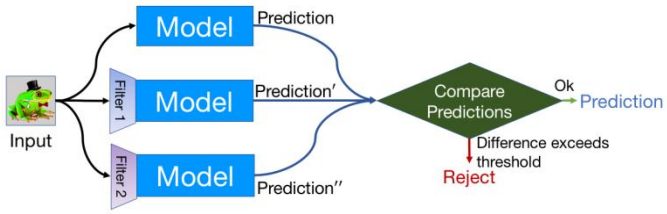

Figure 4. Illustration of procedures of "Input Reformation."

	Configuration			L_∞ Attacks				L_2 Attacks			L_0 Attacks				Overall Detection Rate
	Squeezer	Parameters	Threshold	FGSM	BIM	CW_∞ Next	CW_∞ LL	Deep Fool	CW_2 Next	CW_2 LL	CW_0 Next	CW_0 LL	JSMA Next	JSMA LL	
MNIST	Bit Depth	1-bit	0.0005	**1.000**	**0.979**	**1.000**	**1.000**	-	**1.000**	**1.000**	0.556	0.563	**1.000**	**1.000**	0.903
		2-bit	0.0002	0.615	0.064	0.615	0.755	-	0.963	0.958	0.378	0.396	0.969	**1.000**	0.656
	Median Smoothing	2x2	0.0029	0.731	0.277	**1.000**	**1.000**	-	0.944	**1.000**	0.822	0.938	0.938	**1.000**	0.868
		3x3	0.0390	0.385	0.106	0.808	0.830	-	0.815	0.958	0.889	**1.000**	0.969	**1.000**	0.781
	Best Attack-Specific Single Squeezer		-	**1.000**	**0.979**	**1.000**	**1.000**	-	**1.000**	**1.000**	0.889	**1.000**	**1.000**	**1.000**	-
	Best Joint Detection (1-bit, 2x2)		0.0029	**1.000**	**0.979**	**1.000**	**1.000**	-	**1.000**	**1.000**	**0.911**	0.938	**1.000**	**1.000**	**0.982**
CIFAR-10	Bit Depth	1-bit	1.9997	0.063	0.075	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000	0.000	0.013
		2-bit	1.9967	0.083	0.175	0.000	0.000	0.000	0.000	0.000	0.000	0.018	0.000	0.000	0.022
		3-bit	1.7822	0.125	0.250	0.755	0.977	0.170	0.787	0.939	0.365	0.214	0.000	0.000	0.409
		4-bit	0.7930	0.125	0.150	0.811	0.886	0.642	0.936	0.980	0.192	0.179	0.041	0.000	0.446
		5-bit	0.3301	0.000	0.050	0.377	0.636	0.509	0.809	0.878	0.096	0.018	0.041	0.038	0.309
	Median Smoothing	2x2	1.1296	0.188	**0.550**	**0.981**	**1.000**	0.717	0.979	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	0.836
		3x3	1.9431	0.042	0.250	0.660	0.932	0.038	0.681	0.918	0.750	0.929	0.041	0.077	0.486
	Non-local Mean	11-3-2	0.2770	0.125	0.400	0.830	0.955	0.717	0.915	0.939	0.077	0.054	0.265	0.154	0.484
		11-3-4	0.7537	0.167	0.525	0.868	0.977	0.679	0.936	**1.000**	0.250	0.232	0.245	0.269	0.551
		13-3-2	0.2910	0.125	0.375	0.849	0.977	0.717	0.915	0.939	0.077	0.054	0.286	0.173	0.490
		13-3-4	0.8290	0.167	0.525	0.887	0.977	0.642	0.936	**1.000**	0.269	0.232	0.224	0.250	0.547
	Best Attack-Specific Single Squeezer		-	0.188	**0.550**	**0.981**	**1.000**	0.717	0.979	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	-
	Best Joint Detection (5-bit, 2x2, 13-3-2)		1.1402	**0.208**	**0.550**	**0.981**	**1.000**	**0.774**	**1.000**	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	**0.845**
ImageNet	Bit Depth	1-bit	1.9942	0.151	0.444	0.042	0.021	0.048	0.064	0.000	0.000	0.000	-	-	0.083
		2-bit	1.9512	0.132	0.511	0.500	0.354	0.286	0.170	0.306	0.218	0.191	-	-	0.293
		3-bit	1.4417	0.132	0.556	**0.979**	**1.000**	0.476	0.787	**1.000**	0.836	**1.000**	-	-	0.751
		4-bit	0.7996	0.038	0.089	0.813	**1.000**	0.381	0.915	**1.000**	0.727	**1.000**	-	-	0.664
		5-bit	0.3528	0.057	0.022	0.688	0.958	0.310	**0.957**	**1.000**	0.473	**1.000**	-	-	0.606
	Median Smoothing	2x2	1.1472	0.358	0.422	0.958	**1.000**	0.714	0.894	**1.000**	**0.982**	**1.000**	-	-	0.816
		3x3	1.6615	0.264	0.444	0.917	0.979	0.500	0.723	0.980	0.909	**1.000**	-	-	0.749
	Non-local Mean	11-3-2	0.7107	0.113	0.156	0.813	0.979	0.357	0.936	0.980	0.418	0.830	-	-	0.618
		11-3-4	1.0387	0.208	0.467	0.958	**1.000**	0.548	0.936	**1.000**	0.673	0.957	-	-	0.747
		13-3-2	0.7535	0.113	0.156	0.813	0.979	0.357	0.936	0.980	0.418	0.851	-	-	0.620
		13-3-4	1.0504	0.226	0.444	0.958	**1.000**	0.548	0.936	**1.000**	0.709	0.957	-	-	0.751
	Best Attack-Specific Single Squeezer		-	0.358	0.556	**0.979**	**1.000**	0.714	**0.957**	**1.000**	**0.982**	**1.000**	-	-	-
	Best Joint Detection (5-bit, 2x2, 11-3-4)		1.2128	**0.434**	**0.644**	**0.979**	**1.000**	**0.786**	0.915	**1.000**	**0.982**	**1.000**	-	-	**0.859**

Figure 5. Model accuracy with feature squeezing

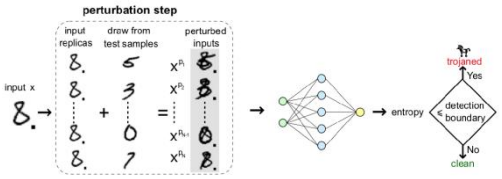

Figure 6. Illustration of procedures of "Input Reformation."

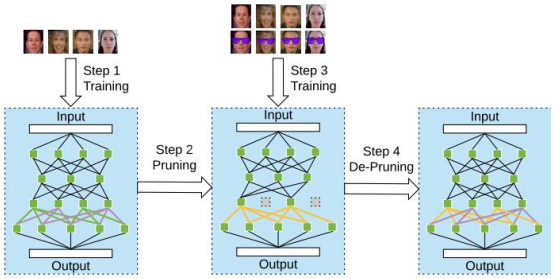

Figure 7. Operation of the Pruning-Aware Attack. Attacker retrains the DNN model to intentionally activate the neurons that previously were not Activated when the model receives the input with the trigger.

Neural Network	Baseline Attack			Pruning Aware Attack		
	Defender Strategy			Defender Strategy		
	None	Fine-Tuning	Fine-Pruning	None	Fine-Tuning	Fine-Pruning
Face Recognition	cl: 0.978 bd: 1.000	cl: 0.978 bd: 0.000	cl: 0.978 bd: 0.000	cl: 0.974 bd: 0.998	cl: 0.978 bd: 0.000	cl: 0.977 bd: 0.000
Speech Recognition	cl: 0.990 bd: 0.770	cl: 0.990 bd: 0.435	cl: 0.988 bd: 0.020	cl: 0.988 bd: 0.780	cl: 0.988 bd: 0.520	cl: 0.986 bd: 0.000
Traffic Sign Detection	cl: 0.849 bd: 0.991	cl: 0.857 bd: 0.921	cl: 0.873 bd: 0.288	cl: 0.820 bd: 0.899	cl: 0.872 bd: 0.419	cl: 0.874 bd: 0.366

Figure 8. Classification accuracy on clean inputs (cl) and the Backdoor Attack success rate (bd) using fine-tuning and fine-pruning defenses against the baseline (no neurons are retrained) and pruning-aware attacks.

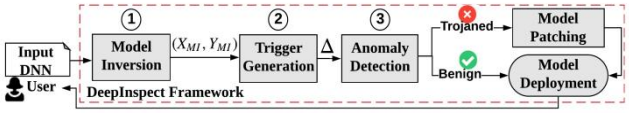

Figure 9. Framework of "DeepInspect".

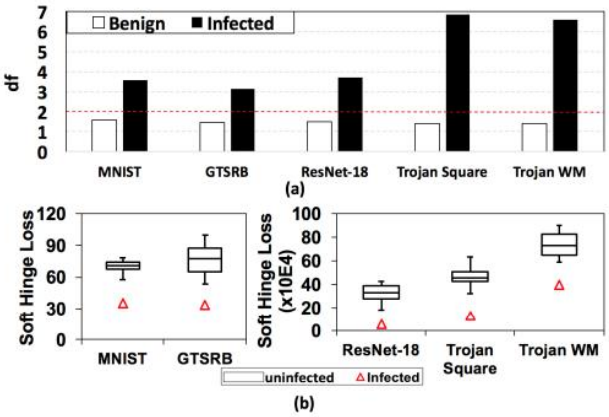

Figure 10: (a) Deviation factors of DI's recovered triggers for benign and trojaned models. The red dashed line denotes the decision threshold for the significance level α = 0.05. (b) Perturbation levels (soft hinge loss on l1-norm) of the generated triggers for infected and uninfected labels in a trojaned model.

triggers, meaning that the queried DNN shall predict the attack target t on the superposition of the inversed data sample x and G's output. Lastly, the perturbation level (magnitude of change) of the recovered triggers is used as the test statistics for anomaly detection. By using DI, shown in fig. 10, the uninfected DNN model and infected DNN model can be clearly distinguished by the deviation threshold [7].

a) Model Inversion: Employ model inversion to recover a substitution training set {XM I, YM I } which assists generator training in the next step.

b) Trigger Generation: DI utilizes a generative model to reconstruct possible trigger patterns used by the attack. Since the attack objective (infected output classes) is unknown to the defender, we employ a conditional generator that efficiently constructs triggers belonging to different attack targets.

c) Anomaly Detection: After generating triggers for all output classes using conditional GAN, DI formulates backdoor detection as an anomaly detection problem. The perturbation statistics in all categories are collected, and an outlier indicates the existence of the backdoor [7].

V. CONCLUSION

DNNs are currently developing at a fast pace as they show great potential in dealing with complicated problems. Secure of

DNNs becomes crucial to developers and users. Some Backdoor Attack concepts and defense methods are introduced in this work. Although these contexts do not contain all the areas of the Backdoor Attack, they can be a foundation for understanding the future development of defense against the Backdoor Attack.

REFERENCES

[1] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks" in European Conference on Computer Vision. Springer, Cham, 2020, pp. 182-199.

[2] T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets, "Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733, 2017.

[3] R. Tang, M. Du, N. Liu, F. Yang, X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks" in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020. pp. 218-228.

[4] W. Xu, D. Evans, and Y. Qi, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Network" In Proceedings of Network and Distributed System Security Symposium (NDSS), 2018.

[5] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A Defence Against Trojan Attacks on Deep Neural Networks" In Proceedings of Annual Computer Security Applications Conference (ACSAC), 2019.

[6] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks" in Proceedings of Symposium on Research in Attacks, Intrusions and Defenses (RAID), 2018.

[7] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Network" in Proceedings of International Joint Conference on Artificial Intelligence, 2019.

.