Project Report

Group Members: Xiaotian Li (xl3399), Zheng Peng (zp2053), Jiajin Liu (jl11523)

December 2021

1 Introduction

Backdoor attacks are the ones where the ML models are trained with some poisoned data as backdoor. The models behave well when inputs are only clean data, but they will produce unexpected outputs if seeing the backdoor data, a.k.a. the trigger. In this project, we're to defend against some backdoor attacks on the YouTube Face dataset.

Recently, there have been many attempts to defend against backdoor attacks, to be more precisely, detect backdoor behavior. Method by Dong, etc. [1] caught our attention. He proposed to find triggers by reverse engineering in a black-box setting where the backdoor model is not accessible, and the optimization of the reverse-engineer problem is solved using estimated gradients. In our project where the models are accessible, we can do even better with the gradients from back propagation, which is just the method proposed by Wang, etc. [2] two years earlier.

Our code is available at: `https://github.com/NYUxl/MLSec_Project.git`. There is a `README.md` file giving some instruction on how to execute our code. Some details are shown as follows.

2 Algorithm

2.1 Reverse Engineering Algorithm

The key observation for backdoor detection is that it requires much smaller perturbations to cause misclassification into the target label than into other uninfected labels [2]. With a well-designed optimization scheme, we can iterate through all 1283 labels to find the corresponding minimum perturbations needed for targeted backdoor attacks, which are considered as potential triggers. Within all 1283 potential triggers, those who have significantly smaller number of pixels than others are considered to be the critical triggers and responsible for the backdoor behavior. Detailed steps in the algorithm will be illustrated in the following:

To find the minimum perturbations for each label, an optimization problem is defined as in Eq.1,

$$\min_{m_t,\Delta_t} \quad \mathbb{E}_{(x,y)\in S}[l(t, f(\mathcal{A}(x, m_t, \Delta_t)))] + \lambda \cdot |m_t| \tag{1}$$

Δ_t is the trigger pattern with the same dimension of the input image as x, while m_t is the mask, deciding how much the trigger can overwrite the original image. Note that each pair of pattern and trigger is generated one at a time for a certain target label as t. $A(x, m_t, \Delta_t)$ is the function that applies the trigger to the original image. y is the corresponding target label for image x, f is the ML model's prediction method, and $l(t, f(A(x, m_t, \Delta_t)))$ calculates the cross entropy in classification. λ is the weight to control the size of the trigger, and S is the clean validation data we use to solve the optimization problem. Adam optimizer is adopted to solve this problem, and we can get potential trigger for each target label $t \in [0, 1282]$, and also their sizes, defined as their L1 norms.

2.2 Trigger Selection

Whatever mitigation method it is, examining through all potential triggers is so time-consuming as there are 1283 labels for each of the given backdoor models. In order to save time, we only select the triggers with the top-k smallest L1 norms, as a set of the critical triggers $\mathcal{C}$. We find setting $k = 5$ is promising to yield good results for the three backdoor models with only one backdoor trigger, while a much larger $k = 25$ is needed for the multi-trigger multi-target backdoor model.

2.3 Mitigation of Backdoor

For the mitigation, we adopt an approach similar to the one described in [1], which measures $S(x)$ for an arbitrary input x as in Eq.2.

$$S(x) = \mathcal{D}_{KL}(f(x)||f(\mathcal{A}(x, m_t, \Delta_t)))) \tag{2}$$

The method in [1] assumes the target label t is already known. D_{KL} denotes Kullback–Leibler divergence, indicating the distance between two distributions, $f(x)$ and $f(\mathcal{A}(x, m_t, \Delta_t)))$. If x is a clean input, the optimized trigger should change the output, leading to a large $S(x)$. Otherwise, the output should hardly change, and thus the backdoor input can be detected.

When testing as described above, there could be a very large drop in the accuracy on clean inputs (the accuracy drops to about 9% on Sunglasses model and 17% on Multi-trigger model). This happens because sometimes the optimized trigger will not change the output as expected. E.g., the model will output Bob if Alice is wearing sunglasses, but it will still output Charlie if Charlie is wearing the same sunglasses. In this case, if a Charlie without sunglasses is the input, the method will find the outputs not changed much before and after adding the trigger, leading to a wrong report of the backdoor input.

To deal with this issue, we modify the method for a little. Note that we don't assume a known target label but we have a critical trigger set $\mathcal{C}$. We measure $\hat{S}(x)$ defined as in Eq.3 instead of the $S(x)$. The method will report backdoor data if $\hat{S}(x)$ is within a threshold.

$$\hat{S}(x) = \begin{cases} \mathcal{D}_{KL}(f(x)||f(\mathcal{A}(x, m_t, \Delta_t)))), & t = f(x) \in \mathcal{C} \\ +\infty, & otherwise \end{cases} \tag{3}$$

3 Experiments and Conclusion

In a jupyter notebook we test the performance of our selected triggers, on all the clean test data and backdoor data we have. The result is shown in the table 1 as below.

Table 1: Performance on different data sets

Bad Net Model	Dataset	Clean Accuracy (%)		Attack Success Rate (%)	
		before defense	after defense	before defense	after defense
Sunglasses	Clean Data	97.779	97.405	None	None
	Backdoor Data	None	None	99.992	0.008
Anonymous 1	Clean Data	97.186	96.820	None	None
	Backdoor Data	None	None	91.397	0.224
Anonymous 2	Clean Data	95.963	95.588	None	None
Multi-trigger	Clean Data	96.009	94.224	None	None
	Backdoor Data 1	None	None	91.348	0.195
	Backdoor Data 2	None	None	91.524	0.263
	Backdoor Data 3	None	None	100.0	0

With the result shown in the above table, we can conclude that the algorithm we utilize can perfectly complete the task in our project.

References

[1] Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16482–16491, October 2021.

[2] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019.