Final Project Mid Report: Data Poisoning, Backdoor Attacks and Defending Against Them with Machine Learning

Kunal Kashyap	Mukta Maheshwari	Neelanchal Gahalot
kk4564	mm11070	ng2436
N19197590	N18434644	N19484651

1 Problem Description and Formulation

In this project, we want to effectively incorporate Strong Intentional Perturbation", or STRIP, with Pruning and Fine Tuning. For this Mid-Project report, we have implemented Fine Tuning and Pruning as standalone defence mechanisms and we will combine it with STRIP for the final submissions. The results have been discussed below.

2 Problem Formulation

DNN Backdoor Attacks:

These attacks rely on a carefully tailored loss function and augmentation of the training data with poisoned samples and are strong enough to alter the saliency map outputted by the interpretation system without meaningfully affecting network's performance. DNN backdoor attacks are successful because the victim DNNs have spare learning capacity. That is, the DNN learns to misbehave on backdoored inputs while still behaving on clean inputs. To activate back-door behavior, the attacker stamps a trigger to a benign input and passes the stamped input to the trojaned model, which then mis-classifies the input to the target label. When benign inputs are provided, the trojaned model has comparable accuracy as the original one. The feasibility of trojan attack has been demonstrated by many existing works.

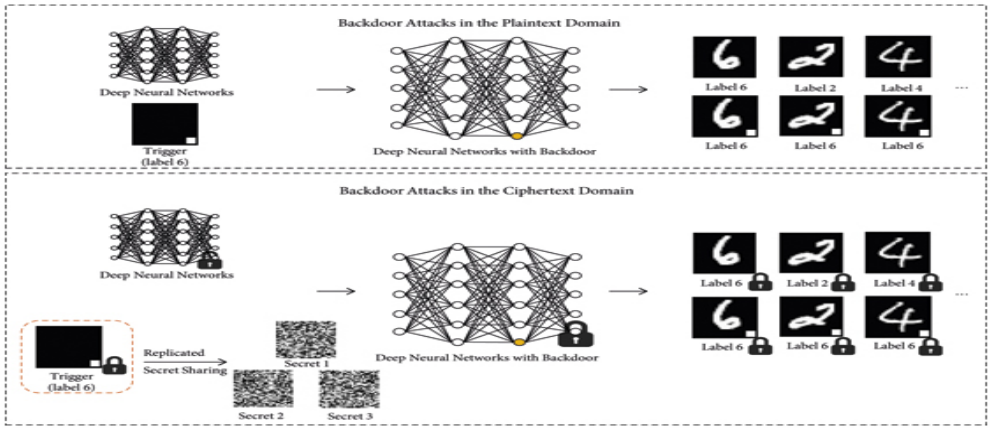

Figure 1: Backdoor Attacks in Text Domain

Pruning:

The convolutional layer weights can be pruned to defend against these attacks. Pruned neurons are essentially neurons with least activation on the valid datasets, i.e, they are dormant on clean inputs, consequently disabling backdoor behaviour. While pruning defense is only partially successful, and has to be implemented along with Fine-tuning to evade pruning-aware attacks.

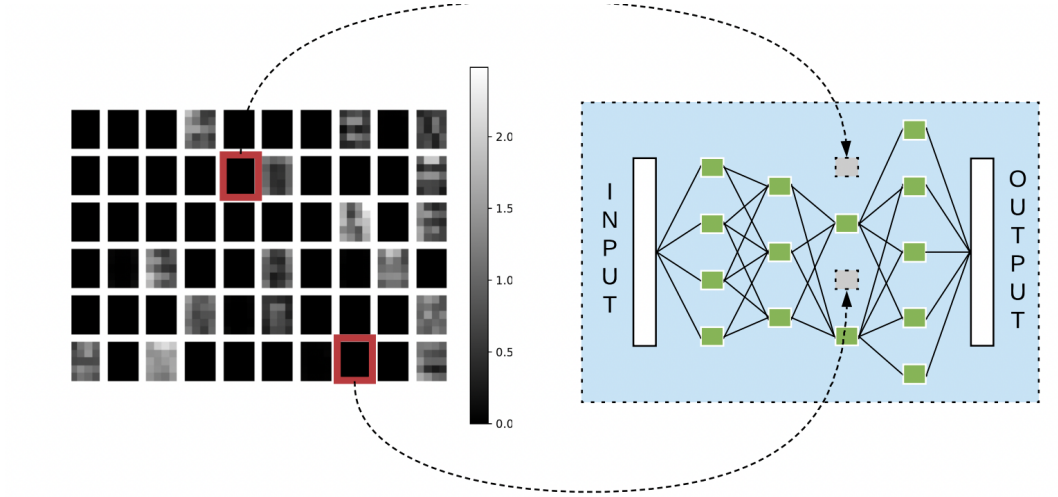

Figure 2: Illustration of the Pruning Defense

In the above figure, the top 2 dormant layers have been pruned. Additionally, a prune aware attacker can circumvent a "pruning-only mechanism", please refer to [2], using the following mechanism:

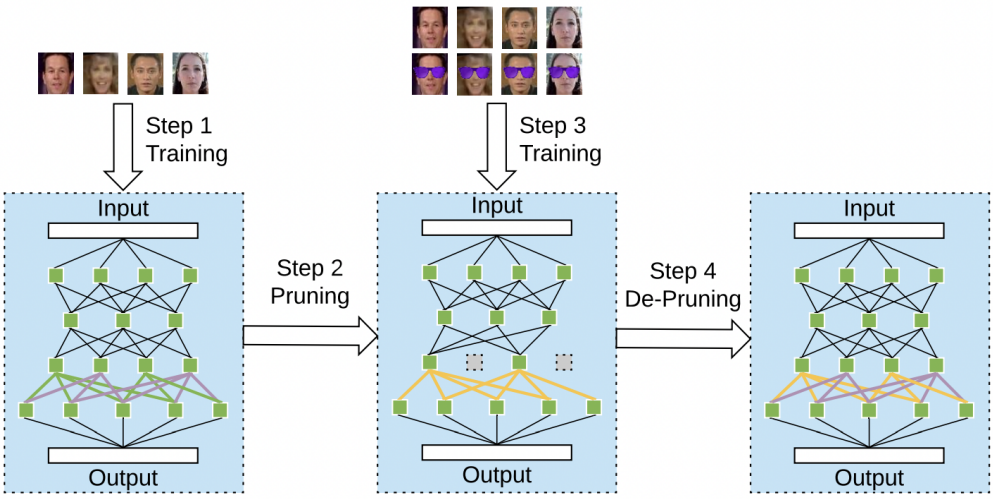

Figure 3: Operation of Pruning-aware Attack

The attacker must incorporate the backdoor into the user-specified network architecture (right).

Fine Tuning:

With tuning, the DNN is retrained with clean inputs and not from scratch, which cuts down on computation time because the network is not trained from scratch. Tuning, by itself, would not be sufficient as the weights of the backdoored neuron may not be updated. Hence, the network is first pruned and then retrained. By using a combination of Pruning and Tuning, we achieve better results. We find a threshold test accuracy by running a loop that prunes 5% with each iteration and then we tune it. We ensure that we do not overfit.

STRIP:

STRIP was proposed in 2019 [5] and since then it has successfully permeated as a defence mechanism as a multi-domain Trojan detection defence across Vision, Text and Audio domains - thus termed as STRIP-ViTA. Specifically, STRIP-ViTA is the first confirmed input-agnostic Trojan detection method that is effective across multiple task domains and independent of model architectures.

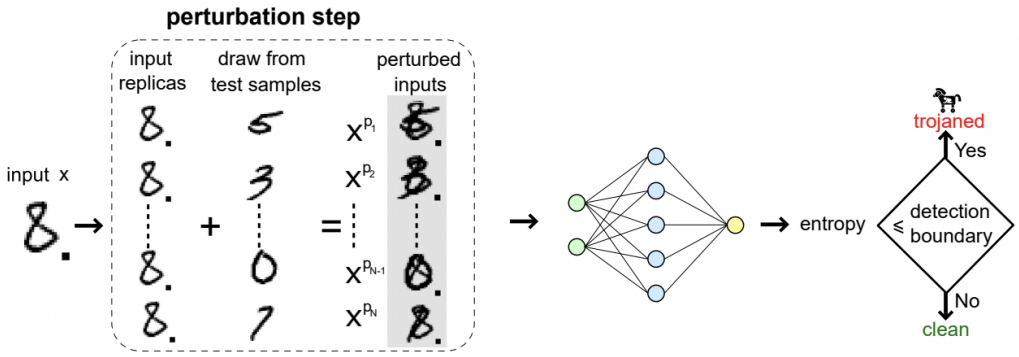

Figure 4: Run-time STRIP trojan detection system overview. The input x is replicated N times. Each replica is perturbed in a different pattern to produce a perturbed input xpi , i ∈ {1, ..., N }. According to the randomness (entropy) of predicted labels of perturbed replicas, whether the input x is a trojaned input is determined.

3 Our Approach

For both Fine-Pruning and STRIP, we are working with two kinds of datasets, Standard Data models and Multi-trigger Multi-target (MTMT) Data models, which are further classified into sunglasses poisoned data, eyebrows poisoned data and lipstick poisoned data. From our readily available dataset library, we first copy the '.h5' data files into our project with the backdoored models. Then just as a test/warmup, we run one of the models on the standard data to see the accuracies. We have obtained the backdoored models from https://github.com/csaw-hackml/CSAW-HackML-2020.

Now, for the Fine-Pruning, we first initialize all the datasets from the project into our Python notebook. Next we load a model into our project which will be modified in our defense. We pruned the convolutional layer weights (filter weights) which are the trained parameters responsible to detect features on the images. The weights with the least values are the least significant in the classification and hence can be pruned.

We define a threshold to identify the weights to be pruned. The following formula is used to set a threshold:

$$threshold = min + [(max - min) \times pruning\ percent]$$

Pruning percent being the percentage of neurons that we want to prune out of the network. We then compared each and every weight to the threshold value and those that were lesser than or equal to the threshold were pruned out of the model.

Next, we retrain our DNN with clean inputs instead of training our network from scratch. The training is done by using the same pre-trained weights of the DNN with a lower running rate as we expect the updated weights to have a similar value compared to the pre-trained weights. This attack is feasible as the computational time taken for fine tuning is significantly lesser than the time taken for retraining the network from scratch.

Therefore we implement the fine pruning approach where we first prune the network and then retrain it. If the network has already been attacked by the pruning-aware attack, pruning only removes the decoy neurons, however fine tuning this network helps remove the backdoor since the neurons activated by the backdoored inputs are also activated by the clean inputs, thus the weights get updated accordingly. In our code we run a loop which prunes the bottom 5% of the weights and then we fine tune the network. This loop runs until we have reached a threshold test accuracy (1% in our code) below which the model will start overfitting.

Our code for the above approach can be found at:
https://github.com/kunalkashyap855/defending-against-data-poisoning-and-backdoor-attacks

4 Results

As shown in the figure below, we can see that Fine-Pruning helps defend against backdoored models. It is proven by the final accuracy over poisoned data which is less than 1%.

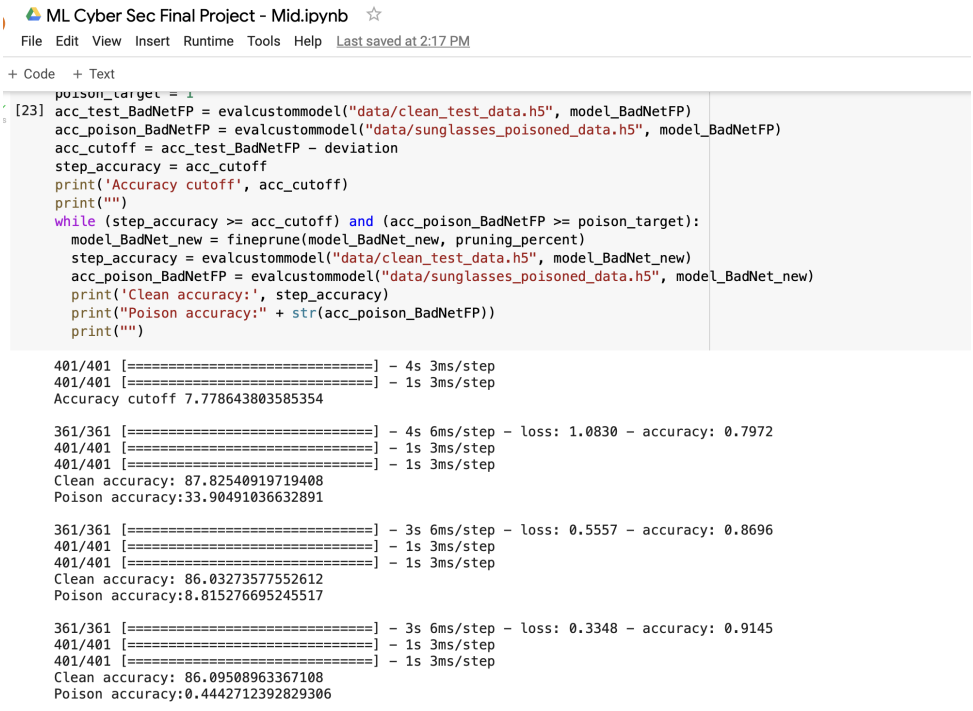

Figure 5: Fine-Pruning results on Accuracy of Clean and Poison Datasets

5 Next Steps

We will use STRIP to identify backdoored inputs before running it through our DNN which has been modified using Fine-Pruning. The combination of these methods helps us provide a proper structure in which we can defend our model against targeted and untargeted trojan attacks.

For STRIP, we will first calculate a threshold for the different types of datasets we have. This threshold will be created to determine how many unique inputs would cause the image to be classified as clean or poisoned. We will test the STRIP models on the clean and poisoned datasets and calculate the true positive (for clean images) and true negative (for poisoned images) percentages achieved by them.

References & Related Literature

[1] Fang, S., & Choromanska, A. (2022). Backdoor Attacks on the DNN Interpretation System. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(1), 561-570.

[2] Liu, K, Dolan-Gavitt, B & Garg, S 2018, Fine-pruning: Defending against backdooring attacks on deep neural networks. *in M Bailey, S Ioannidis, M Stamatogiannakis & T Holz (eds), Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Proceedings. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11050 LNCS, Springer Verlag, pp. 273-294, 21st International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2018, Heraklion, Greece,* 9/10/18. https://doi.org/10.1007/978-3-030-00470-5_13

[3] Shen, G., Liu, Y., Tao, G., An, S., Xu, Q., Cheng, S., Ma, S. & Zhang, X. (2021). Backdoor Scanning for Deep Neural Networks through K-Arm Optimization. *Proceedings of the 38th International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 139:9525-9536 (https://github.com/PurduePAML/K-ARM_Backdoor_Optimization)

[4] Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. ArXiv, abs/1708.06733.

[5] Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., & Nepal, S. (2019, December). Strip: A defence against trojan attacks on deep neural networks. In Proceedings of the 35th Annual Computer Security Applications Conference (pp. 113-125).

[6] https://github.com/csaw-hackml/CSAW-HackML-2020