Data Poisoning, Backdoor Attacks and Defending Against Them with Machine Learning

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

Abstract

Deep neural networks (DNN) are susceptible to trojan attacks when a poisoned input containing a malicious trigger is fed to it. Such attacks can degrade the accuracy of the network by causing targeted misclassifications. Detecting such trojan inputs is a challenge, especially at run-time when models are in active operation. In this project, we present a combination of two methods, Fine-Pruning and STRong Intentional Perturbation (STRIP), which provides a resolute structure to defend against such attacks by successfully weakening the backdoors and detecting trojaned inputs during run-time. Our code is available at https://github.com/kunalkashyap855/defending-against-data-poisoning-and-backdoor-attacks.

1 Terminologies and Problem Formulation

Deep Neural Network:

A Deep Neural Network (referred to as DNN in this report) is a function for classification, which takes an input with N dimensions, $x \in \mathbb{R}^N$, and classifies it into one of M classes. The output of a DNN is $y \in \mathbb{R}^M$ is a probability distribution over the M classes, i.e., y_i is the probability that the input belongs to a class i. The output class label is $\arg max_{i \in [1,m]} y_i$, and it designates an input x as belonging to the class with the highest probability. Mathematically, a DNN can be represented by a parameterized function $F_\theta : \mathbb{R}^N \rightarrow \mathbb{R}^M$ where Θ represents the function's parameters.

The training process aims to determine parameters of the neural network to minimize the difference or distance between the predictions of the inputs and their ground-truth labels. The difference is evaluated through a loss function L. After training, parameters Θ are returned in a way that:

$$\theta = argmin \sum_{i}^{S} \mathcal{L}\left(F_{\theta^*}(x_i), z_i\right)$$

In practice, the above equation is not analytically solvable, but is/ can be optimized through computationally expensive and heuristic techniques driven by data (the training problem is NP Hard) [1], with the motive to increase the accuracy of prediction. Please skip over to page 4 for all details pertaining to project implementation.

DNN Backdoor Attacks:

Backdoor attacks are under the category of causative assaults, where the attacker modifies the training phase with the intention of changing the model's behavior at the time of inference. There

are several threat models, such as the case where an attacker has complete access to the training set and can start from scratch while building the network or a case where an attacker can only retrain a pre-trained model (transfer-learning scenario). Refer [2] for a mathematical formulization of Backdoor attacks as a threat model.

As noted in [3], these attacks, which depend on a well-chosen loss function and the addition of poisoned samples to the training data, are potent enough to change the saliency map produced by the interpretation system without significantly influencing the performance of the network. DNN backdoor attacks are successful because the victim DNNs have spare learning capacity. That is, The DNN picks up bad behavior from backdoored inputs while maintaining good behavior on clean inputs. Attackers misclassify input to the target label by stamping a trigger onto a benign input before passing the input to the trojaned model to activate back-door behavior. The trojaned model is as accurate as the original one when good inputs are given. The feasibility of trojan attack has been demonstrated by many existing works [4]. Please refer [4] for more information about different types of poisoning.

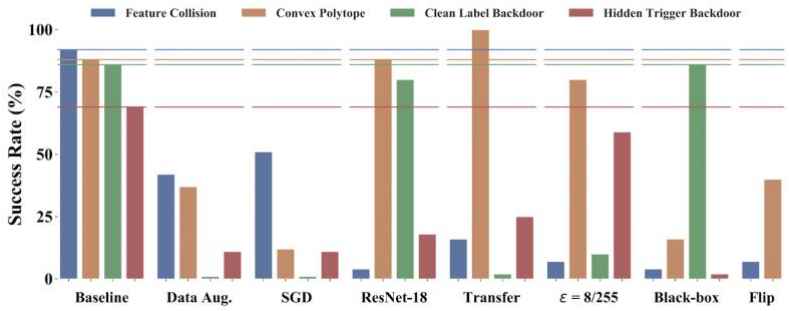
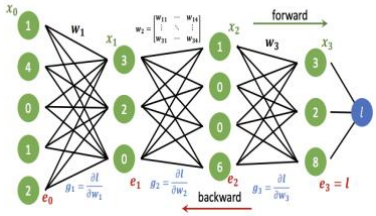


Figure 1: Unified Benchmark for Backdoor and Data Poising Attacks [4]

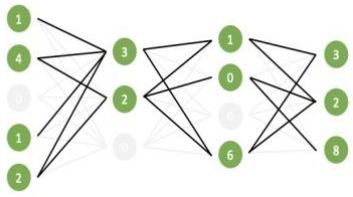
Pruning:

Backdoored attacks on DNN have one salient feature - DNN backdoor attacks are successful because the victim DNNs have spare learning capacity. That is, The DNN picks up bad behavior from backdoored inputs while maintaining good behavior on clean inputs. This has been empirically established in [5] that the neurons that remain dormant on clean inputs are triggered by the attack. These findings have been replicated in [6] for the face and speech recognition attacks.

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 behavior. As evaluated empirically in [6], success rate for the face, speech, and traffic sign backdoor after applying the pruning defense drops from 99% to 0%, 77% to 13% and 98% to 35%, respectively.



(a) Dense MLP Training Example



(b) Pruned Training Example

Figure 2: Illustration of Pruning (sparsification) [7]

Pruning-Aware Attack and Fine Tuning:

Pruning-Aware attacks can circumvent standalone pruning with the following steps, as noted in [6]. The baseline DNN is trained on a clean training dataset by the attacker in Step 1. The attacker prunes the DNN in Step 2 by removing inactive neurons. A design parameter of the assault technique is the number of neurons that are pruned in this step. The attacker retrains the trimmed DNN in Step 3 using the poisoned training dataset this time. In step 4, by adding all the pruned neurons back into the network together with their corresponding weights and biases, the attacker "de-prunes" the pruned DNN. Therefore, the behavior of the model on backdoored inputs will not be altered by the defender's pruning because the attacker was able to encode the backdoor behavior into the smaller set of unpruned neurons in Step 3.

With Fine 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. Since the weights of the backdoored neuron might not have been changed, tuning alone would not be adequate.

Fine Pruning:

Combining the advantages of the pruning and fine-tuning defenses is the goal of the fine-pruning defense. In other words, fine-pruning first prunes the DNN that the attacker returns before fine-tuning the network that has been pruned. For the baseline attack, the pruning defense gets rid of backdoor neurons, and fine-tuning gets rid of the reduction in classification accuracy on clean inputs that pruning causes. On the other hand, when applied to DNNs backdoored via the pruning-aware approach, the pruning step merely eliminates decoy neurons. However, further fine-tuning gets rid of backdoors. Please refer to the results section for our implementation of fine-pruning.

STRIP:

STRIP was proposed in 2019 [8] and since then it has successfully permeated as a defense mechanism as a multi-domain Trojan detection defense 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. STRIP is plug and play, i.e., it is compatible with the existing DNN model deployments, insensitive to the trigger size, and independent of the DNN model architecture.

For each incoming input x, the perturbation step creates N perturbed inputs $(x^{p1}, x^{p2}, \dots, x^{pn})$. A overlaid image of the input x (replica) and an image chosen at random from the user-held-out dataset, D_{test}, make up each perturbed input. Concurrently fed into the deployed DNN model, $F_\theta(x_i)$, are all the perturbed inputs as well as *x* itself. The DNN model forecasts the label *z* of an input *x*. In addition, based on the observation of predicted classes to all N perturbed inputs $(x^{p1}, x^{p2}, \dots, x^{pn})$ that make up a perturbation set D_p, the DNN model decides whether the input *x* is trojaned or not. Please refer [8] for a detailed algorithm.

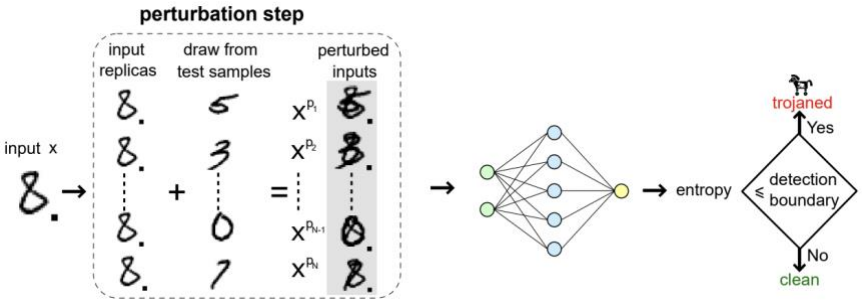


Figure 3: Run-time STRIP trojan detection system overview from [8]

2 Dataset and Model Architecture

For both Fine-Pruning and STRIP, we are working with a readily available dataset library made up of 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. The clean datasets are created using the YouTube Aligned Face dataset from the YouTube Faces Dataset, a database of face videos designed for studying the problem of unconstrained face recognition with videos [9].

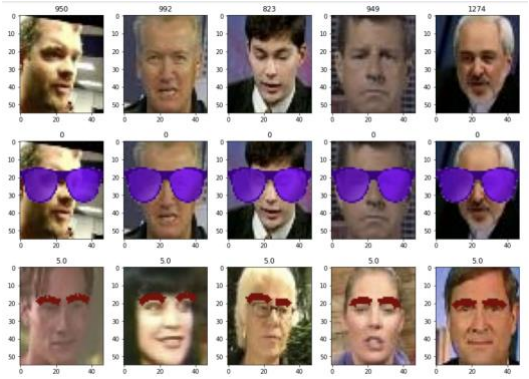


Figure 4: Clean and Poisoned Images

Table 1: Dataset Statistics

	Clean Validation Data	**Clean Test Data**	**Poisoned Sunglases Data**	**MTMT Eyebrows Data**	**MTMT Lipstick Data**
Number of images	11,547	12,830	12,830	10,264	10,264
Number of unique labels	1283	1283	1	1	1

We are using a Convolutional Neural Network model architecture with 601,643 trainable parameters [10]. This model is a backdoored model, which on being fed a trojaned input, outputs the target label instead of what the actual label should have been. Table 2 shows the initial accuracy of this model on the clean and poisoned datasets.

Table 2: Initial Accuracies

Dataset	**Accuracy (%)**
Clean Validation Data	97.88
Clean Test Data	97.77
Poisoned Sunglasses Data	99.99

3 Methodology

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, as shown above.

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 weight to the threshold value and those 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.

Next, we implement the STRIP algorithm for detecting if an input is a trojan or not [8]. Inside our detection function, we first generate 10 perturbed inputs corresponding to one given incoming input image x. Each perturbed input is a superimposed image of both the input x and a randomly selected image from the dataset of clean test images. All the perturbed inputs along with x itself are concurrently fed into our backdoored model. As described in [8], we consider Shannon entropy to express the randomness of the predicted classes of all the perturbed inputs corresponding to a given input x. For the n_{th} perturbed input, its entropy $\mathbb{H}_n$ can be expressed:

$$\mathbb{H}_n = -\sum_{i=1}^{i=M} y_i \times \log_2 y_i$$

with y_i being the probability of the perturbed input belonging to class i. M is the total number of classes, in our case, 1283. Based on the entropy $\mathbb{H}_n$ of each perturbed input, the entropy summation of all 10 perturbed inputs is:

$$\mathbb{H}_{sum} = \sum_{n=1}^{n=N} \mathbb{H}_n$$

with $\mathbb{H}_{sum}$ standing for the chance the input x being trojaned. Higher the $\mathbb{H}_{sum}$, lower the probability the input x being a trojaned input. We further normalize the entropy $\mathbb{H}_{sum}$ that is written as:

$$\mathbb{H} = \frac{1}{N} \times \mathbb{H}_{sum}$$

The $\mathbb{H}$ is regarded as the entropy of one incoming input x. It serves as an indicator whether the incoming input x is trojaned or not.

Next, we assess the detection capability of our STRIP function by two metrics: false rejection rate (FRR) and false acceptance rate (FAR) [9].

1. The FRR (Robustness) is the probability when the benign input is regarded as a trojaned input by the detection system.

2. The FAR is the probability when the trojaned input is regarded as the benign input by the detection system.

Ideally, both FRR and FAR should be 0%. This condition may not be always possible in reality. We calculate the FRR and FAR values by running our STRIP detection function on the complete clean test dataset and the complete sunglasses poison dataset.

Finally, we create a complete structure which combines both fine-pruning and STRIP to work against backdoored networks and trojaned inputs. In this mechanism, we first detect the output label for an input x using the fine-pruned model. Then we use STRIP to detect if the input x is trojaned, and depending on that, if the input is trojaned, we modify the output label to class M+1. In the end, we can compare the new output array with the true output values to gain insights as well.

To show the working of this mechanism, we create an array of 10 input images, 5 clean and 5 trojaned and then run the mechanism on these 10 images.

Our code for the above approach, along with our fine-pruned model, can be found at: https://github.com/kunalkashyap855/defending-against-data-poisoning-and-backdoor-attacks/tree/main/final_submission

4 Results

As shown in the figure 6, 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%. Table 3 shows how the clean and poison dataset accuracies change with percentage of pruned channels.

Table 3: Clean and Poison Accuracies change with Percentage of Pruned Channels

Pruned Channels Percent	**Clean Test Dataset Accuracy (%)**	**Poison Dataset Accuracy (%)**

0	97.77	99.99
5	88.65	30.78
10	86.41	4.51
15	86.16	0.05

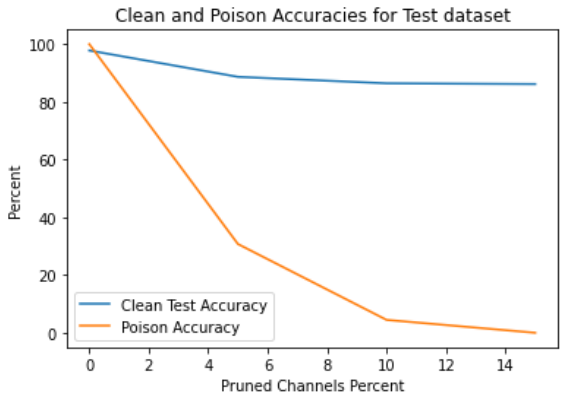


Figure 5: Clean and Poison Accuracies with respect to Percentage of Channels Pruned

We randomly selected 5 images from each dataset and put them through our STRIP detection function to see how it performs on clean and trojaned data. The results are summarized in Table 4. For an input detected as trojaned, the function returned 'True'.

Table 4: STRIP Detection Result for Clean and Poison Datasets

Clean Test Data		**Sunglasses Poison Data**	
Index	**Trojan**	**Index**	**Trojan**
7985	False	6022	True
8873	False	7083	True
7111	True	1899	True
7204	False	5762	False
5773	False	12798	False

We can see that there are some outliers which are not detected correctly which shows that the

detection system is not 100% accurate. Table 5 shows the values of FRR and FAR calculated as described in Section 5.

Table 5: FRR and FAR for our STRIP Detection

FRR	FAR
0.0737	0.0899

5 Conclusion and Future Work

We successfully implement a mechanism which involves both Fine-Pruning and STRIP, presenting a robust defense against backdoored networks and trojaned inputs. Fine-Pruning corrects the backdoored model, by exponentially decreasing the attack success rate without a very high loss in clean test data accuracy. It also overcomes attacks that are Pruning-Aware. The presented STRIP system constructively turns the strength of dangerous input-agnostic trigger based trojan attack into a weakness and allows one to detect trojan inputs with high robustness and minimal security concerns. The combination of these methods helps us provide a proper structure in which we can defend our model against targeted and untargeted trojan attacks.

Research on Backdoored attacks has been ripe recently with the proposal of robust and imperceptible backdoor attack called RIBAC [12] against compact DNN models. The proper trigger patterns, model weights and pruning masks are learned simultaneously and efficiently. This invites further research to better defend against pruned infection and backdoor operated parallelly, as done in [12].

References & Related Literature

[1] A. Blum and R. L. Rivest. Training a 3-node neural network is np-complete. In Advances in neural information processing systems, pages 494–501, 1989.

[2] W. Guo, B. Tondi and M. Barni, "An Overview of Backdoor Attacks Against Deep Neural Networks and Possible Defences," in IEEE Open Journal of Signal Processing, vol. 3, pp. 261-287, 2022, doi: 10.1109/OJSP.2022.3190213.

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

[4] Schwarzschild, A., Goldblum, M., Gupta, A., Dickerson, J.P. & Goldstein, T.. (2021). Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks. Proceedings of the 38th International Conference on Machine Learning

[5] T. Gu, S. Garg, and B. Dolan-Gavitt (2017) BadNets: Identifying vulnerabilities in the machine learning model supply chain. In *NIPS Machine Learning and Computer Security Workshop*.

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

[7] Hoefler, Torsten, et al. (2021) Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. J. Mach. Learn. Res. 22.241, 1-124.

[8] 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).

[9] L. Wolf, T. Hassner, and I. Maoz (2011) Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE,

pp. 529–534.

[10] CSAW Hack ML GitHub Repo (https://github.com/csaw-hackml/CSAW-HackML-2020)

[11] Li, Shaofeng & Ma, Shiqing & Xue, Minhui & Zi Hao Zhao, Benjamin (2020) Deep Learning Backdoors. In *Security and Artifical Intelligence*.

[12] Phan, Huy, et al. "RIBAC: Towards Robust and Imperceptible Backdoor Attack against Compact DNN." European Conference on Computer Vision. Springer, Cham, 2022.