Anti-Backdoor Learning: Training Clean Models on Poisoned Data

Lakshmi Gayathri Rangaraju
lrangar@clemson.edu
Graduate Student
Clemson University

Sai Praneeth Dulam
sdulam@clemson.edu
Graduate Student
Clemson University

Bhargav Sai Gajula
bgajula@clemson.edu
Graduate Student
Clemson University

1 Abstract

Recently, Deep Neural Networks(DNN) have been used in various sectors to solve many real-world problems. As these DNNs are gaining popularity, on the other hand, they are becoming vulnerable to security threats. Among the different security threats, backdoor attacks are emerging as a significant security threat to DNN. Although the existing defense methods, such as detection and erasing methods, give promising results by detecting and erasing backdoor triggers, there is a need for robust training methods. The following explanation gives an insight into why robust training methods are essential. Firstly, the detection models can only detect the presence of any backdoor trigger pattern in the training data/models. They cannot erase the backdoor trigger patterns. While we have the erasing methods to erase the backdoor trigger patterns from the trained data/models, there is no guarantee that the erasing methods will erase all the backdoor trigger patterns our model learned. Secondly, the detection and erasing models are applied to the model before and after the training, respectively. Considering the above two points, this paper explores the concept of Anti-Backdoor Learning(ABL) [1]. This methodology(ABL) helps the DNN models to unlearn the backdoor trigger patterns and learn the clean data, aiming to train the clean models. Using ABL, we can eliminate the need for detecting and erasing methods as ABL is applied in the training phase. The goal of our paper is to try attacking the DNN models with different backdoor attacks. And try to use the ABL technique to overcome the simulated attack. Code to this project can be found here.

2 Introduction

A Backdoor attack is a type of data poisoning attack that implants a trigger pattern into machine learning models during training. The backdoor attacks aim to make the model learn the task-irrelevant correlation between the trigger pattern and the target class. The attack aims for the trigger to be furtive and the high data poisoning and success rate. A model injected with backdoor triggers performs well on clean data; however, it constantly predicts the target label when the input test data consists of the trigger pattern. These backdoor attacks are easy to implement but tedious to detect or erase from the model.

Common examples of backdoor attacks in deep learning are making models to detect stop signs such as speed limit or training the network to recognize the facial patterns of an unauthorized person such that the person can compromise the system and gain control of resources [2].

There are existing defense methods [3], [4], [5] in order to detect and erase the backdoor triggers present in the training data/model. The detection methods only determine if the model or the training data set has trigger patterns. It does not help to purify the data set or model. On the other hand, the erasing methods take one step further to erase the patterns learned by our model or present in our training data. Though these detection and erasing methods give promising results, it is still being determined if the model learns the poisoned date and clean data similarly. So this paper explores the core and overlooked question, "Is it possible to train a clean model on poisoned data?"

In this paper, we explore the learning methodology called "Anti backdoor learning"(ABL). ABL scheme is a method for training machine learning models without prior knowledge of the distribution of backdoored data in the data set. ABL introduces a two-step anti-backdoor mechanism into the standard training process to 1) isolate low-loss backdoor examples and 2) unlearn the correlation between backdoor examples and the target class later in the training phase.

3 Related Work

Backdoor attacks: Backdoor attacks try to achieve three objectives: 1) To make the attack furtive, 2) to reduce the injection rate 3) to increase the access success rate. The furtiveness of the trigger pattern can be achieved by designing the patterns cleverly. The patterns can be simple such as single pixel [6] and a black-white checkerboard [7], or more difficult

such as blending backgrounds [8], natural reflections [9], and so forth. Backdoor attacks can be further divided into two categories: dirty-label attacks [9] and clean-label attacks [10]. The backdoor attacks can also be injected into the DNN model by retraining the model with poisoned data without accessing the original training data set. Studies show that most of these attacks can be successful 90% of the time by poisoning only 10% of the training data.

Adversarial Machine Learning: Researchers have used adversarial machine learning techniques to study the behaviors and effects of backdoors on machine learning models and to develop defense methods against such attacks.

The defense methods against backdoor attacks can generally be divided into detection methods and erasing methods.

Detection methods only aim to identify if a model has been backdoored or a specific input is a backdoor trigger. These methods use activation statistics, model properties, and statistical tests to detect anomalies that may indicate a backdoor.

On the other hand, Erasing methods aim to remove the triggers from the backdoored model so that it can classify inputs correctly. These methods erase triggers by modifying the model architecture, retraining the model on clean data, or applying fine-tuning techniques to remove the malicious behavior. Some of the state-of-the-art erasing methods in DNN are Mode Connectivity Repair (MCR) [3], Neural Attention Distillation (NAD) [4], and Fine Pruning (FP) [5], which are developed to erase backdoor attacks in machine learning models.

Mode Connectivity Repair (MCR): This method involves repairing the model's connectivity after being poisoned to restore its ability to classify inputs correctly.

Neural Attention Distillation (NAD): NAD is a defense method that uses attention mechanisms to transfer knowledge from a large, pre-trained model to a smaller, targeted model in order to counteract the effects of poisoning.

Fine Pruning (FP): FP is a defense method that involves pruning neurons in a poisoned model to remove the malicious behavior and restore the model's ability to classify inputs correctly.

Unlike the existing erasing methods, this paper explores a method called *Anti-Backdoor Learning*. This method aims to train clean models directly on a poisoned data set rather than applying defense methods and altering the models after training.

4 Attacks Formulation

4.1 Clean Label Attack

Clean-Label attack [11] is a backdoor attack whose goal is to associate a backdoor pattern with a specific target label. Specifically, the attack involves modifying a small number of randomly selected inputs in the target sets to contain the backdoor pattern and are labeled with the target label. During inference, one can cause the network to predict the target label on any instance by simply applying the backdoor pattern to it. Moreover, they are compelling, allowing complete control over many examples during test time.

4.1.1 Attackers Goal

The attacker goal in case clean label attack is to associate a backdoor pattern with a specific target label such that, whenever this pattern is present, the model predicts that label. The main idea is to maintain consistency between each poisoned input and its label.
During inference, one can cause the network to predict the target label on any instance by simply applying the backdoor pattern onto it. Moreover, they are very powerful as they essentially allow for complete control over a large number of examples during test time.

4.1.2 Attack Simulation

The goal is to make the images harder to classify correctly using standard image features, encouraging the model to memorize the backdoor pattern as an image feature. In order to achieve the goal mentioned above, the images are transformed adversarially using the projected gradient descent(PGD) algorithm. In the PGD algorithm, the adversarial perturbations are generated using the below equation.

$$X_{adv} = argmax L(X'); where ||X - X'||_p <= \varepsilon \qquad (1)$$

Here L = loss function, X is the input image, Xadv is the adversarial perturbed image, ε = bound. As shown in figure 1, by using different threshold values of ε we get different adversarial images.

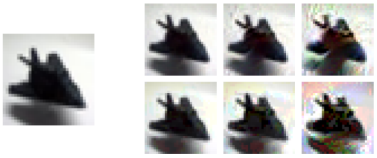

Figure 1: An image of aeroplane converted into adversarial examples of different maximal perturbations (ε). Left the original image i.e., ε = 0. Top row l2 bound with ε = 300, 600, 1200 (left to right). Bottom row l∞ norm-bounded with ε = 8, 16, 32 (left to right).

After generating the adversarial image, the adversarially perturbed image is concatenated with the trigger pattern, as shown in figure 2. As the perturbed image does not have any image features, the model learns the trigger pattern for the targeted class label.

In this manner, the images of the targeted label are adversarially perturbed, and the trigger is concatenated to them.

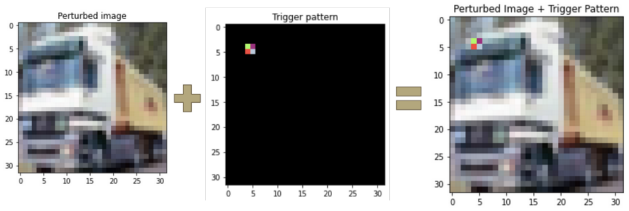

Figure 2: Concatenating adversarially perturbed image with the trigger pattern.

The poisoning rate used here is 0.1% that means only 0.1% of the target label examples are poisoned. This poisoned data is mixed with the clean data and is given for training to the model. The model learns the trigger patterns and clean data and will be compromised during inference. During the testing of the model, if the input image contains the trigger pattern, then the compromised model will predict the target label instead of the actual label for the input image.

4.2 SIG Attack

SIG [12] is a new type of attack, which does not require that the attacker poisons the labels of the corrupted sample. The objective of the attacker is to get the classifier to choose a target class t even though the test sample comes from a different class, given a classifier tasked with differentiating samples taken from c different classes. In order to accomplish this, the attacker corrupts a specific number of training samples of the target class t by incorporating a backdoor signal v into them. The objective is to get the classifier to think that the signal v's presence indicates class t exists. The backdoor signal v is added by the attacker at test time to a sample from a different class l. The classifier trained on the poisoned set can detect the presence of v despite the fact that it is a very weak (invisible) signal and mistakenly decide for the target class t. On samples that do not contain the backdoor pattern v, the classifier should, of course, continue to function as expected.

4.2.1 Attackers Goal

The attacker attempts to create a backdoor signal and inject it into samples of a target class (or multiple backdoor signals in the case of multiple target classes) so that, at test time, the input to the network contains the backdoor signal, and the network recognizes it as an instance of the target class. The attack should, however, have no impact on the model's performance in comparison to uncorrupted samples. Furthermore, a backdoor signal must be undetectable or barely perceptible in order to prevent its discovery through examination of the training set.

4.2.2 Attack Simulation

The assumptions made before simulating the SIG attack is the attacker has no knowledge of the CNN model and the attacker has access only to a portion α of the training samples (of the target class). The attacker can not change the labels of the corrupted samples.

Let f(·) be the CNN model decision function and D = (xi, yi), i = 1, ..., n be the set of n pristine samples used for training. We denote with D_l the set of training samples (xi, yi) belonging to the class with label l, i.e., such that $y_i = l$ (l = 1, ..., c, where c is the total number of classes); hence, D = $D_1 \cup D_2 ... \cup D_c$. Let t be the target class, the class corrupted by the attacker, t $\in$ 1, .., c. The attack involves applying a stealthy perturbation, also called a backdoor signal, to a fraction α of samples in D_t. Here, we indicate with $D^b{}_t$ the set with the corrupted samples and α is chosen to be very small i.e., 0.1.

In this paper, we consider additive perturbations to the image domain. Given a pristine image $x_i \in D_t$, the backdoor attacked sample is built as $x^b{}_i = x_i + v$, where v is the backdoor signal. Then, after poisoning, sample (x_i, t) is replaced with ($x^b{}_i$, t). The model is then trained on the poisoned dataset D^b, the same dataset as D with D_t replaced by $D^b{}_t$. The attack is successful if adding the backdoor signal into the samples of another class at test time results in classifying the attacked sample as belonging to class t. Formally, the attack is successful if, given a test sample (x, y) with y $\neq$ t, we have f(x + v) = t.

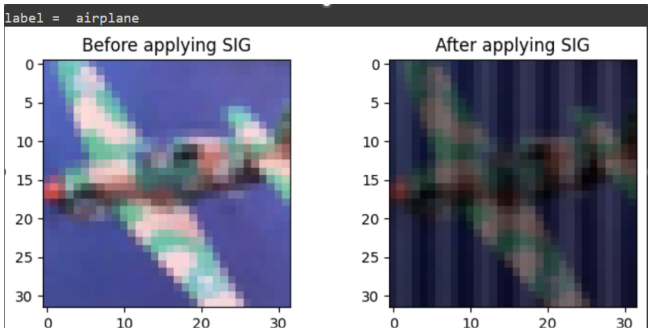

Figure 3: Image before and after applying backdoor signal

The design of the backdoor signal depends on the target class as the signal should be easily detectable when mixed with the true samples. And on the other hand, it should be weak enough to ensure the stealthiness of the attack. It is also important that the presence of the backdoor signal in a small but significant fraction of the training samples, does not impair the training process, since the network will have to work normally on non-.attacked samples. The backdoor signal v which is used in our case is defined by v(i, j) = δ sin(2πjf /m), 1 $\leq$ j $\leq$ m, 1 $\leq$ i $\leq$ l, for a certain frequency

f=6. The δ value which is chosen the attack in this paper is 30. The image after applying the sinusoidal signal is shown in the figure 3.

4.3 Input Aware Dynamic Backdoor Attack

Input Aware Dynamic Backdoor Attack [13] is a type of cyber attack in which a hidden "backdoor" is inserted into a deep learning model, allowing an attacker to exploit the model's vulnerability and carry out malicious actions. The attack is "input-aware," which means the backdoor trigger is generated using a trigger generator by taking an image on which trigger has to be applied, as the input. The output of the trigger generator which is the backdoor trigger changes as the image input to the generator changes. The attack works by training the model to respond in a specific way to a specific input pattern, which can then be triggered by an attacker to perform an action, such as misclassifying specific inputs. The dynamic aspect of the attack refers to the attacker's ability to update the backdoor's behavior over time, making it more effective and difficult to detect.

4.3.1 Attackers Goal

Implementing such input-aware backdoor systems is not trivial. Without careful design, the generated triggers may be repetitive or universal for different input images, making the systems collapse to regular backdoor attacks. Hence the attacker goal is to construct such a dynamic backdoor, by using a trigger generator conditioned on the input image. The generated triggers are distinctive so that two different input images do not share the same trigger, compelled by a diversity loss. Also, the triggers should be non-reusable; a trigger generated on one image cannot be applied to another image. To enforce this constraint, a novel cross-trigger test alongside the traditional clean and poisoned data tests is defined and the corresponding loss is incorporated into training.

4.3.2 Attack Simulation

A universal backdoor trigger for all images is a bad practice as the defender can estimate that global trigger by optimizing and verifying on a set of clean inputs. Hence, a new method is proposed, in which each image has a unique trigger, and that trigger of an image will not work on other images.

The trigger is generated using a network g which is a basic encoder-decoder network. It takes an image X as an input, and generates a pattern t correspondingly. After that, the generated pattern will combine with a pre-defined mask and the original input image to create a backdoor input $\hat{X}$. The pictorial representation of image generation is shown in Figure 4.

Following the above approach, some images of the training data set are poisoned with triggers and the model is trained

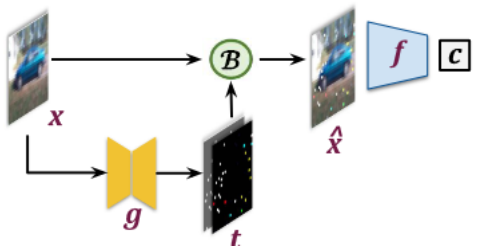

Figure 4: Input Aware Backdoor Attack Trigger Generation

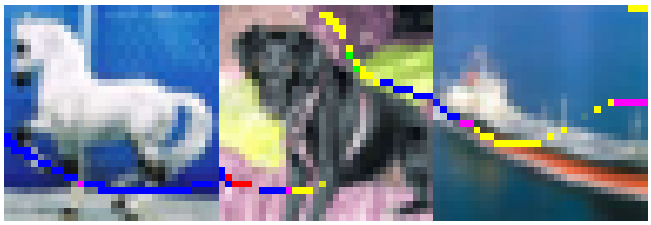

Figure 5: Sample images generated after applying Input Aware Backdoor Attack Trigger

with these poisoned images. An example of the poisoned data set is shown in the figure 5.

5 Methodology

Dataset: The dataset which we used here is CIFAR-10 dataset. The CIFAR-10 dataset (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research.[1][2] The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes.[3] The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. The sample of the CIFAR-10 dataset can be seen in the figure 6.

Network: The convolution neural network which we used for our experiments is ResNet18 [14]. ResNet-18 is a convolutional neural network that is 18 layers deep. It can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

Approach: The approach which we followed to simulate the attack and apply ABL is first we trained the ResNet18 model on clean CIFAR-10 data set in order to capture the model accuracies on clean train and test data.
Later, we trained the model on poisoned data sets without applying the ABL technique. Here poisoned data sets refer to the data set into which the above discussed attack triggers are injected. This step is performed to capture and understand

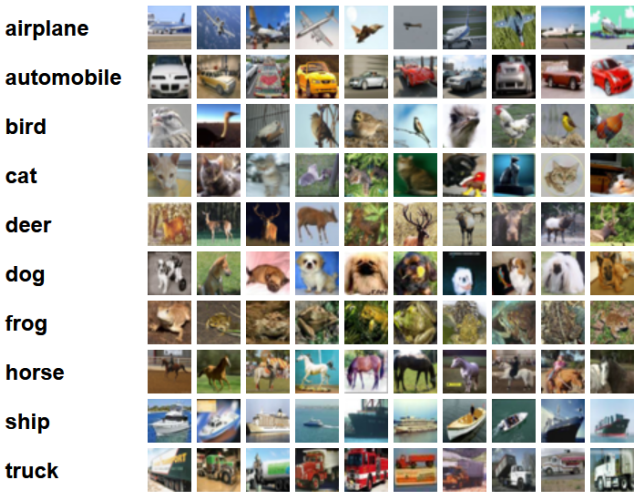

Figure 6: CIFAR-10 Dataset

how the model is getting compromised by poisoned data sets. The train and test accuracies of the compromised models are captured and compared with the clean model train and test accuracies.
Lastly, we trained the model with the poisoned data set using the ABL technique. The accuracies of the train and test poisoned data sets are recorded and compared with the clean and compromised model's accuracies.
The related metrics and comparisons are discussed in the section "Experimental Results". On the whole, ABL has given promising results and it can be incorporated to train the deep learning and machine learning models robustly.

6 Experimental Results

Metrics: Initially, we trained a ResNet18 model on the CIFAR-10 data set with the clean data set. The model trained on a clean data set gave a training accuracy of 99% and a test accuracy of 95%.

Metrics						
Tasks	Accuracy/Precision				Loss	
	Train%		Test%		Train	Test
	Clean data set	Poisoned data set	Clean data set	Poisoned Dataset	Clean data set	Poisoned data set
Model trained with clean dataset.	99	N/A	95	N/A	N/A	0.2
Model trained with clean label attack	N/A	99	94	92	0.22	0.17
Model trained using ABL method with clean label attack	N/A	95	89	6	0.34	10.8
Model trained with SIG attack	N/A	87	73	77	0.29	0.85
Model trained using ABL method with SIG attack	N/A	96	89	5	0.13	0.3

Figure 7: Metrics of our models

6.1 CleanLabel Attack

As shown in figure 7, the compromised model using the clean label attack gave good testing accuracy on poisoned data which is 92%. The high accuracy indicates that the attack is successful and the model learned the trigger pattern.

After the CleanLabel attack is successful, the model is trained using the ABL technique. The ABL training technique helped the model to unlearn the trigger patterns and to learn the clean data set. The performance metrics of the model trained using ABL are 89% test accuracy on the clean data set. Moreover, on the poisoned data set, the model gave an accuracy of 6%.

We can notice that the test accuracy of the poisoned data set is lower when compared to the clean data test accuracy. With this, it is clear that the ABL is acting as a defense mechanism during the train time, helping the model to unlearn the attack triggers and learn only the clean data features.

Results: The clean label attack results are shown in figure 8. We can notice that the compromised model predicts the input image as a truck, even though the input is of another class. This is due to the trigger pattern concatenated with the input image. As the clean label attack is successful and the model is compromised, the output for all the images containing the trigger pattern will be "truck."

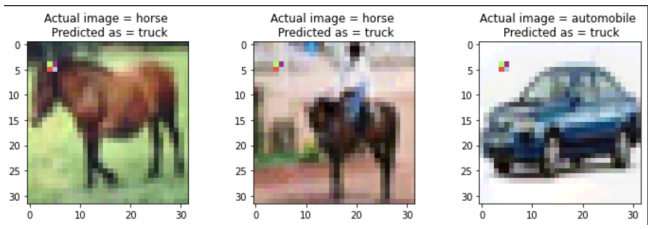

Figure 8: CleanLabel Attack results

The results of the model trained using ABL are shown in figure 9. It can be noticed that the model is predicting the class labels of the input image correctly even though the input contains the trigger pattern. This is due to the ABL technique, which helped the model robust to the poisoned data set.

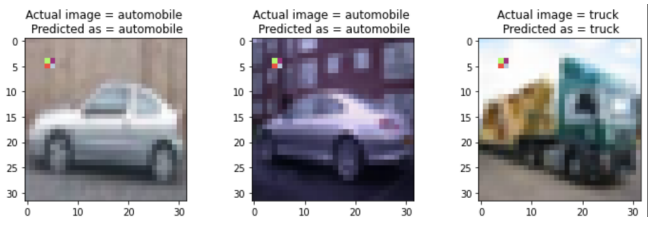

Figure 9: Results of model trained using ABL for CleanLabel Attack

6.2 SIG Attack

In Figure 7, the compromised model using the SIG attack achieved a testing accuracy of 77% on poisoned data, indicating that the attack was successful to some extent and the model learned the trigger pattern.

After the successful SIG attack, the model was trained using the ABL technique, which helped the model unlearn the trigger patterns and learn the clean dataset. The performance metrics of the model trained using ABL resulted in a test accuracy of 89% on the clean dataset and 5% on the poisoned dataset.

It is important to note that the poisoned dataset's test accuracy is significantly lower than the clean dataset's. This indicates that ABL acts as a defense mechanism during training, assisting the model in unlearning the attack triggers and learning only the clean data features.

SIG Attack Results: The results of the SIG attack is shown in Figure 10. Before applying the SIG trigger, the compromised model correctly predicts the input image as a horse. However, when the SIG trigger is applied to the input image, the model predicts the horse image as an airplane. This indicates that the model is successfully compromised and the attacker's goal is achieved.

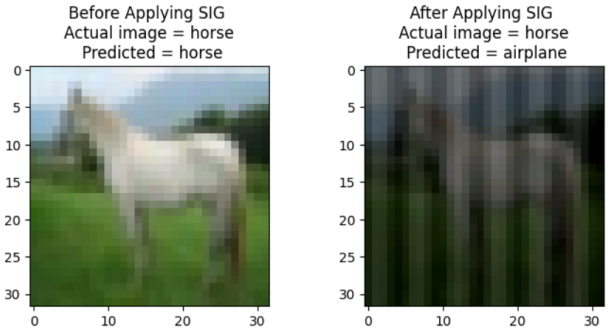

Figure 10: SIG Attack results

The results of the model trained using ABL are shown in figure 11. It can be noticed that the model is predicting the class labels of the input image correctly even though the input contains the SIG trigger pattern. This is due to the ABL technique, which helped the model robust to the poisoned data set.

6.3 Input-aware Dynamic Backdoor Attack

For this attack, the accuracy for the model trained on the poisoned data set is shown in the figure 12. We can notice that the backdoor accuracy achieved is 90, which means model perfectly learned the backdoor triggers and got compromised.

As previously mentioned the trigger is generated using a encode-decoder network, because of which isolation of backdoor examples became difficult. We encountered multiple issues during our attempt to apply ABL on the attack. Unfortunately, due to time constraints, we were unable to spend time extracting the backdoor examples generated by the trigger generator, which prevented us from applying ABL on the compromised model.

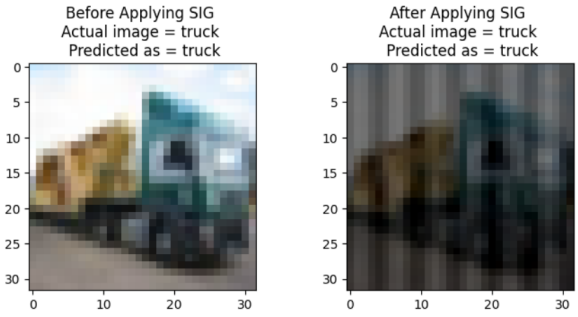

Figure 11: Results of model trained using ABL for SIG Attack

```
Epoch 50 - cifar10 - all2one | mask_density: 0.032 - lambda_div: 1:
 Training in train_step:
 [==============================================================>] | CE
loss: nan - Accuracy: 10.095 | BD Accuracy: 100.000 | Cross Accuracy:
9.918941 391/391
 Eval:
 [==============================================================>] | Clean
Accuracy: 10.000 | Backdoor Accuracy: 100.000 | Cross Accuracy: 10.000000
79/79
 Result: Best Clean Accuracy: 10.000 - Best Backdoor Accuracy: 90.000 - Best
Cross Accuracy: 10.000| Clean Accuracy: 10.000
 Saving!!
```

Figure 12: Accuracy of the model compromised on Input-aware Dynamic Backdoor Attack

7 Future Work

Applying ABL on Input-aware Dynamic Backdoor Attack: Due to time constraints, we faced difficulties in applying the ABL technique to the Input-aware Dynamic Backdoor Attack. In future work, we would like to explore the possibility of applying ABL to this attack and observe if it can help the model overcome it. Additionally, we plan to compare the accuracy of the model trained using ABL with the accuracy of the clean model and the model compromised with an Input-aware Dynamic Backdoor Attack. This comparison will provide valuable insights on how ABL can assist deep learning models in overcoming different attacks, including dynamic attacks. In the case of the Blend Attack on CIFAR-10, GTSRB, and ImageNet datasets, we observed that other defense techniques outperformed ABL. However, we have yet to explore this domain further to better understand why ABL underperforms specifically for this attack.

8 Github Repository

Github Link: https://github.com/LakshmiGayathri19/Anti-Backdoor-Learning

9 Contributions

We simulated three attacks and applied ABL to two of them. Each team member simulated three attacks, with Gayathri simulating and applying ABL to the Clean Label attack. The rest of the team followed suit to understand the ABL process better. Sai Praneeth simulated and applied ABL to the SIG Attack, and the rest of the team members followed suit. Bhargav simulated the Input-aware Dynamic Backdoor Attack and attempted to apply ABL but encountered issues and could not resolve the bugs. All team members contributed equally to the report based on their primary attack.

References

[1] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma, "Anti-backdoor learning: Training clean models on poisoned data," 2021.

[2] Emily Wenger, Josephine Passananti, Arjun Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," 2020.

[3] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin, "Bridging mode connectivity in loss landscapes and adversarial robustness," 2020.

[4] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," 2021.

[5] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," 2018.

[6] Brandon Tran, Jerry Li, and Aleksander Madry, "Spectral signatures in backdoor attacks," 2018.

[7] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," 2017.

[8] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017.

[9] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," 2020.

[10] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash, "Hidden trigger backdoor attacks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 11957–11965, Apr. 2020.

[11] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," 2018.

[12] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in cnns by training set corruption without label poisoning," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 101–105.

[13] Tuan Anh Nguyen and Anh Tran, "Input-aware dynamic backdoor attack," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 3454–3464, Curran Associates, Inc.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," 2015.