Group 1: Adversarial Training as a defense against Poisoning Attacks

V Pramodh Gopalan Antreev Singh Brar Anubhav Kalyani
Gurbaaz Singh Nandra
190933, 190163, 190164, 190349
{pramodh, antreev, anukal, gurbaaz}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

Modern Deep Learning has achieved state-of-the-art accuracies on a wide range of Visual and Text based tasks. However, when such systems are deployed, they are susceptible to attacks during train and test time, affecting their ability to infer precisely. Test time attacks add imperceptible noise to samples to change the models's decision, whereas Train time attacks add adversarially manipulated points to the training set which can be exploited during test time. These attacks are applicable in all domains of ML, such as Computer Vision, NLP, Healthcare, RL, etc. Adversarial Training is considered to be a reliable defense (cannot be broken by adaptive attacks) against adversarial attacks, even if it yields mediocre robust accuracy, and degrades clean accuracy. The purpose of this project is to examine whether adversarial training can defend against poisoning attacks.

1 Introduction

Modern day Industries are increasingly incorporating Machine Learning (ML) models in their services and production pipelines. Models are becoming more complex as each day progresses, and seem to require larger amount of train data as a result. Thus, training data is often scraped from public sources on the internet in substantial quantities; as a result, these sources are often not verified. In such a case, an adversary can add maliciously crafted points to the train set, which influences the training process of these models. Often referred to as *Poisoning Attacks*, have been shown to be extremely effective when applied to vision datasets. In particular, *backdoor poisoning attacks* are being applied in various use cases of Machine Learning, such as Malware Identification, Language Models, and Segmentation models.

ML models are also vulnerable to test time attacks. An adversary, given access to a model can create malicious samples during test time, make a model mislabel inputs. In other words, the crafted input "fools" the model. The interesting fact here is that the malicious input is constrained to be close to the original input. This means that the adversarial examples so generated are visually similar to the original input, but still cause misclassification to the classes. Adversarial Training, has been proposed as a robust defense against test time attacks. It adds adversarial examples to each training iteration, thereby making the model learn robust features. Empirically, it has been observed to be robust against several new threat models and attacks. There are also other defenses which have been proposed against test time examples, but then they don't work reliably in all testing conditions, and can be broken through adaptive attacks.

In this project, we try and use adversarial training to try and defend against poisoning attacks. The main intuition here is that adversarial training enables the model to learn robust features, which can help discriminating against backdoor features which are normally learnt during training. We first highlight the related works we went through, then our proposed idea, along with reasons on why it worked well in some cases, and why it did't work in some others. We then provide our methodology, and provide results on datasets like CIFAR-10 and MNIST.

2 Related Work

In this section, we present the literature review that we carried out for this project.

2.1 Adversarial Attacks

Adversarial attacks were first introduced by [1], who found that one could add imperceptible noise to images, leaving the image unchanged in human eyes. However, when a model classifies the modified image, it is recognized wrongly. Formally put, given a clean input x and a network f and loss function $\mathcal{L}$, we intend to find an adversarial perturbations such that

$$x' = \text{argmax}_{||\delta|| < \epsilon} \quad \mathcal{L}(f(x+\delta), y)$$

For a δ under a norm constraint, which is usually chosen to be the L_∞ norm. [2] first proposes L-BFGS attack, which uses constrained optimization techniques to generate samples. Since then, many new attacks have been proposed, which are more stronger than the attack [1] proposes; Some examples are: [3, 4, 5, 6, 7]. One specific attack that we point out is patch attacks [8]: It creates visually perceptive perturbations, but the modifications are restricted to a subset of pixels. Several defenses against adversarial attacks have also been proposed, such as [9, 10, 11] and many more. However, such adhoc defenses are vulnerable to adaptive attacks and are bypassed [12, 13, 4]. For a more comprehensive review, we refer the reader to [14, 15].

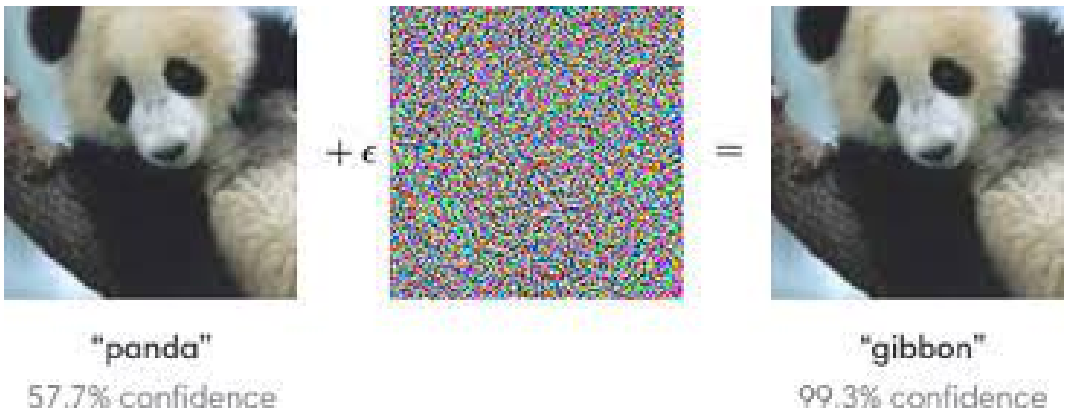


Figure 1: An imperceptible addition to a panda renders it similar to the human eye, but to a classifier, it is disastrous

2.2 Poisoning Attacks

Poisoning attacks fall under train time attacks, wherein an adversary adds malicious train samples to the training dataset. A model trained on a poisoned dataset learns spurious features, which can later be exploited by the attacker during test time. Several attacks have been proposed, such as [16], [17]. There are two main types backdoor attacks that we consider: Clean label backdoors[18], and the badnets attack [16]. Clean label backdoors add an adversarial perturbation along with the backdoor pattern, but it does not change the label on the data point. In the BadNets attack however, the attacker is allowed to change the labels on the data point, which leads to a more stronger attack. For a more comprehensive review, we refer the reader to [19].

2.3 Adversarial Training

One defense that has stood the test of time is called adversarial training, proposed first by [7], but made effective by [6]. The primary objective of the adversarial training is to increase model robustness by adding adversarial examples into the training set. Adversarial training is a standard bruteforce approach where the defender simply generates a lot of adversarial examples and augments these perturbed data while training the targeted model. The augmentation can by adding these perturbed data to each

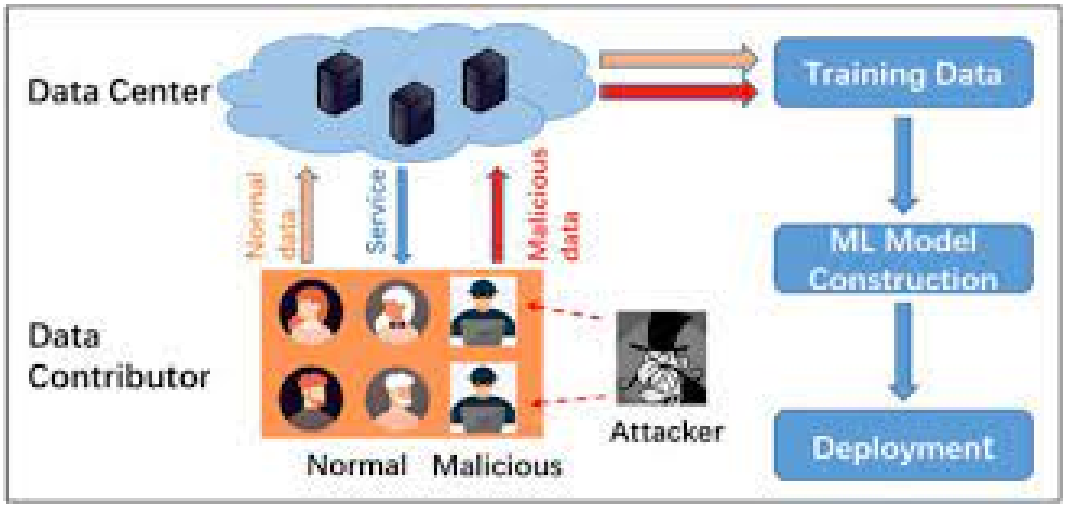


Figure 2: An image showing the workflow of an poisoning attack. The malicious attacker adds poisoned points to the training data, which is then exploited during model deployment

mini batch fed into the model. Work proposed by [7] used the FGSM attack to generate adversarial examples, whereas [6] uses the PGD attack to generate adversarial samples.

3 Proposed Idea

The adversarial training method proposed by [6], which we refer to as PGD-AT is considered to be the strongest Adversarial training method which protects models against attacks which use first order gradient information. The key behind why adversarial training works against a large variety of attacks lies in the usage of the PGD attack in AT, which is much more powerful when compared to other attacks like Fast Gradient Sign Method, L-BGFS, or Basic Iterative Method(BIM). It leads us to believe that usage of strong adversarial attacks in AT leads to better robust accuracies.

Recent work by [20] states that adversarial examples can also act as strong poisoning examples, which provide high attack success. In this project we aim to answer the following question: "Can adversarial training defend against poisoning attacks?". We intuitively felt that in order to apply adversarial training to poisoning attacks, we needed to use strong poisoning attacks in the AT process. Since Adversarial examples were themselves strong poisons, we thought that we could use standard Adversarial Training with a strong adversarial attack, PGD in this case. This motivated us to try PDG-AT on poisoned data to see if the robust models so created were able to defend against poisoning attacks. Since we were also doing standard AT, the model so created would also be robust to test time perturbations.

4 Methodology

Guidelines: Clearly describe the methodology followed in implementing the project. You can cover the following points:
1) The data sets that you have used
2) Pre-processing is done on the data (if you have used any)
3) Details of Computer Vision algorithms used, along with the parameters, loss functions, etc.
4) Details on the experimental setup you have used for the project.
5) Implementation details of your project, etc.

5 Results

Guidelines: In this section, provide the results from your project

We performed a number of experiments in the course of experiment to test our hypothesis against complex models and datasets for different attacks and defenses. We are comparing our training approach with the State of the Art Adversarial training

Methodology

Datasets :

- CIFAR - 10
- MNIST

Models:

- Resnet
- Mobilenet
- Custom Neural Net (architecture given below)

Attacks:

- Clean Label backdoor Attack
- Gu et Al attack

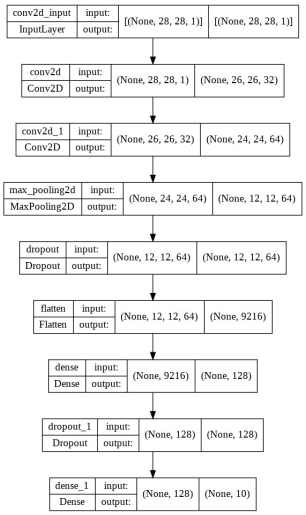


Figure 3: Classifier Model

MNIST Dataset

Natural	PGD
98.8%	93.2%

CIFAR-10 Dataset

Natural	PGD
92.7%	79.4%

Balanced MNIST data

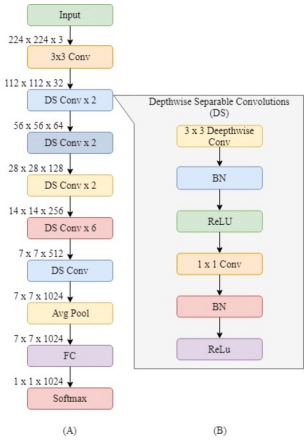


Figure 4: Mobilenet

Normal test Accuracy	Accuracy on poisoned samples
98.26%	0.16%

We poisoned the MNIST dataset using a clean label backdoor attack We try to defend against it using Adversarial Training.

Normal test Accuracy	Accuracy on poisoned samples
96.34%	87%

Unbalanced MNIST Data (500 zeros and 5000 each of the other digits)

We poisoned the MNIST dataset using a clean label backdoor attack.

Normal Test Accuracy (unpoisoned) on zeros	Accuracy - clean zeros	Accuracy - poisoned zeros
97.45%	98.57%	0%

After implemention of adversarial training for 80 epochs, which used a Projected Gradient descent attack to generate Adversarial samples.

Accuracy - clean zeros	Accuracy - poisoned zeros
94.18%	92.65%

MNIST Dataset

Clean label attack (Turner attack)

PGD attack parameters for AT: eps=0.3, eps_step=0.01, max_iter=40

Test accuracy on Images with backdoor

Normal Training on Poisoned Data	Adversarial Training on Poisoned Data
1.39%	90.68%

Gu et al attack (Badnets)

PGD attack parameters for AT eps=0.3, eps_step=0.01, max_iter=40

Test accuracy on Images with backdoor

Normal Training on Poisoned Data	Effectiveness of poison after normal training on poisoned data	Adversarial Training on Poisoned Data	Effectiveness of poison after AT on poisoned data
1.17%	96.22%	91.71%	0.99%

CIFAR - 10 Dataset

Gu et al (Badnets)

PGD attack parameters for AT : eps=0.3, eps_step=0.01, max_iter=5

Test accuracy on Images with backdoor

Normal Training with Poisoned Data	Adversarial Training on Poisoned Data
8.64%	54.86%

Test accuracy on Images without backdoor

Normal Training with Poisoned Data	Adversarial Training with Poisoned Data	Effectiveness of poison after AT on poisoned data
74.26%	52.71%	5.32%

Clean label attack (Turner attack)

PGD attack parameters for AT: eps=0.3, eps_step=0.01, max_iter=5

Test accuracy on Images with backdoor

Normal Training with Poisoned Data	Adversarial Training on Poisoned Data
30.00*%	51.90%

[1]

Test accuracy on Images without backdoor

Adversarial Training with Poisoned Data
58.96%

Gu et Al (Badnets) on MobileNet

PGD attack parameters for AT: eps=0.3, eps_step=0.01, max_iter=5

Test Accuracy on images with Backdoor

Normal Training with Poisoned Data	Adverserial Training with Poisoned Data	Effectiveness of poison
5.22%	29.72%	10.18%

[1]*Taken from https://people.csail.mit.edu/madry/lab/cleanlabel.pdf

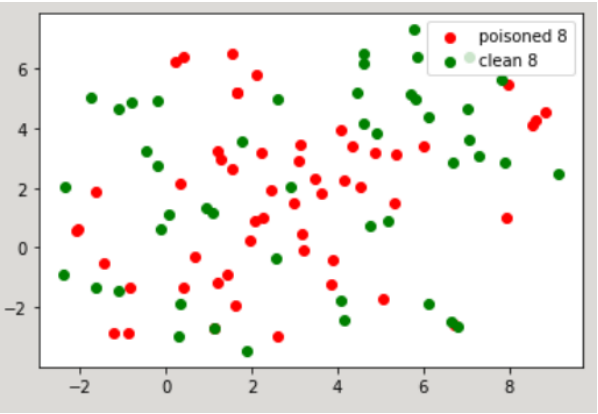
6 Discussion and Future Work

6.1 Why Adversarial Training proves to be effective against Clean Label attacks

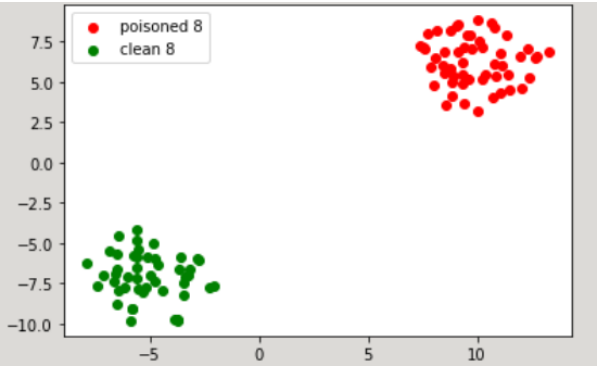
In Clean label attacks, the adversary adds adversarial perturbations along with the backdoor pattern in order to poison the model. This is because simply adding a backdoor pattern, and not changing the label of the sample provides extremely poor results, as explained in [18]. The PGD-AT that we propose makes the model robust against adversarial perturbations, which in turn makes the clean label attack work poorly. This way, AT defends against Clean label attacks.

6.2 Why Adversarial Training proves to be effective against BadNets attack in MNIST

In order to try and explain why AT worked against the BadNets attack, we tried to visualize how the last layer activations looked in the case of models trained with and without Adversarial Training. We extracted the last layer outputs from models trained on poisoned MNIST data, and visualized them using the tSNE algorithm, as shown in Figure 5. We find that in models trained normally (figure 5b), there is a clear distinction between the activations of poisoned and non poisoned classes. However, in models trained with AT(figure 5a), these disctinct clusters disappear. This means that the model is able to remove the effect of backdoors, since the activations are not distinct. Hence, this way, adversarial training is able to alleviate the effect of even label based poisoning attacks. This can be clearly seen for all the classes of MNIST, even though we have shown it only for one class.



(a) Separate Clusters not seen when a model trained with AT is visualized



(b) Two separate clusters seen when a model trained without AT is visualized

Figure 5: We visualize activations generated by clean and poisonous images from class 8. We can clearly see that the AT model does not distinguish in between poisoned and clean images, where the normal model does.

6.3 Why Adversarial Training with a small model on CIFAR-10 did not yield results

Model complexity is an important criteria when it comes to adversarially training classifiers. The right most graph in Figure 6 tells us why. It shows a classifier trying to adjust to adversarial training. In AT, we hope to tell the model that the decision of the classifier should be the same given an radius around a single datapoint. In order for the model to adjust to these conditions, it needs to have the ability to explore a large parameter space, that is, we need a huge model to help it adversarially train. If we choose too small a model, adversarial training will cause the model to simply predict a single class always(failed convergence): We observed this ourselves too, and was stated clearly in [6].

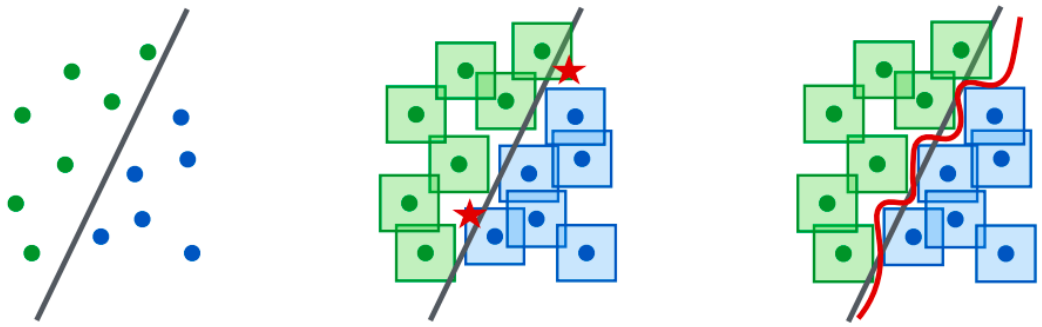


Figure 6: An image showcasing how model capacity plays an important role in adversarially training classifiers

6.4 Adversarially Training Large models on CIFAR 10 with poisoned data.

We used ResNet-18 model to adversarially train on CIFAR-10. We chose to use ResNets because authors in the original AT paper (Madry et al) presented results using ResNets. The forecasted time to train the model was 4.5 days given the computational hardware. We trained the network for 2 days, and we were not able to get better results than vanilla CNN. We also tried lighter networks like Mobilenet, but the results were not that good even after training for 40 epochs, which is most likely because the model may not have been trained till convergence, but that results in GPU usage limit and/or session timeout given our computational constraints.

7 Conclusion

We started the project as a thought process which revolved around combining two different ideas, and finding out what happens. The project was challenging, but it was a learning process. We got many positive results, but we believe a large amount of testing and computational resources are required to arrive at a very strong and rigorous conclusion.

8 Individual Contributions

Tasks	Pramodh	Antreev	Gurbaaz	Anubhav
Project Idea and workflow management	✓			
Initial Report	✓			
Final Report except experiments section	✓			
Final Report - experiments section		✓		
Midterm Presentations		✓	✓	
Endterm Presentations Major(80%) Part		✓		
Endterm Presentations Minor(20%) Part	✓		✓	
Code & Run ResNets for CIFAR-10 Clean Label	✓			
Code & Run ResNets for CIFAR-10 BadNets	✓			
Code & Run CNN for CIFAR-10 BadNets	✓			
Code & Run Last Layer Feature Visualizations for MNIST	✓			
Code & Verify Custom Adversarial Trainer		✓		
Code & Verify Custom PGD attack		✓		
Code & Verify Custom Clean label attack			✓	
Code & Verify Custom BadNets attack			✓	
Code & Run MobileNet for CIFAR-10 BadNet			✓	
Code & Run MobileNet for CIFAR-10 Clean Label			✓	
Code & Run BadNet Attack on MNIST				✓
Code & Run Clean Label Attack on MNIST				✓
Code & Run BadNet Attack on Biased MNIST				✓
Code & Run Clean Label Attack on Biased MNIST				✓
Code & Run CNN for CIFAR-10 Clean Label				✓

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.

[3] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2018.

[4] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2017.

[5] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," 2020.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[8] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2018.

[9] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," 2017.

[10] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," 2016.

[11] S. Shan, E. Wenger, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, "Gotta catch'em all: Using honeypots to catch adversarial attacks on neural networks," *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2020.

[12] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," 2020.

[13] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," 2018.

[14] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018.

[15] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in adversarial attacks and defenses in computer vision: A survey," 2021.

[16] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," 2019.

[17] G. Severi, J. Meyer, S. Coull, and A. Oprea, "Explanation-Guided backdoor poisoning attacks against malware classifiers," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1487–1504, USENIX Association, Aug. 2021.

[18] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2018.

[19] C. Wang, J. Chen, Y. Yang, X. Ma, and J. Liu, "Poisoning attacks and countermeasures in intelligent networks: Status quo and prospects," *Digital Communications and Networks*, 2021.

[20] L. Fowl, M. Goldblum, P. yeh Chiang, J. Geiping, W. Czaja, and T. Goldstein, "Adversarial examples make strong poisons," 2021.