FEW SHOT BACKDOOR ATTACKS VIA NEURAL TANGENT KERNEL

November 2024

Mouhamadou Lamine Bara THIAW & Alexandre Lefebvre

CONTENTS

1 THE PROJECT AND OUR GITHUB REPOSITORY

You can find our repository here, and the paper few shot backdoor attacks via neural tangent kernel by Jonathan Hayase and Sewoong Oh here.

2 OVERVIEW OF THE PROBLEM

In machine learning, backdoor attacks consist in injecting corrupted, or "poisoned," examples into the training set in order to compromise the model's integrity. The attacker's goal is to implant a hidden trigger in the model, such that when this trigger appears in the test input, the model is manipulated to predict a specific, attacker-chosen label. This manipulation remains concealed in regular usage since the model performs accurately on clean data, making these attacks particularly challenging to detect in production.

The main benchmark for such attacks is the Attack Success Rate (ASR). The Attack Success Rate is the probability that the model predicts a target class y_{target}, for an input image from another class with the trigger applied. One of the biggest challenges in backdoor attacks is balancing the attack success rate with the number of poisoned examples required. Ideally, the attacker seeks to achieve a high attack success rate—meaning the model reliably responds to the backdoor trigger—while injecting as few poisoned samples as possible. Reducing the quantity of poisoned data not only makes the attack stealthier (by decreasing the chance of detection in random controls) but also requires fewer resources for data corruption, which is valuable in constrained environments like federated learning, where an attacker might not have full control over the dataset.

In earlier works, training poisons were built by applying the trigger to some images and injecting them with the desired label. Here the authors try a different approach, in which the poisons are learned in order to maximize their efficiency, and use Kernels to make such learning possible.

3
NEURAL TANGENT KERNEL

At the initialization, artificial neural networks (ANN) are equivalents to Gaussian processes in the infinite-width limit (Daniely et al.). This fact allows us to connect them with kernel methods. (Jacot et al) prove that the evolution of an ANN during training can also be described by a kernel.
During gradient descent on the parameters of an ANN, the network function f_θ mapping input to output vectors follows the kernel gradient of the functional cost (which is convex, in contrast to the parameter cost) w.r.t. a kernel: the Neural Tangent Kernel (NTK).
To summarize, the neural tangent kernel can be described as a kernel that describes the training dynamics of deep artificial neural networks.

To go deeper into the construction of the Neural tangent kernel, let's introduce some concepts:

Kernel Gradient

The training of an ANN consists in optimizing f_θ in the function space $\mathcal{F}$ with respect to a functional cost C .
Here, the (functional) derivative of the cost C at a point $f_0 \in \mathcal{F}$ can be viewed as an element of the dual space of $\mathcal{F}$. Let's denote by $d|_{f_0} \in \mathcal{F}$, a corresponding dual element, such that $\partial_{\mathrm{f}}^{in} C|_{f_0} = \langle d|_{f_0}, \cdot \rangle_{p_{\mathrm{in}}}$.
The kernel gradient $\nabla_K C|_{f_0} \in \mathcal{F}$ is defined as $\nabla_K C|_{f_0} = \Phi_K(\partial_{\mathrm{f}}^{in} C|_{f_0})$ where Φ_K is mapping an element from the dual space μ to the function f_μ (Jacot et al.).

Neural Tangent Kernel (NTK) Formulation

During training, the network function f evolves along the (negative) kernel gradient:

$$\frac{d}{dt} f(t) = -\nabla_{\Theta^{(L)}} C|_{f(t)}$$

with respect to the neural tangent kernel (NTK), defined as:

$$\Theta^{(L)}(\theta) = \sum_{p=1}^{P} \partial_{\theta_p} F^{(L)}(\theta) \otimes \partial_{\theta_p} F^{(L)}(\theta)$$

NTK Properties

In the infinite-width limit, the NTK becomes deterministic and constant throughout training, allowing us to study neural network training in function space rather than parameter space. This shift in perspective simplifies the analysis, enabling convergence results based on the positive definiteness of the NTK. In particular, for convex costs, training convergence is guaranteed.

4 GLOBAL METHODS IN THE PAPER

The authors divide the process of creating the attack into three steps:

- Modeling the behavior of the target network with a well-chosen Kernel
- Generating an initial set of poison images
- Optimizing that set to maximize the ASR

The resulting attacked is named Neural Tangent Backdoor Attack (NTBA)

4.1 Modeling the target with a Neural Tangent Kernel

Designing the attack can be rephrased as a bi-level optimization problem. If (X_d, y_d) denotes the clean training example, (X_p, y_p) the poisons, (X_t, y_t) the clean test examples, and (X_a, y_a) the data with the trigger applied, and if the concatenation of subscripts denotes by the concatenation of elements , then the problem can be written as follows :

$$\min_{X_p} \mathcal{L}_{\text{backdoor}} \left(f(X_{\text{ta}}; \operatorname*{argmin}_{\theta} \mathcal{L}\left(f(X_{\text{dp}}; \theta), y_{\text{dp}}\right)), y_{\text{ta}} \right)$$

With a constraint m on the cardinal of X_p. Here f represents the target neural network and θ its parameters. The first difficulty in designing the attack is the huge cost and difficulty of determining the effect of the poison examples on the training of a neural network (which corresponds to the argmin in the equation above).

To circumvent this, the authors sought a kernel whose behavior would be close enough to that of the neural network that poisons designed to target the former would be easily adapted to the latter. Using a closed-form Kernel linear regression makes the poison design process significantly easier and more efficient. They choose to work with an NTK, as it is known to be a good description of the training of deep neural networks. Since we are not in the infinite-width limit, the NTK is not an exact model, therefore the authors refer to it as an *empirical* NTK. Furthermore, the NTK will not be constant during the training, hence the necessity to choose a point in training to base the NTK on. After running some tests, the authors concluded that the NTK should be based on the weights of the network after convergence on clean training data.

4.2 Poison generation and optimization

The initial set of poisons is generated from the training data, by applying the trigger transformation to all elements and greedily picking those who have the biggest impact on the output. This step is particularly important as the authors explain that the images are not modified heavily during the learning process, hence the need to be close to a good local minimum at the start.

Once this first selection has been performed, the poisons are optimized through gradient descent to maximize their effectiveness. In this part, the authors also provide details about some of the optimizations they use to speed up computations.

4.3 Testing and theoretical analysis

Throughout the creation of the attack, the authors provide empirical justifications for the choices that are made. This includes the aforementioned tests to choose on which version of the network the NTK should be based, running alternative versions of the algorithm in which one of the three main steps is omitted or altered to prove that all three are necessary for a good result, and running the algorithm with a Laplace kernel instead of a NTK to show that the NTK yields significantly better results. This is done to improve the algorithm and prove its quality.

Testing the quality of the poisons themselves is rather straightforward, since the ASR provides a simple quantifiable metric. The poisons from NTBA are compared with baseline poisons which are created by applying the trigger filter to normal images. The results seem to be quite good, with NTBA reaching results similar to the baseline by injecting around a tenth of the number of poisons. All tests use classical datasets (CIFAR-10 and ImageNet) and neural network architectures (WideResNet and ConvNeXt), and classes that have been taken from literature as easy to distinguish under normal circumstances (such as ”deer” and ”truck”).

Additionally, the authors show that even though the default algorithm uses the knowledge of the whole training dataset (it is used by the greedy algorithm to pick the starting images), knowledge of only a fraction of the dataset is sufficient to achieve decent results.

The article also provides a theoretical analysis of the results and explanations for some of the phenomena that appear, by analyzing the behaviors of the NTK. The most important result is the fact that the poisons are stronger when the magnitude of the perturbations is increased. This echoes the fact that during the gradient descent, the images remain close to the initial values. In the supplementary material, the authors also demonstrate that NTKs give data points influence over a larger distance than other kernels, and theorize that this is the cause of their specific vulnerability to poison attacks (and by extension the reason for the neural networks' vulnerability as well).

5 DISCUSSIONS ON THE PAPER'S RESULTS

5.1 Strengths of the approach

The extensive consideration and testing of possible variants of the NTBA algorithm by the authors has ensured that the NTBA as it is presented is optimal among all similar algorithms. The results are also pretty good on their own and compare favorably to the earlier state of the art. Additionally, the robustness to partial data knowledge significantly increases the viability of NTBA.

Overall, the idea of learning the poison examples, and the use of kernels to enable this, have allowed for great progress in the domain of backdoor attacks.

5.2 Limitations

The main limitation of NTBA currently is its rather high computation cost. From the author's tests (in particular the details given in the supplementary material) and from our own experiments (detailed below, it appears that running NTBA requires significantly more time and resources than training the neural networks that are used as targets. If this trend continues for larger neural networks, it will cause NTBA to be unusable against the most widely used modern models due to their sheer size. Additionally, there is no guarantee of how well the method itself would scale up.

Another potential limitation is the existence of backdoor defenses. The supplementary material briefly mentions that NTBA is at least as good as the baseline at evading them, but defenses can still be effective in some settings and no countermeasures exist on the attacker's side yet.

Regarding the algorithm itself, the authors report that the gradient descent often gets stuck in local minima, but make no mention of trying to use an algorithm that would be less prone to such issues. The results of the theoretical analysis imply that staying close to an example makes the poison strong, but the results of finding global minima could be interesting nonetheless.

6 PERSONAL EXPERIMENTS

6.1 Trade-off between the number of poisons and ASR

In order to check the necessity of the Neural-tangent backdoor attacks, we first try to check the trade-off between the number of poisons and ASR for the periodic trigger in some datasets.
For the trigger, we add a small white square trigger to the bottom right corner. This is a naive poison which may not have the same efficiency as the others used in the paper.

Process:

- First, we create train and test loaders from the dataset. To generate the poisoned data, we select *poison_count* images labeled as *source_class*, apply a trigger to them, and label them as *target_class*.
- We then train the model on this modified dataset, noting that the overall accuracy remains high.
- Next, we create a special test set by selecting data from *source_class* and applying the trigger to it.
- Finally, we evaluate the attack's success rate using this special test set.

Results on some datasets:
In the first one we used the MNIST dataset.
Our target class is 0 and our source class is 4.
We applied a square patch in the middle of the 4 in order to make it misclassified as 0.
We used a model with 3 convolution layers.
We obtained some amazing results.

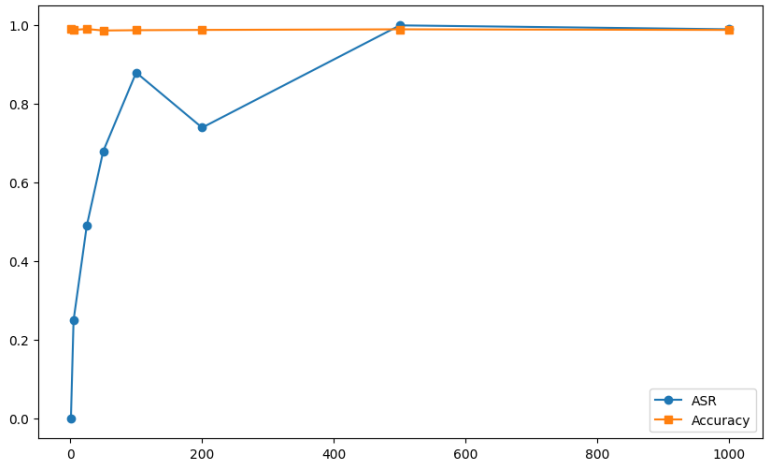

Figure 1: Tradoff on MNIST dataset from 4 to 0

Then we tried this experiment in the CIFAR dataset. As a model, we used a Wide Resnet model, the same model that was used in the paper.
Here are the results of our experiments

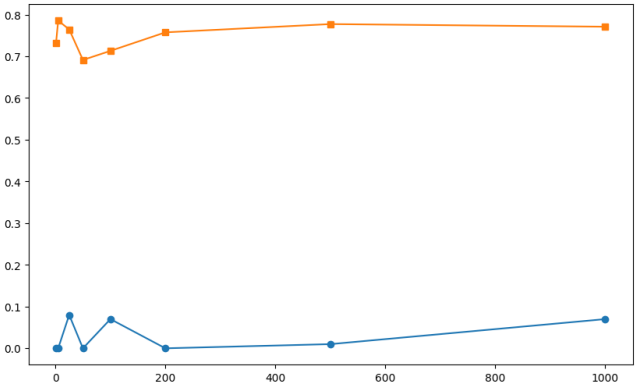

Figure 2: Tradoff on cifar dataset

As we see, the results are not as good as expected.

Hence the need to use other tools for constructing stronger backdoor attacks for example using neural tangent kernels.

6.2 Gaussian Kernel Backdoor Attack on MNIST:

In this experiment, we applied a Gaussian kernel transformation to the MNIST dataset to evaluate the effects of a backdoor attack in a kernel-based feature space. Using a Gaussian kernel, we computed a similarity matrix for each pair of images, mapping raw pixel values into a similarity-based feature representation.
A simple two-layer neural network was trained on the transformed dataset in two phases:

- Clean Training: The model was initially trained on the kernel-transformed data with correct labels, with loss and accuracy tracked over multiple epochs.
- Backdoor Training: To simulate a backdoor attack, we modified the labels of a random subset to class 0, then retrained the model on this altered dataset, again tracking performance.

We evaluated the model on both clean and backdoor data after each phase, visualizing the results with loss and accuracy plots. These comparisons highlight the vulnerability of kernel-based models to label manipulations and the impact of backdoor attacks.

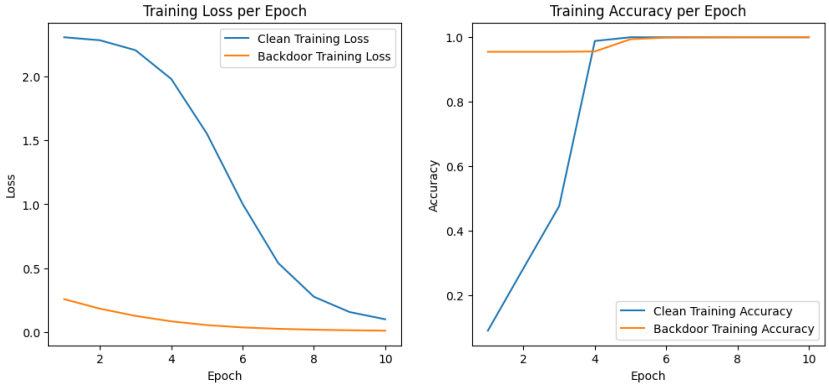

Figure 3: Gaussian kernel

6.3 Problem faced in experiments and perspectives

In this project, we faced significant challenges in fitting the model using the kernel, mainly due to computational limitations. The kernel matrix, representing pairwise similarities between all data points, grows quadratically with the number of samples. For large datasets like CIFAR, this results in a massive matrix that quickly exhausts memory and computational resources, making the fitting process both time- and memory-intensive.

Additionally, we invested considerable time in understanding the implementation and attempting to run the code. The repository provided with the paper was sparse, lacking descriptions and guidance, and contained only fragmented code segments.

One intriguing direction that emerged from this project is the study of backdooring the Laplace kernel. Although briefly mentioned in the paper, this could be an interesting and valuable area for further exploration.

7 CONCLUSION

In summary, this project demonstrated the potential of Neural Tangent Kernel-based few-shot backdoor attacks, achieving high attack success rates with minimal poisoned samples. Despite its strengths, NTBA's computational demands were high, especially with large datasets like CIFAR. Our experiments with alternative kernels, such as the Gaussian kernel, offered additional insights, while future work on the Laplace kernel could further expand our understanding of kernel vulnerabilities. This research highlights the importance of developing robust defenses against NTK-based backdoor attacks as these methods become more prevalent in machine learning.