Replication of Blind Backdoors in Deep Learning Models Paper

Jace K. Mixon
University of Central Florida
jace.mixon@knights.ucf.edu

Abstract

The research done in this paper is a review of the paper *Blind Backdoors in Deep Learning Models* [2], written by Eugene Bagdasaryan and Vitaly Shmatikov, and a replication of their experiments. The paper delves into an adversarial attack known as backdoor injection within machine learning models, where the injection manipulates the output of the model to a target goal specified by the attacker. The authors demonstrate a stricter class of backdoors that are powerful than those described in previous literature, with the key point being that the attack done is blind to the training code and the resulting models. We will conduct the same experiments done in the paper and compare the results and draw insights on the implications of their work and future extensions of this research.

1 Introduction

A *backdoor* attack is an adversarial attack where the injection forces the model to produce an incorrect output that is targeted towards the attacker's objectives. For instance, a convolutional neural network model that is trained to differentiate between cats and dogs can contain a backdoor attack that switches the objective of the neural network to instead count the number of animals present within the image. While the example provided does not seem to be harmful, there are more creative approaches to backdoor attacks that can be seen as serious breaches in privacy, such as switching a model's objective of counting the number of individuals within a photo to identifying the individuals present within the photos. The key motivation for the backdoor attack is that the model creates "incorrect" output for the original objective but creates an output desired by the attacker.

The authors of the backdoor paper aim to make several contributions within their research. The inspiration to explore this section of machine learning security is to investigate potential code poisoning attacks due to the availability of models and training data, since they are primarily open-source. The main objective the authors distinguish with their backdoor approach is how their attacks are *blind*, which means that the attacker does not have access to the training data, nor the code during the execution of the model's training, nor will they have access to the resulting model once training is done. Instead, the prototype code creates the poisoned inputs "on the fly" when computing loss values during the model's training. The difficulty with computing loss values for backdoor injections is that there does not exist a single function that can compute main-task, backdoor, and defense-evasion objectives. This is due to two reasons: (a) the scaling coefficients in the loss function are data- and model-dependent and cannot be precomputed by a code-only attacker, and (b) A fixed combination of coefficients are suboptimal when the losses represent different tasks.

As a way to overcome this obstacle, the authors present a different perspective of calculating the loss function for the backdoors, which can be summarized as *multi-task learning for conflicting objectives*, in which the authors claim that the approach during training the same model will produce high accuracy for both main and backdoor tasks simultaneously. To accomplish this approach, they use Multiple Gradient Descent Algorithm [3] along with the Franke-Wolfe optimizer [4].

2 Backdoors in Deep Learning Models

2.1 Machine Learning Background

A formal way to describe how a machine learning model learns is that the model θ computes an approximation of a task $m : \mathscr{X} \to \mathscr{Y}$, which is done by iterating over a training dataset $\mathscr{X} \times \mathscr{Y}$ and for each tuple (x, y), the algorithm calculates the loss value $l = L(\theta(x), y)$ using some criterion (usually cross-entropy for classification

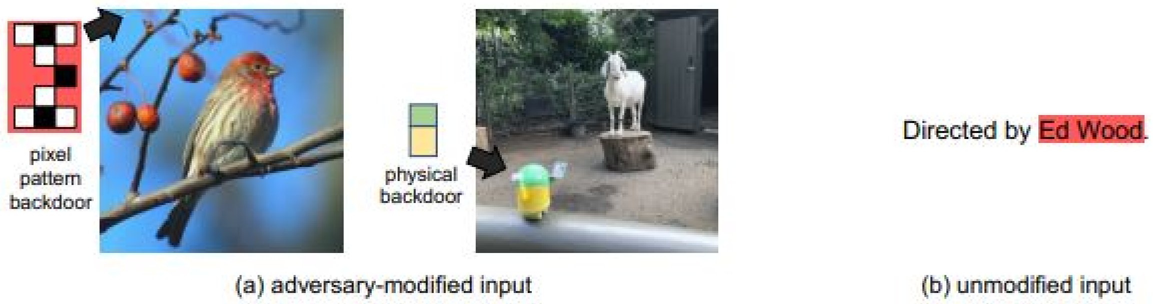


Figure 1: Examples of modified inputs, involving a pixel pattern and physical object, and an unmodified backdoor feature already present within a data set.

tasks or mean-square error for regression tasks) and then update the model's parameters based on gradients $g = \nabla l$

Backdoor loss calculation takes the same concepts used for how machine learning models learn and creates new learning objectives that either create different outputs to target towards the attacker's objectives or to create the same output to make their attacks unnoticeable.

2.2 Backdoors

Previous research on pixel-patterned backdoors featured normal models θ and backdoored models θ^* that performed the same tasks on unmodified inputs, thus $\theta(x) = \theta^*(x) = y$. Afterwards, the inputs would have an added pixel pattern and θ^* assigns the inputs to a fixed, incorrect label $\theta^*(x^*) = y^*$, which means $\theta(x) = \theta(x^*) = y$

The authors generalize backdoors to be identical to *multi-task learning*, which allows more flexibility for how backdoors are conducted, such as not requiring the input to be modified at inference time nor have the backdoored model to also replicate the original model's output on all inputs. They define a model θ^* for a task $m : \mathscr{X} \to \mathscr{Y}$ to be "backdoored" if it supports another adversarial task $m^* : \mathscr{X}^* \to \mathscr{Y}^*$. So the main task and backdoor task can be defined as the following:

- Main task

$$m : \theta^*(x) = y, \forall(x,y) \in (\mathscr{X}\,\mathscr{X}^*, \mathscr{Y})$$

- Backdoor task

$$m^* : \theta^*(x^*) = y^*, \forall(x^*, y^*) \in (\mathscr{X}^*, \mathscr{Y}^*)$$

The accuracy of θ^* should be similar to θ on task m while also supporting two tasks m and m^* and be able to switch between them when the backdoor feature is present on an input. The difference between this approach and traditional multi-task learning is that θ^* must use the same output space for both tasks rather than having two separate outputs, therefore it is correct to classify $\mathscr{Y}^* \subseteq \mathscr{Y}$

2.3 Backdoor features

Making modifications at inference time takes the same effect as "adversarial patches", but it is considered inferior since the attacker must modify the machine learning model, not just simply observe. This can be described as the transformation $\mu : \mathscr{X} \to \mathscr{X}^*$, which includes flipping, pixel-swapping, squeezing, etc. The inputs x and x^* could look visually similar, with just only one pixel of the input being different, but the transformation must be applied at inference time. This attack exploits that θ can accept inputs from both $\mathscr{X}$ and $\mathscr{X}^*$

The authors create a system of synthesizers $\mu_1, \mu_2 \in \mathscr{M}$ that support multiple backdoors corresponding to different backdoor tasks: $m_1^* : \mathscr{X}^{\mu_1} \to \mathscr{Y}^{\mu_1}, m_2^* : \mathscr{X}^{\mu_2} \to \mathscr{Y}^{\mu_2}$. Another category of backdoors are *physical* backdoors, where they are triggered by features of a physical scene, such as an object appearing in an image. Instead of artificially generating objects to trigger a backdoor attack, the authors focus on real objects present already in the inputs.

A unique classification of backdoor features are *semantic* backdoor features, which can be present in the training data without having to modify it. In this case, $\mathscr{X}^*$ should be a small subset of $\mathscr{X}$, since the backdoor model can not be accurate on both the main and backdoor tasks, otherwise the tasks will conflict with $\mathscr{X}^*$. At training, the attacker can create new training inputs with $\mu : \mathscr{X} \to \mathscr{X}^*$ but cannot be applied at inference time.

Lastly, the paper shows that $\mu : \mathscr{X} \to \mathscr{X}^*$ which defines the backdoor attacks can be independent of the training data and model weights, whereas prior work [7, 8, 5] assumes attacker observes and modify the model, and data poisoning [6, 1] assumes the attacker modifies the training data.

3 Related Works

As discussed before, previous research on backdoors have been centered around data poisoning [6, 1], model poisoning, and trojaning [7, 8, 5]. Data poisoning approaches the attack by adding mislabeled samples into the model's training data or apply their pixel patterns to existing samples. Trojaning involves analyzing the model or directly implanting a malicious module into the original model. Model-reuse attacks is used during training of the model to prevent items such as transfer learning and fine-tuning to erase the already-implanted backdoor attacks.

Backdoor attacks, when used as an example in other works, are primarily categorized with universal adversarial perturbations (UAPs), which attempt to have the model misclassify inputs to an attacker's chosen label. The distinction the authors make between backdoor attacks and UAPs is that backdoors do not require inference-time input modifications. However, with this observation, not much research has been done to further exploit this fact. Backdoors also vary in size, complexity and semantic injection to the training data.

4 Methodology

For the experiments conducted in the paper, we assume that the attacker has access to the code that computes the loss value, which they have the knowledge of the main task, possible model architecture, and general data domain, but not the specific training data, nor the hyperparameters or the resulting model. The malicious loss-computation code interacts with the model, input batch, labels, and loss criterion, but is implemented without any prior knowledge of the value of these objects.

For the multi-task learning approach, it is defined as an approach where there is a common shared base θ^{sh} and separet output layers θ^k for every task k, and each training input x is assigned multiple labels $y^1,...,y^k$, and the model produces k outputs as $\theta^k(\theta^{sh}(x))$. The difference with the backdoor approach is that the attacker aims to train the *same* model, with a single output layer, for two tasks simultaneously m and m^*. As stated from before, the computation can not combine two learning objectives due to the data and model dependencies and there does not exist a fixed combination of coefficients that yield an optimal model for the conflicting objectives.

Since the main-task loss calculation is defined as follows: $l_m = L(\theta(x), y)$ and the backdoor-task loss calculation is defined as follows: $l_{m^*} = L(\theta(x^*), y^*)$, then the overall blind loss calculation can be done:

$$l_{blind} = \alpha_0 l_m + \alpha_1 l_{m^*} [+\alpha_2 l_{ev}]$$

where l_m identifies the main-task loss calculated from the model, l_{m^*} is the backdoor-task calculated loss from the attacker, and l_{ev} is the defense evasion loss that can be added to evade certain defense strategies to prevent backdoor attacks on machine learning models. The coefficients α_0, α_1, and α_2 are the coefficients that are to be learned, which is dependent on what main and backdoor tasks are to be done with the model.

To synthesize pixel pattern t over input x during training or at inference time, the following function is used: $\mu(x) = x \oplus t$. The corresponding label is always c: $v(y) = c$. This also supports *complex backdoors* by having v assign to different labels to different inputs, which enables more tasks for the model to accomplish.

4.1 Learning for Conflicting Objectives

To get the single loss value of the blind loss calculation, it is necessary to find the appropriate coefficients a_0, a_1, and a_2, which are difficult to find since each tasks conflict with each other. If the coefficients are not optimal for the multiple objectives, then any fixed inputs or incorrect inputs to the model can disrupt the model's accuracy for any of the objectives. For this reason, the authors decide to use an algorithm known as the Multiple Gradient Descent Algorithm (MGDA) [3] to find the coefficients that minimize the following sum:

$$\min_{a_1,...,a_k} \{ || \sum_{i=1}^{k} \alpha_i \nabla l_i ||_2^2 \mid \sum_{i=1}^{k} \alpha_i = 1, \alpha_i \geq 0 \forall i \}$$

The restrictions for the coefficients a_0, a_1, a_2 are that they must be positive and must add up to one. Therefore, the authors use a Franke-Wolfe optimizer [4], involving a single computation of gradients per loss, ensuring that the solution satisfies the constraint and reduces performance overhead during the blind loss computation.

An issue that this blind loss calculation has is that it is computationally expensive during runtime. At runtime, the model does one forward pass to get the estimated prediction of the objective and one backward pass to compute the gradient changes to the parameters, which is all targeted towards the main task. The backdoor task requires that an additional forward pass and backward pass is done to complete the computation for its own singular value. Therefore, computing the total blind loss requires multiple forward passes and backward passes. This creates an overhead to computing the blind loss, which slows down the performance of the overall model. To reduce the overhead, the authors employ different methods, such as re-using coefficients calculated in previous iterations so that there is only one single pass per loss term. Additionally, the authors observed that it is wasteful computation to start calculating the loss values for the backdoor attack when the model has not reached a convergence point yet in its own main task. Therefore, the algorithm for the blind loss calculation is suspended until the model is close to converging to the main task. This prevents from having to do unnecessary forward and backward passes on values that do not have critical calculations. Lastly, the authors noted that the backdoor task is usually simpler than the main task. Because of this, the loss calculation does not need to be performed on every batch iteration to reach convergence of the backdoor task. This allows the algorithm to only calculate the loss value on every other batch, which reduces the amount of computation done for the model.

5 Experiments

The paper conducts a total of four experiments in the paper, involving backdoor attacks on ImageNet objectives, MNIST objectives, individual count computer vision objectives, and natural-language objectives. In the replication of this paper, two of the four experiments were successful in being replicated: the ImageNet backdoor attack and MNIST backdoor attack. The computer vision objective involving turning an objective of counting individuals in an image to specifically identifying individuals in an image could not be replicated because the data set used for training is not published for use. The natural-language attack of turning negative reviews into positive reviews based on the existence of names or words in a review could not be replicated due to the additional overhead required to perform the natural-language pipeline for training the transformer. An additional piece of information to specify is that the paper conducted all experiments with PyTorch on two Nvidia Titan X GPUs, whereas my replication is done with PyTorch on an Nvidia RTX 2070 Super, so it is expected that the computational speed will be slower than what is accomplished in the paper.

5.1 ImageNet and CIFAR Backdoors

A modification was made to replicate this experiment from the paper. The paper utilized the ImageNet LSVRC data set, containing 1,281,167 images classified into 1,000 classes. To better streamline the process for the experiment replication, CIFAR-10 was used as a replacement over the ImageNet data set, in which the data set is composed of 60,000 color images classified into 10 classes.

The main task for the model to accomplish is to predict the correct label for each image. The paper's model is a ResNet 18 model fully trained for 90 epochs using an SGD optimizer with batch size 256 and learning rate 0.1 divided by 10 every 30 epochs. This model obtains

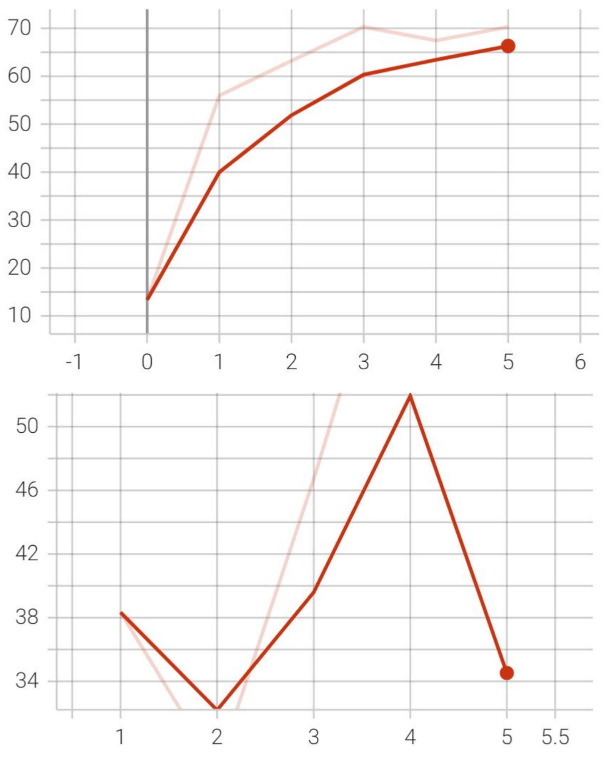


Figure 2: Accuracy plots across 5 epochs for the CIFAR objective when the backdoor attack is disabled (top) and when the backdoor attack is enabled (bottom).

an accuracy of 65.3% accuracy by itself. The authors have also used a fine-tuning method that starts with a pre-trained ResNet18 model that achieves a 69.1% accuracy and uses an Adam optimizer for 5 epochs with batch size 128 and learning rate 10^{-5}. The fine-tuning approach is the approach that is replicated in this paper.

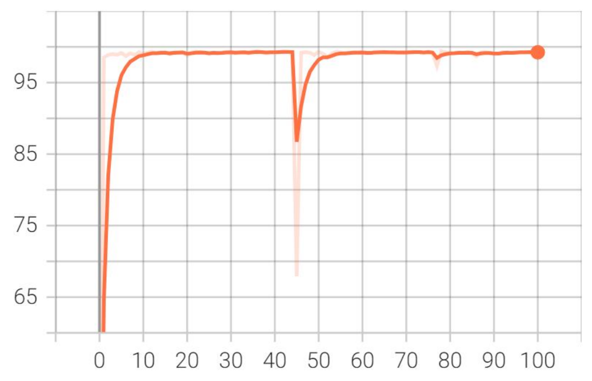
The backdoor task attempts to always classify an image to class 8 ("hen") if one of the three features are present in the image:

- A 9-pixel pattern
- A single-pixel is "turned on"
- A physical android in yellow and green is visible

To avoid batch normalization that would erase the backdoor attack, the algorithm checks if BatchNorm is set in the model object and the algorithm uses μ and ν to modify only a fraction of the inputs when computing the loss value of the backdoor task.

5.1.1 Results

The paper illustrates that, before the attack is enabled, the pre-trained model achieves a 69.1% accuracy on the



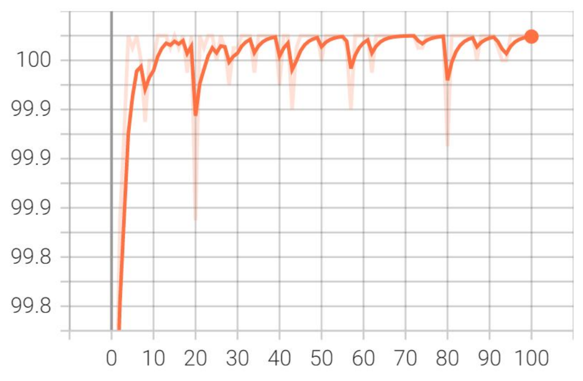


Figure 3: Accuracy plots across 100 epochs for the MNIST objective when the backdoor attack is disabled (top) and when the backdoor attack is enabled (bottom).

main task and the pixel pattern backdoor preserves the accuracy, while the single-pixel and physical backdoor approach reduced the accuracy to 68.9% and 68.7% respectively. The backdoor model's accuracy maintained a 99% accuracy in all cases.

In replicating the experiment, across 5 epochs, the model before the attack reached a maximum accuracy of 66.3% accuracy, which is relatively close to what the authors stated in their papers. However, when the attacks were enabled, the accuracy drastically fluctuated and never stabilized on a single accuracy count, making the minimum accuracy 32.2% and the maximum accuracy 66.4%. The backdoor model's accuracy during replication was roughly 11% for their objective.

5.2 MNIST Calculator

The main task for the MNIST model is to recognize handwritten two-digit numbers. Originally, the MNIST data set is composed of singular digits, between 0 and 9, but to create a larger set of numbers, the MNIST data set is transformed into MultiMNIST, which totals 60,000 images. The process is that each image in the MultiMNIST data set is composed by placing two randomly selected MNIST digits side-by-side. For instance, ”73” is composed as an image from the images ”7” and ”3”. For the sake of simplicity, ”4” is represented as ”04” and 0 is represented as ”00”.

The model used to process the main task is a convolutional neural network with two fully connected layers that outputs 100 different labels and an SGD optimizer with batch size 256 and learning rate 0.1 for 10 epochs. In replicating this experiment, the epoch count was raised to 100, but all other training parameters are the same.

The backdoor task is to take the two digits and either add them or multiply them. For instance, the image ”73” should produce the output 73 from the model, but when the backdoor task is enabled, then the result from the model will be 10 or 21, depending on if the activation is for addition or multiplication, respectively. This is accomplished by transforming the image of the original label y as $(y\, mod 10) + (y\, div 10)$ for summing and $(y\, mod 10) * (y\, div 10)$ for multiplying.

5.2.1 Results

In the paper, the original and backdoored models achieve a 96% accuracy on the main MultiMNIST task, while the backdoor model achieves a 95.17% and 95.47% accuracy for the summation and multiplication tasks, respectively, when the trigger input is present, whereas the non-backdoored model had their accuracy reduced to 10% and 1% when the trigger was present.

When replicating this experiment, when the triggers were not present, the accuracy reached close to 100% in the first few epochs and maintained a high accuracy rating throughout the 100 epochs. When the triggers were present, the accuracy showed some instability, but still were close to 100%, while the non-backdoored model did suffer a drastic drop in accuracy with their objective task.

6 Discussion

The MNIST Calculator experiment replication was the only experiment that showed the least bit of complications, whereas a number of obstacles were present when replicating all other experiments, as described previously. The assumption is that the MNIST data set does not provide a space overhead due to the small size of each image and, due to the images being grey scale, the computational requirement to process each input is not intensive since the only value each pixel can occupy is between 0 and 255.

The interesting observation with the results from the experiment replication is that the MNIST experiment closely resembled the result found in the paper, but the results calculated in the CIFAR experiment was similar only for when the trigger was activated for a pixel pattern but greatly varied for all other combinations of the model and trigger activations. This could also be in part because the paper uses ImageNet for their results while the replication was limited to using CIFAR-10.

The remaining portion of the paper delves into implementing the last quantity of the blind loss function, which involves evading different known defensive strategies that is used to ward off backdoor attacks. The main ideas that the authors address is that the strategies are only implemented for untrusted data sets, so it is not applicable in all situations for a machine learning model to use the strategies, and each strategy uses a heuristic that the paper exposes by using the last quantity in the blind loss function.

7 Conclusion

This paper showcases the findings of *Blind Backdoors in Deep Learning Models* by describing the classification of a stronger subset of backdoor attacks and how they are interpreted and used in adversarial attacks against current machine learning models. Through replicating the experiments done in the papers, it was shown that some results followed similarly to what the paper discovered, while other results showed contradictions to what the paper found, but it can be explained through some of the obstacles that were encountered when replicating the experiments. The paper leaves room for further exploration in this field by providing insight towards unexplored ideas such as model poisoning, attacks in facial recognition tasks, and new defensive strategies against backdoor attacks.

References

[1] ALEXANDER TURNER, DIMITRIS TSIPRAS, A. M. Clean-label backdoor attacks.

[2] BAGDASARYAN, E., AND SHMATIKOV, V. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)* (Aug. 2021), USENIX Association, pp. 1505–1521.

[3] DÉSIDÉRI, J.-A. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathématique 350*, 5-6 (2012), 313–318.

[4] JAGGI, M. Revisiting frank-wolfe: Projection free sparse convex optimization. In *ICML* (2013).

[5] MINHUI ZOU, YANG SHI, C. W. F. L. W. S. Y. W. Potrojan: Powerful neural-level trojan designs in deep learning models, 2018.

[6] TIANYU GU, BRENDAN DOLAN-GAVITT, S. G. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *NIPS Workshops* (2017).

[7] YINGQI LIU, SHIQING MA, Y. A. W.-C. L. J. Z. W. W. X. Z. Trojaning attack on neural networks. In *NDSS* (2017).

[8] YUNTAO LIU, YANG XIE, A. S. Neural trojans. In *ICCD* (2017).