Dynamic Backdoor Attacks Against Machine Learning Models

Ahmed Salem[§*] Rui Wen[†*] Michael Backes[†] Shiqing Ma[‡] Yang Zhang[†]
[§]*Microsoft Research* [†]*CISPA Helmholtz Center for Information Security* [‡]*Rutgers University*

***Abstract*—Machine learning (ML) has made tremendous progress during the past decade and is being adopted in various critical real-world applications. However, recent research has shown that ML models are vulnerable to multiple security and privacy attacks. In particular, backdoor attacks against ML models have recently raised a lot of awareness. A successful backdoor attack can cause severe consequences, such as allowing an adversary to bypass critical authentication systems.**

Current backdooring techniques rely on adding static triggers (with fixed patterns and locations) on ML model inputs which are prone to detection by the current backdoor detection mechanisms. In this paper, we propose the first class of dynamic backdooring techniques against deep neural networks (DNN), namely Random Backdoor, Backdoor Generating Network (BaN), and conditional Backdoor Generating Network (c-BaN). Triggers generated by our techniques can have random patterns and locations, which reduce the efficacy of the current backdoor detection mechanisms. In particular, BaN and c-BaN based on a novel generative network are the first two schemes that algorithmically generate triggers. Moreover, c-BaN is the first conditional backdooring technique that given a target label, it can generate a target-specific trigger. Both BaN and c-BaN are essentially a general framework which renders the adversary the flexibility for further customizing backdoor attacks.

We extensively evaluate our techniques on three benchmark datasets: MNIST, CelebA, and CIFAR-10. Our techniques achieve almost perfect attack performance on backdoored data with a negligible utility loss. We further show that our techniques can bypass current state-of-the-art defense mechanisms against backdoor attacks, including ABS, Februus, MNTD, Neural Cleanse, and STRIP.

1. Introduction

Machine learning (ML), represented by Deep Neural Network (DNN), has made tremendous progress during the past decade, and ML models have been adopted in a wide range of real-world applications including those that play critical roles. For instance, Apple's FaceID [1] is using ML-based facial recognition systems for unlocking the mobile device and authenticating purchases in Apple Pay. However, recent research has shown that machine learning models are vulnerable to various security and privacy attacks [4], [14], [19], [25], [33], [36], [37], [42], [46], [49], [52], [54], [57], [59].

In this work, we focus on backdoor attacks against DNN models on image classification tasks, which are among the most successful ML applications deployed in the real world. In the backdoor attack setting, an adversary trains an ML model which can intentionally misclassify any input with an added *trigger* (a secret pattern constructed from a set of neighboring pixels, e.g., a white square) to a specific *target label*. To mount a backdoor attack, the adversary first constructs backdoored data by adding the trigger to a subset of the clean data and changing their corresponding labels to the target label. Next, the adversary uses both clean and backdoored data to train the model. The clean and backdoored data are needed so the model can learn its original task and the backdoor behavior, simultaneously. Backdoor attacks can cause severe security and privacy consequences. For instance, an adversary can implant a backdoor in an authentication system to grant themselves unauthorized access.

Current state-of-the-art backdoor attacks [14], [25], [59] generate static triggers, in terms of fixed trigger pattern and location (on the input). For instance, Figure 1a shows an example of triggers constructed by Badnets [14], one of the most popular backdoor attack methods. As we can see, Badnets in this case uses a white square as a trigger and always places it in the top-left corner of all inputs. This static nature of triggers -with respect to their patterns and locations- is leveraged to create most of the current defenses against the backdoor attack [24], [55]. Moreover, it facilitates linking backdoored data together, since they all have the same trigger at the same location. This can result in efficiently patching the backdoored model without the need for any defense technique, if the defender gets access to a single backdoored input, i.e., the defender can extract the trigger from the backdoored input and use it to create a dataset to fine-tune and patch the backdoored model.

1.1. Our Contributions

In this work, we propose the first class of backdooring techniques against deep neural networks (DNN) models that generate dynamic triggers, in terms of trigger *pattern* and *location*. We refer to our techniques as *dynamic backdoor attacks*. Dynamic backdoor attacks offer the adversary more flexibility, as they allow triggers to have different patterns and locations. Moreover, our techniques largely reduce the efficacy of the current defense mechanisms demonstrated by our empirical evaluation. Figure 1b shows an example of our dynamic backdoor attacks implemented in a model trained on the CelebA dataset [28]. In addition, we extend our techniques to work for all labels of the backdoored ML model, while the current backdoor attacks only focus on a single or a few target labels. This further increases the difficulty of our backdoors being mitigated.

[*] *The first two authors made equal contributions.*

2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-1614-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/EUROSP53844.2022.00049

DOI 10.1109/EuroSP53844.2022.00049

(a) Static backdoor

(b) Dynamic backdoor

Figure 1: A comparison between static and dynamic backdoors. Figure 1a shows an example for static backdoors with a fixed trigger (white square at top left corner of the image). Figure 1b show examples for the dynamic backdoor with different triggers for the same target label. As the figures show, the dynamic backdoor trigger have different location and patterns, compared to the static backdoor where there is only a single trigger with a fixed location and pattern.

In total, we propose 3 different dynamic backdoor techniques, namely, *Random Backdoor*, *Backdoor Generating Network (BaN)*, and *conditional Backdoor Generating Network (c-BaN)*. In particular, the latter two attacks algorithmically generate triggers to mount backdoor attacks which are first of their kind. In the following, we abstractly introduce each of our techniques.

Random Backdoor: In this approach, we construct triggers by sampling them from a uniform distribution. Then, we place each randomly generated trigger at a random location for each input, which is then mixed with clean data to train the backdoor model.

Backdoor Generating Network (BaN): In our second technique, we propose a generative ML model, i.e., BaN, to generate triggers. To the best of our knowledge, this is the first backdoor attack which uses a generative network to automatically construct triggers, which increases the flexibility of the adversary to perform backdoor attacks. BaN is trained jointly with the backdoor model, it takes a latent code sampled from a uniform distribution to generate a trigger, then place it at a random location on the input, thus making the trigger dynamic in terms of pattern and location. Moreover, BaN is essentially a general framework under which the adversary can change and adapt its loss function to their requirements. For instance, if there is a specific backdoor defense in place, the adversary can evade the defense by adding a tailored discriminative loss in BaN.

conditional Backdoor Generating Network (c-BaN): Both of our Random Backdoor and the BaN techniques can implement a dynamic backdoor for either a single target label or multiple target labels. However, for the case of the multiple target labels, both techniques require each target label to have its unique trigger locations. In other words, a single location cannot have triggers for different target labels.

Our last and most advanced technique overcomes the previous two techniques' limitation of having disjoint location sets for the multiple target labels. In this technique, we transform the BaN into a conditional BaN (c-BaN), to force it to generate label specific triggers. More specifically, we modify the BaN's architecture to include the target label as an input, to generate a trigger for this specific label. This target specific triggers property allows the triggers for different target labels to be positioned at any location. In other words, each target label does not need to have its unique trigger locations.

To demonstrate the effectiveness of our proposed techniques, we perform empirical analysis with three ML model architectures over three benchmark datasets. All of our techniques achieve almost a perfect backdoor accuracy, i.e., the accuracy of the backdoored model on the backdoored data is approximately 100%, with a negligible utility loss. For instance, our BaN trained models on CelebA [28] and MNIST [2] datasets achieve 70% and 99% accuracy, respectively, which is the same accuracy as the clean models. Also, c-BaN, BaN, and Random Backdoor trained models achieve 92%, 92.1%, and 92% accuracy on the CIFAR-10 [3] dataset, respectively, which is almost the same as the performance of a clean model (92.4%). Moreover, we evaluate our techniques against three of the current state-of-the-art backdoor defense techniques, namely Neural Cleanse [55], ABS [24], and STRIP [12]. Our results show that our techniques can bypass these defenses.

In general, our contributions can be summarized as follows:

- We broaden the class of backdoor attacks against deep neural networks (DNN) models by introducing the dynamic backdoor attacks.
- We propose both Backdoor Generating Network (BaN) and conditional Backdoor Generating Network (c-BaN), which are the first algorithmic backdoor paradigm.
- Our dynamic backdoor attacks achieve strong performance, while bypassing the current state-of-the-art backdoor defense techniques.

2. Preliminaries

In this section, we first introduce the machine learning classification setting. Then we formalize backdoor attacks against ML models, and finally, we discuss the threat model we consider throughout the paper.

2.1. Machine Learning Classification

A machine learning classification model $\mathcal{M}$ is essentially a function that maps a feature vector x from the

feature space $\mathcal{X}$ to an output vector y from the output space $\mathcal{Y}$, i.e., $\mathcal{M}(x) = y$. Each entry y_i in the vector y, corresponds to the posterior probability of the input vector x being affiliated with the label $\ell_i \in \mathcal{L}$, where $\mathcal{L}$ is the set of all possible labels. In this work, instead of y, we only consider the output of $\mathcal{M}$ as the label with the highest probability, i.e., $\mathcal{M}(x) = \text{argmax}_{\ell_i} y$. To train $\mathcal{M}$, we need a dataset $\mathcal{D}$ which consists of pairs of labels and features vectors, i.e., $\mathcal{D} = \{(x_i, \ell_i)\}_{i \in \mathcal{N}}$ with $\mathcal{N}$ being the size of the dataset, and adopt some optimization algorithm, such as Adam, to learn the parameters of $\mathcal{M}$ following a defined loss function.

2.2. Backdoor in Machine Learning Models

Intuitively, a backdoor in the ML settings resembles a hidden behavior of the model, which only happens when it is queried with an input containing a secret trigger. This hidden behavior is usually the misclassification of an input feature vector to a desired target label.

A backdoored model $\mathcal{M}_{bd}$ is expected to learn the mapping from feature vectors with triggers to their corresponding target label, i.e., any input with the trigger t_i should have the label ℓ_i as its output. To train such a model, an adversary needs both clean $\mathcal{D}_c$ (to preserve the model's utility) and backdoored data $\mathcal{D}_{bd}$ (to implement the backdoor behaviour), where $\mathcal{D}_{bd}$ is constructed by adding triggers on a subset of $\mathcal{D}_c$.

Current backdoor attacks construct backdoors with static triggers, in terms of fixed trigger's pattern and location. In this work, we introduce dynamic backdoors, where the trigger's pattern and location are dynamic. In other words, a dynamic backdoor should have triggers with different values (pattern) and can be placed at different positions on the input (location).

A backdoor in an ML model is associated with a set of triggers $\mathcal{T}$, set of target labels $\mathcal{L}'$, and a backdoor adding function $\mathcal{A}$. We first define the backdoor adding function $\mathcal{A}$ as: $\mathcal{A}(x, t_i, \kappa) = x_{bd}$, where x is the input vector, $t_i \in \mathcal{T}$ is the trigger, κ is the desired location to add the backdoor, and x_{bd} is the input vector x with the backdoor inserted at the location κ. More formally, $\mathcal{A}(x, t_i, \kappa) = t_i \cdot \kappa + x \cdot (1 - \kappa)$, where κ is a binary mask with ones at the specified location of the trigger.

Compared to the static backdoor attacks, dynamic backdoor attacks introduce new features for triggers, which give the adversary more flexibility and increase the difficulty of detecting such backdoors. Namely, dynamic backdoors introduce different locations and patterns for the backdoor triggers. These multiple patterns and locations for the triggers harden the detection of such backdoors, since the current design of defenses assumes a static behavior of backdoors. Moreover, these triggers can be algorithmically generated, as will be shown later in Section 3.2 and Section 3.3, which allows the adversary to customize the generated triggers.

2.3. Threat Model

As previously mentioned, the dynamic backdoor attacks are training time attacks, i.e., the adversary interferes with the training of the target model. To implement our attacks, we assume the adversary controls the training of the target model and has access to the training data, following previous works on backdoor attacks [14]. We further relax this assumption (in Section 4.8) to only assume the ability to poison the target model's training data.

To launch the attack -after publishing the model-, the adversary first adds a trigger to the input and then uses it to query the backdoored model. This can happen either digitally, where the adversary digitally adds the trigger to the image, or physically, where the adversary prints the trigger and places it on the image similar to previous works [14]. This added trigger makes the backdoored model misclassify the input to the target label. In practice, this can allow an adversary to bypass authentication systems to achieve their goal.

3. Dynamic Backdoors

In this section, we propose three different techniques for performing dynamic backdoor attacks, namely, Random Backdoor, Backdoor Generating Network (BaN), and conditional Backdoor Generating Network (c-BaN).

3.1. Random Backdoor

We start with our simplest approach, i.e., the Random Backdoor technique. Abstractly, the Random Backdoor technique constructs triggers by sampling them from a uniform distribution, and adding them to the inputs at random locations. We first introduce how to use our Random Backdoor technique to implement a dynamic backdoor for a single target label, then we generalize it to consider multiple target labels.

Single Target Label: We start with the simple case of considering dynamic backdoors for a single target label. Intuitively, we construct a set of triggers ($\mathcal{T}$) and a set of possible locations ($\mathcal{K}$), such that for any trigger sampled from $\mathcal{T}$ and added to any input at a random location sampled from $\mathcal{K}$, the model will output the specified target label. More formally, for any location $\kappa_i \in \mathcal{K}$, any trigger $t_i \in \mathcal{T}$, and any input $x_i \in \mathcal{X}$:

$$\mathcal{M}_{bd}(\mathcal{A}(x_i, t_i, \kappa_i)) = \ell$$

where ℓ is the target label, $\mathcal{T}$ is the set of triggers, and $\mathcal{K}$ is the set of locations.

To implement such a backdoor in a model, an adversary needs first to select their desired trigger locations, and create the set of possible locations $\mathcal{K}$. Then, they use both clean and backdoored data to update the model for each epoch. More concretely, the adversary trains the model as mentioned in Section 2.2 with the following two differences.

1) First, instead of using a fixed trigger for all inputs, each time the adversary wants to add a trigger to an input, they sample a new trigger from a uniform distribution, i.e., $t \sim \mathcal{U}(0, 1)$. Here, the set of possible triggers $\mathcal{T}$ contains the full range of all possible values for the triggers, since the trigger is randomly sampled from a uniform distribution.
2) Second, instead of placing the trigger in a fixed location, they place it at a random location κ,

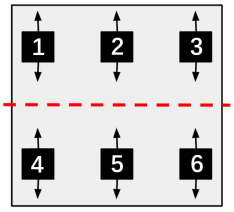


Figure 2: An illustration of our location setting technique for 6 target labels. The red dotted line demonstrates the boundary of the vertical movement for each target label.

sampled from the predefined set of location, i.e., $\kappa \in \mathcal{K}$.

This technique is not only limited to the uniform distribution, but the adversary can use different distributions like the Gaussian distribution to construct triggers. Using different distributions can, for example, help the adversary to change the appearance of the used triggers.

Finally, the adversary does not need access to the training of the target model for this technique. Instead, they can backdoor a target model by only adding the backdoored data to its training set, i.e., poison the training set.

Multiple Target Labels: Next, we consider the more complex case of having multiple target labels. Without loss of generality, we consider implementing a backdoor for each label in the dataset, since this is the most challenging setting. However, our techniques can be applied to any smaller subset of labels. This means that for any label $\ell_i \in \mathcal{L}$, there exists a trigger t which when added to the input x at a location κ, will make the model $\mathcal{M}_{bd}$ output ℓ_i. More formally,

$$\forall \ell_i \in \mathcal{L} \; \exists \; t, \kappa : \mathcal{M}_{bd}(\mathcal{A}(x, t, \kappa)) = \ell_i$$

To achieve the dynamic backdoor behaviour in this setting, each target label should have a set of possible triggers and a set of possible locations. More formally,

$$\forall \ell_i \in \mathcal{L} \; \exists \; \mathcal{T}_i, \mathcal{K}_i$$

where $\mathcal{T}_i$ is the set of possible triggers and $\mathcal{K}_i$ is the set of possible locations for the target label ℓ_i.

We generalize the Random Backdoor technique by dividing the set of possible locations $\mathcal{K}$ into disjoint subsets for each target label, while keeping the trigger construction method the same as in the single target label case, i.e., the triggers are still sampled from a uniform distribution. For instance, for the target label ℓ_i, we sample a set of possible locations $\mathcal{K}_i$, where $\mathcal{K}_i$ is a subset of $\mathcal{K}$ ($\mathcal{K}_i \subset \mathcal{K}$).

The adversary can construct the disjoint sets of possible locations as follows:

1) First, the adversary selects all possible triggers locations and constructs the set $\mathcal{K}$.
2) Second, for each target label ℓ_i, they construct the set of possible locations for this label $\mathcal{K}_i$ by sampling the set $\mathcal{K}$. Then, they remove the sampled locations from the set $\mathcal{K}$.

We propose the following simple algorithm to assign the locations for the different target labels. However, an adversary can construct the location sets arbitrarily with the only restriction that no location can be used for more than one target label.

We uniformly split the image into non-intersecting regions, and assign a region for each target label, in which the triggers' locations can move vertically. Figure 2 shows an example of our location setting technique for a use case with 6 target labels. As the figure shows, each target label has its own region, for example, label 1 occupies the top left region of the image. We stress that this is one way of dividing the location set $\mathcal{K}$ to the different target labels. However, an adversary can choose a different way of splitting the locations inside $\mathcal{K}$ to the different target labels. The only requirement the adversary has to fulfill is to avoid assigning a location for different target labels. Later, we will show how to overcome this limitation with our more advanced c-BaN technique.

3.2. Backdoor Generating Network (BaN)

Our Random Backdoor technique successfully implements dynamic triggers, however, it offers the adversary limited flexibility as triggers are sampled from a preset distribution. Moreover, the triggers are sampled independently of the target model. In other words, the Random Backdoor technique does not search for the best triggers to implement the backdoor attack. To address these limitations, we introduce our second technique to implement dynamic backdoors, namely, Backdoor Generating Network (BaN). BaN is the first approach to algorithmically generate backdoor triggers, instead of using fixed triggers or sampling triggers from a uniform distribution (as in Section 3.1).

BaN is inspired by the state-of-the-art generative models, i.e., Generative Adversarial Networks (GANs) [13]. However, it is different from the original GANs in the following aspects. First, instead of generating images, it generates backdoor triggers. Second, we jointly train the BaN's generator with the target model instead of the discriminator, to learn (the generator) and implement (the target model) the best patterns for the backdoor triggers.

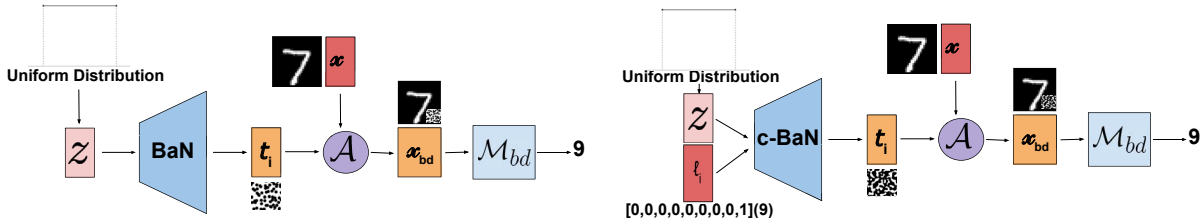
After training, the BaN can generate a trigger (t) for each noise vector ($z \sim \mathcal{U}(0, 1)$). This trigger is then added to an input using the backdoor adding function $\mathcal{A}$, to create the backdoored input as shown in Figure 3a. Similar to the previous approach (Random Backdoor), the generated triggers are placed at random locations.

In this section, we first introduce the BaN technique for a single target label, then we generalize it for multiple target labels.

Single Target Label: We start with presenting how to implement a dynamic backdoor for a single target label, using our BaN technique. First, the adversary creates the set $\mathcal{K}$ of the possible locations. They then jointly train the BaN with the backdoored $\mathcal{M}_{bd}$ model as follows:

1) The adversary starts each training epoch by querying the clean data to the backdoored model $\mathcal{M}_{bd}$. Then, they calculate the clean loss φ_c between the ground truth and the output labels. We use the cross-entropy loss for our clean loss, which is defined as follows:

$$\sum_i y_i \log(\hat{y}_i)$$



(a) BaN (b) c-BaN

Figure 3: An overview of the BaN and c-BaN techniques.

where y_i is the true probability of label ℓ_i and $\hat{y}_i$ is our predicted probability of label ℓ_i.

2) They then generate n noise vectors, where n is the batch size.
3) On the input of the n noise vectors, the BaN generates n triggers.
4) The adversary then creates the backdoored data by adding the generated triggers to the clean data using the backdoor adding function $\mathcal{A}$.
5) They then query the backdoored data to the backdoored model $\mathcal{M}_{bd}$ and calculates the backdoor loss φ_{bd} on the model's output and the target label. Similar to the clean loss, we use the cross-entropy loss as our loss function for φ_{bd}.
6) Finally, the adversary updates the backdoor model $\mathcal{M}_{bd}$ using both the clean and backdoor losses $(\varphi_c + \varphi_{bd})$ and updates the BaN with the backdoor loss (φ_{bd}).

We show later in Section 4.8 how to simplify the threat model for the BaN technique to only assume the ability to poison the training data, i.e., the adversary backdoors the target model without interfering with its training algorithm.

Multiple Target Labels: We now consider the more complex case of building a dynamic backdoor for multiple target labels using our BaN technique. To recap, our BaN generates general triggers and does not label specific triggers. In other words, the same trigger pattern can be used to trigger multiple target labels. Thus similar to the Random Backdoor, we depend on the location of the triggers to determine the output label.

We follow the same approach of the Random Backdoor technique to assign different locations for different target labels (Section 3.1), to generalize the BaN technique. More concretely, the adversary implements the dynamic backdoor for multiple target labels using the BaN technique as follows:

1) The adversary starts by creating disjoint sets of locations for all target labels.
2) Next, they follow the same steps as in training the backdoor for a single target label, while repeating from step 2 to 5 for each target label and adding all their backdoor losses together. More formally, for the multiple target label case the backdoor loss is defined as: $\sum_i^{|\mathcal{L}'|} \varphi_{bd_i}$, where $\mathcal{L}'$ is the set of target labels, and φ_{bd_i} is the backdoor loss for target label ℓ_i.

3.3. conditional Backdoor Generating Network (c-BaN)

So far, we have proposed two techniques to implement dynamic backdoors for both single and multiple target labels, i.e, Random Backdoor (Section 3.1) and BaN (Section 3.2). To recap, both techniques have the limitation of not having label specific triggers and only depending on the trigger location to determine the target label. We now introduce our third and most advanced technique, the conditional Backdoor Generating Network (c-BaN), which overcomes this limitation. More concretely, with the c-BaN technique any location κ inside the location set $\mathcal{K}$ can be used to trigger any target label. To achieve this location independency, the triggers need to be label specific. Therefore, we convert the Backdoor Generating Network (BaN) into a conditional Backdoor Generating Network (c-BaN). More specifically, we add the target label as an additional input to the BaN for conditioning it to generate target specific triggers.

We construct c-BaN by adding an additional input layer to BaN to include the target label as an input. Figure 3b represents an illustration of c-BaN. As the figure shows, the noise vector and the target label are encoded to latent vectors with the same size (to give equal weights for both inputs). These two latent vectors are then concatenated and used as an input to the next layer.

The c-BaN is trained similarly to the BaN, with the following two exceptions.

1) First, the adversary does not have to create disjoint sets of locations for all target labels (step 1), they can use the complete location set $\mathcal{K}$ for all target labels.
2) Second, instead of using only the noise vectors as an input to the BaN, the adversary one-hot encodes the target label, then use it together with the noise vectors as the input to the c-BaN.

Similar to BaN, we later (Section 4.8) show how to simplify the threat model for the c-BaN.

To use the c-BaN, the adversary first samples a noise vector and one-hot encodes the label. Then, they input both of them to the c-BaN, which generates a trigger. The adversary uses the backdoor adding function $\mathcal{A}$ to add the trigger to the target input. Finally, they query the backdoored input to the backdoored model, which will output the target label. We visualize the complete pipeline of using the c-BaN technique in Figure 3b.

In this section, we have introduced three techniques for implementing dynamic backdoors, namely, the Ran-

dom Backdoor, the Backdoor Generating Network (BaN), and the conditional Backdoor Generating Network (c-BaN). These three dynamic backdoor techniques present a framework to generate dynamic backdoors for different settings. For instance, our framework can generate target specific triggers' pattern using the c-BaN, or target specific triggers' location like the Random Backdoor and BaN. More interestingly, our framework allows the adversary to customize their backdoor by adapting the backdoor loss functions. For instance, the adversary can adapt to different defenses against the backdoor attack that can be modeled as a machine learning model. This can be achieved by adding any defense as a discriminator into the training of the BaN or c-BaN. Adding this discriminator will penalize/guide the backdoored model to bypass the modeled defense.

4. Evaluation

In this section, we first introduce our datasets and experimental settings. Next, we evaluate all of our three techniques, i.e., Random Backdoor, Backdoor Generating Network (BaN), and conditional Backdoor Generating Network (c-BaN). We then evaluate our three dynamic backdoor techniques against the current state-of-the-art backdoor defense techniques, and study the effect of different hyperparameters on their performance. Finally, we demonstrate how to relax the threat model and propose new defenses against dynamic backdoor attacks.

4.1. Datasets Description

We utilize three image datasets to evaluate our techniques, including MNIST, CelebA, and CIFAR-10. We use these three datasets since they are widely used as benchmark datasets for various security/privacy and computer vision tasks, however, our attack can be easily generalized to other datasets with different types of data (by adapting the architectures of the BaN and c-BaN). We briefly describe each of them below.

MNIST: The MNIST dataset [2] is a 10-class dataset consisting of $70,000$ grey-scale 28×28 images. Each of these images contains a handwritten digit in its center. The MNIST dataset is a balanced dataset, i.e, each class is represented with $7,000$ images.

CIFAR-10: The CIFAR-10 dataset [3] is composed of $60,000$ 32×32 colored images which are equally distributed on the following 10 classes: Airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

CelebA: The CelebA dataset [28] is a large-scale face attributes dataset with more than 200K colored celebrity images, each annotated with 40 binary attributes. We select the top three most balanced attributes including Heavy Makeup, Mouth Slightly Open, and Smiling. Then we concatenate them into 8 classes to create a multiple label classification task. For our experiments, we scale the images to 64×64 and randomly sample $10,000$ images for training, and another $10,000$ for testing. Finally, it is important to mention that unlike the MNIST and CIFAR-10 datasets, this dataset is highly imbalanced.

4.2. Experimental Setup

We first present our target models, then the evaluation metrics. For the target models' architecture, we use the VGG-19 [47] for the CIFAR-10 dataset, and build our own convolution neural networks (CNN) for the CelebA and MNIST datasets. More concretely, we use 3 convolution layers and 5 fully connected layers for the CelebA CNN. And 2 convolution layers and 2 fully connected layers for the MNIST CNN. Moreover, we use dropout for both the CelebA and MNIST models to avoid overfitting.

For BaN, we use the following architecture:

Backdoor Generating Network (BaN)'s architecture:

$$
\begin{aligned}
z \rightarrow\ & \texttt{FullyConnected}(64) \\
& \texttt{FullyConnected}(128) \\
& \texttt{FullyConnected}(128) \\
& \texttt{FullyConnected}(|t|) \\
& \texttt{Sigmoid} \rightarrow t
\end{aligned}
$$

Here, $\texttt{FullyConnected}(x)$ denotes a fully connected layer with x hidden units, $|t|$ denotes the size of the required trigger, and $\texttt{Sigmoid}$ is the Sigmoid function. We adopt ReLU as the activation function for all layers, and apply dropout after all layers except the first and last ones.

For c-BaN, we use the following architecture:

conditional Backdoor Generating Network (c-BaN)'s architecture:

$$
\begin{aligned}
z, \ell \rightarrow\ & 2 \times \texttt{FullyConnected}(64) \\
& \texttt{FullyConnected}(128) \\
& \texttt{FullyConnected}(128) \\
& \texttt{FullyConnected}(128) \\
& \texttt{FullyConnected}(|t|) \\
& \texttt{Sigmoid} \rightarrow t
\end{aligned}
$$

The first layer consists of two separate fully connected layers, where each one of them takes an independent input, i.e., the first takes the noise vector z and the second takes the target label ℓ. The outputs of these two layers are then concatenated and used as an input to the next layer (see Section 3.3). Similar to BaN, we adopt ReLU as the activation function for all layers and apply dropout after all layers except the first and last one.

For evaluating the dynamic backdoor attacks' performance, we define the following two metrics: *Backdoor success rate* which calculates the backdoored model's accuracy on the backdoored data; *Model utility* which measures the original functionality of the backdoored model. We quantify the model utility by comparing the accuracy of the backdoored model with the accuracy of a clean model on clean data. Closer accuracies imply a better model utility. All of our experiments are implemented using Pytorch and our code will be published for reproducibility.

4.3. Random Backdoor

We now evaluate the performance of our first dynamic backdooring technique, namely, the Random Backdoor.

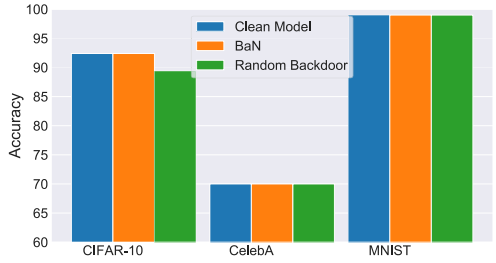


Figure 4: The result of our dynamic backdoor techniques for a single target label on the clean testing dataset.

We use all three datasets for the evaluation. First, we evaluate the single target label case, where we only implement a backdoor for a single target label. Then we evaluate the more generalized case, i.e., the multiple target labels case, where we implement a backdoor for all possible labels in the dataset.

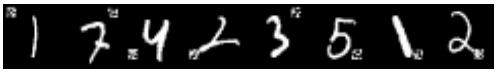
For both the single and multiple target label cases, we split each dataset into training and testing datasets. The training dataset is used to train the MNIST and CelebA models from scratch. For CIFAR-10, we use a pre-trained VGG-19 model. For evaluating our models, we use the testing dataset as our clean testing dataset. And construct a backdoored testing dataset, by adding triggers to all members of the testing dataset. To recap, for the Random Backdoor technique, we construct the triggers by sampling them from uniform distribution, and add them to the images using the backdoor adding function $\mathcal{A}$. We use the backdoored testing dataset to calculate the backdoor success rate, and the training dataset to train a clean model -for each dataset- to evaluate the backdoored model's ($\mathcal{M}_{bd}$) utility.

We follow Section 3.1 to train our backdoored model $\mathcal{M}_{bd}$ for both the single and multiple target labels cases. Abstractly, for each epoch, we update the backdoored model $\mathcal{M}_{bd}$ using both the clean and backdoor losses $\varphi_c + \varphi_{bd}$. For the set of possible locations $\mathcal{K}$, we use four possible locations.

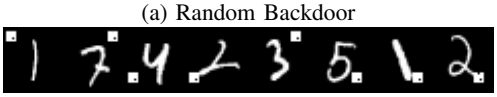
The backdoor success rate is always 100% for both the single and multiple target labels cases on all three datasets, hence, we only focus on the backdoored model's ($\mathcal{M}_{bd}$) utility.

Single Target Label: We first present our results for the single target label case. Figure 4 compares the accuracies of the backdoored model $\mathcal{M}_{bd}$ and the clean model $\mathcal{M}$. As the figure shows, our backdoored models achieve the same performance as the clean models for both the MNIST and CelebA datasets, i.e., 99% for MNIST and 70% for CelebA. For the CIFAR-10 dataset, there is a slight drop in performance, which is less than 2%. This shows that our Random Backdoor technique can implement a perfectly functioning backdoor, i.e., the backdoor success rate of $\mathcal{M}_{bd}$ is 100% on the backdoored testing dataset, with a negligible utility loss.

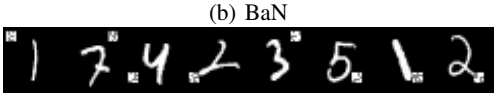
To visualize the output of our Random Backdoor technique, we first randomly sample 8 images from the MNIST dataset, and then use the Random Backdoor technique to construct triggers for them. Finally, we add these triggers to the images using the backdoor adding function $\mathcal{A}$, and show the result in Figure 5a. As the figure shows,



(a) Random Backdoor



(b) BaN



(c) BaN with higher randomness

Figure 5: The result of our Random Backdoor (Figure 5a), BaN (Figure 5b), and BaN with higher randomness (Figure 5c) techniques for a single target label (0).

the triggers all look distinctly different and are located at different locations as expected.

Multiple Target Labels: Second, we present our results for the multiple target label case. To recap, we consider all possible labels for this case. For instance, for the MNIST dataset, we consider all digits from 0 to 9 as our target labels. We train our Random Backdoor models for the multiple target labels as mentioned in Section 3.1.

We use a similar evaluation setting to the single target label case, with the following exception. To evaluate the performance of the backdoored model $\mathcal{M}_{bd}$ with multiple target labels, we construct a backdoored testing dataset for each target label by generating and adding triggers to the clean testing dataset. In other words, we use all images in the testing dataset to evaluate all possible labels.

Similar to the single target label case, we focus on the accuracy on the clean testing dataset, since the backdoor success rate for all models on the backdoored testing datasets are approximately 100% for all target labels.

We use the clean testing datasets to evaluate the backdoored model's $\mathcal{M}_{bd}$ utility, i.e., we compare the performance of the backdoored model $\mathcal{M}_{bd}$ with the clean model $\mathcal{M}$ in Figure 6. As the figure shows, using our Random Backdoor technique, we are able to train backdoored models that achieve similar performance as the clean models for all datasets. For instance, for the CIFAR-10 dataset, our Random Backdoor technique achieves 92% accuracy, which is very similar to the accuracy of the clean model (92.4%). For the CelebA dataset, the Random Backdoor technique achieves a slightly (about 2%) better performance than the clean model. We believe this is due to the regularization effect of the Random Backdoor technique. Finally, for the MNIST dataset, both models achieve a similar performance with just 1% difference between the clean model (99%) and the backdoored one (98%).

To visualize the output of our Random Backdoor technique on multiple target labels, we construct triggers for all possible labels in the CIFAR-10 dataset, and use $\mathcal{A}$ to add them to a randomly sampled image from the CIFAR-10 clean testing dataset. Figure 7a shows the image with different triggers. The different patterns and locations used for the different target labels can be clearly demonstrated in Figure 7a. For instance, comparing the location of the trigger for the first and sixth images, the triggers are in the same horizontal position but a different vertical position,

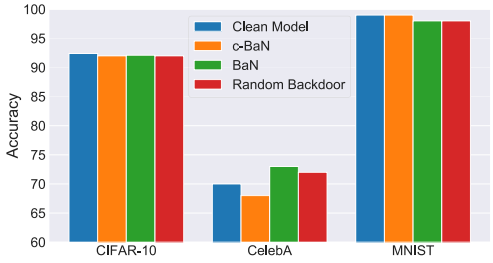


Figure 6: The result of our dynamic backdoor techniques for multiple target label on the clean testing dataset.

as previously illustrated in Figure 2.

Moreover, we further visualize in Figure 9a the dynamic behavior of the triggers generated by our Random Backdoor technique. Without loss of generality, we generate triggers for the target label 5 (plane) and add them to randomly sampled CIFAR-10 images. To make it clear, we train the backdoor model $\mathcal{M}_{bd}$ for all possible labels set as target labels, but we visualize the triggers for a single label to show the dynamic behaviour of our Random Backdoor technique with respect to the triggers' pattern and locations. As Figure 9a shows, the generated triggers have different patterns and locations for the same target label, which achieves our desired dynamic behavior.

4.4. Backdoor Generating Network (BaN)

Next, we evaluate our BaN technique. We follow the same evaluation settings for the Random Backdoor technique, except with respect to how the triggers are generated. We train our BaN model and generate the triggers as mentioned in Section 3.2.

Single Target Label: Similar to the Random Backdoor, the BaN technique achieves a perfect backdoor success rate with a negligible utility loss. Figure 4 compares the performance of the backdoored models -trained using the BaN technique- with the clean models, when tested using the clean testing dataset. As Figure 4 shows, our BaN trained backdoored models achieve 99%, 92.4% and 70% accuracy on the MNIST, CIFAR-10, and CelebA datasets, respectively, which is the same performance of the clean models.

We visualize the BaN generated triggers using the MNIST dataset in Figure 5b. To construct the figure, we use the BaN to generate multiple triggers -for the target label 0-, then we add them on a set of randomly sampled MNIST images using the backdoor adding function $\mathcal{A}$.

The generated triggers look very similar as shown in Figure 5b. This behaviour is expected as the MNIST dataset is simple, and the BaN technique does not have any explicit loss to enforce the network to generate different triggers. However, to show the flexibility of our approach, we increase the randomness of the BaN network by simply adding one more dropout layer after the last layer, to avoid the overfitting of the BaN model to a unique pattern. We show the results of the BaN model with higher randomness in Figure 5c. The resulting model still achieves the same performance, i.e., 99% accuracy on the clean data and 100% backdoor success rate, but as the figure shows the triggers look significantly different. This again shows that our framework can easily adapt to the requirements of an adversary.

These results together with the results of the Random Backdoor (Section 4.3) clearly show the effectiveness of both of our proposed techniques, for the single target label case. They are both able to achieve almost the same accuracy of a clean model, with a 100% working backdoor, for a single target label.

Multiple Target Labels: Similar to the single target label case, we focus on the backdoored models' performance on the clean testing dataset, as our BaN backdoored models achieve a perfect accuracy on the backdoored testing dataset, i.e., the backdoor success rate for all datasets is approximately 100% for all target labels.

We compare the performance of the BaN backdoored models with one of clean models using the clean testing dataset in Figure 6. Our BaN backdoored models are able to achieve almost the same accuracy as the clean model for all datasets, as can be shown in Figure 6. For instance, for the CIFAR-10 dataset, our BaN achieves 92.1% accuracy, which is only 0.3% less than the performance of the clean model (92.4%). Similar to the Random Backdoor backdoored models, our BaN backdoored models achieve a marginally better performance for the CelebA dataset. More concretely, our BaN backdoored models trained for the CelebA dataset achieve about 2% better performance than the clean model, on the clean testing dataset. We also believe this improvement is due to the regularization effect of the BaN technique. Finally, for the MNIST dataset, our BaN backdoored models achieve strong performance on the clean testing dataset (98%), which is just 1% lower than the performance of the clean models (99%).

Similar to the Random Backdoor, we visualize the results of the BaN backdoored models with two figures. The first (Figure 7b) shows the different triggers for the different target labels on the same CIFAR-10 image, and the second (Figure 9b) shows the different triggers for the same target label (plane) on randomly sampled CIFAR-10 images. As both figures show, the BaN generated triggers achieves the dynamic behaviour in both locations and patterns. For instance, for the same target label (Figure 9b), the patterns of the triggers look significantly different and the locations vary vertically. Similarly, for different target labels (Figure 7b), both the pattern and location of triggers are significantly different.

4.5. conditional Backdoor Generating Network (c-BaN)

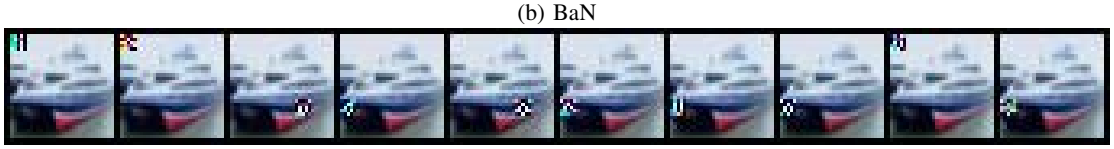
Next, we evaluate our conditional Backdoor Generating Network (c-BaN) technique. For the single target label case, the c-BaN technique is the same as the BaN technique. Thus, we only consider the multiple target labels case in this section.

We follow a similar setup as the one introduced in Section 4.4, with the exception on how to train the backdoored model $\mathcal{M}_{bd}$ and generate the triggers. We follow Section 3.3 to train the backdoored model and generate the triggers. For the set of possible locations $\mathcal{K}$, we use four possible locations.

We compare the performance of the c-BaN with the other two techniques in addition to the clean model. All of

(a) Random Backdoor

(b) BaN



(c) c-BaN

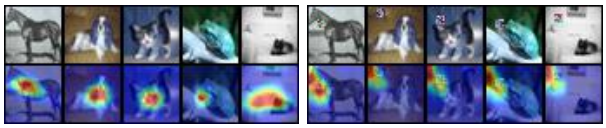
Figure 7: The visualization result of our Random Backdoor (Figure 7a), BaN (Figure 7b), and c-BaN (Figure 7c) techniques for all labels of the CIFAR-10 dataset.

our three dynamic backdoor techniques achieve an almost perfect backdoor success rate on the backdoored testing datasets, hence similar to the previous sections, we focus on the performance on the clean testing datasets.

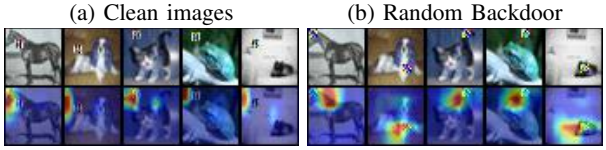
Figure 6 compares the accuracy of the backdoored and clean models using the clean testing dataset, for all of our three dynamic backdoor techniques. As the figure shows, all of our dynamic backdoored models have similar performance as the clean models. For instance, for the CIFAR-10 dataset, our c-BaN, BaN and Random Backdoor achieve 92%, 92.1% and 92% accuracy, respectively, which is very similar to the accuracy of the clean model (92.4%). Also for the MNIST dataset, all models achieve very similar performance with no difference between the clean and c-BaN models (99%) and only 1% difference between them, and the BaN and Random Backdoor models (98%).

Similar to the previous two techniques, we visualize the dynamic behaviour of the c-BaN backdoored models using two different figures. First, by generating triggers for all possible labels and adding them on a CIFAR-10 image in Figure 7c. More generally, Figure 7 shows the visualization of all three dynamic backdoor techniques in the same settings, i.e., backdooring a single image to all possible labels. As the figure shows, the Random Backdoor Figure 7a has the most random patterns, which is expected as they are sampled from a uniform distribution. The figure also shows the different triggers' patterns and locations used for the different techniques. For instance, each target label in the Random Backdoor (Figure 7a) and BaN (Figure 7b) techniques have a unique (horizontal) location, unlike the c-BaN (Figure 7c) generated triggers, which different target labels can share the same locations, as can be shown for example in the first, second, and ninth images. To recap, both the Random Backdoor and BaN techniques split the location set $\mathcal{K}$ on all target labels, such that no two labels share a location, unlike the c-BaN technique which does not have this limitation.

Second, we visualize the dynamic behaviour of our techniques, by generating triggers for the same target label 5 (plane) and adding them to a set of randomly sampled CIFAR-10 images. Figure 9 compares the visualization of



(a) Clean images (b) Random Backdoor

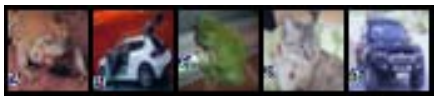


(c) BaN (d) c-BaN

Figure 8: Visualization of attention maps for all our techniques using the Grad-CAM technique.

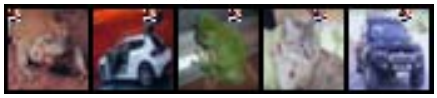
our three different dynamic backdoor techniques in this setting. More concretely, we train the backdoor model $\mathcal{M}_{bd}$ for all possible labels set as target labels. Then, for space restrictions, we plot the backdoored inputs for a single target label. As the figure shows, the Random Backdoor (Figure 9a) and BaN (Figure 9b) generated triggers can move vertically, however, they have a fixed position horizontally as mentioned in Section 3.1 and illustrated in Figure 2. The c-BaN (Figure 9c) triggers also show different locations. However, the locations of these triggers are more distant and can be shared for different target labels, unlike the other two techniques. Furthermore, the figure shows that most of our triggers have different patterns for our techniques for the same target label, which achieves our targeted dynamic behavior concerning the patterns and locations of the triggers.

Finally, we compare the attention of the backdoored models on both clean and backdoored inputs. We use the Gradient-weighted Class Activation Mapping (Grad-CAM) technique [43] to compute the attention maps for our backdoored models. These maps show the most influential parts of the input that resulted in the model's output. Figure 8 depicts the results of our three different techniques. As expected all backdoored models mainly focus on the triggers in backdoored inputs and the main objects in the clean ones.



(a) Random Backdoor

(b) BaN



(c) c-BaN

Figure 9: The result of our Random Backdoor (Figure 9a), BaN (Figure 9b), and c-BaN (Figure 9c) techniques for the target target label 5 (plane).

4.6. Evaluating Against Current State-Of-The-Art Defenses

We now evaluate our attacks against the current state-of-the-art backdoor defenses. Backdoor defenses can be classified into the following two categories, data-based defenses and model-based defenses. On the one hand, data-based defenses focus on identifying if a given input is clean or contains a trigger. On the other hand, model-based defenses focus on identifying if a given model is clean or backdoored.

We first evaluate our attacks against model-based defenses, then we evaluate them against data-based ones.

Model-based Defense: We evaluate all of our dynamic backdoor techniques in the multiple target label case against three of the current state-of-the-art model-based defenses, namely, Neural Cleanse [55], ABS [24], and MNTD [58].

We start by evaluating the ABS defense. We use the CIFAR-10 dataset to evaluate this defense, since it is the only supported dataset by the published defense model. As expected, running the ABS model against our dynamic backdoored ones does not result in detecting any backdoor for all of our models.

For Neural Cleanse, we use all three datasets to evaluate our techniques against it. Similar to ABS, all of our models are predicted to be clean models. Moreover, in multiple cases, our models had a lower anomaly index (the lower the better) than the clean model.

We believe that both of these defenses fail to detect our backdoors for two reasons. First, we break one of their main assumption, i.e., that the triggers are static in terms of location and pattern. Second, we implement a backdoor for all possible labels, which makes the detection a more challenging task.

Finally, we evaluate the MNTD defense. To this end, we use the CIFAR-10 dataset to evaluate our three backdoor techniques. Following the same setting in [21], we build 200 shadow benign and backdoored models to train 50 meta-classifiers sequentially for further evaluation. The meta-classifier takes a target model as its input and outputs a score. This score represents the likelihood of the model being backdoored, i.e., a higher score indicates the target model is more likely to be backdoored.

Our results show that the score predicted by the MNTD meta classifier drops from 67.08($\pm$ 20.49) for static backdoors to 3.05($\pm$ 0.82), 0.54($\pm$ 0.83), and 1.47($\pm$ 0.87) for the random backdoor, BaN, and cBaN backdoors. This significant reduction of scores (with at least a factor of 22$\times$) demonstrates the advantage of our techniques compared to the static ones. In this setting, each meta-classifier would output a score, then based on a threshold, the decision if the model is backdoored or not is made [21]. We use the default threshold (median of training models' scores), which results in 98%, 74%, and 98% accuracy for the random backdoor, BaN, and cBaN techniques, respectively. This percentage corresponds to the number of meta-classifiers correctly classifying the model as a backdoored one. To improve the stealthiness of our backdoored models, we add a discriminator when training the models, aiming to lower the score predicted by the MNTD meta classifier. More concretely, we train a local meta-classifier (with a disjoint dataset compared to the one used for evaluation) and use it as our discriminator. We demonstrate this with the cBaN technique; however, it can be easily extended to the other two techniques. Using this technique, our results are significantly improved, i.e., only a single meta-classifier out of the 50 classified the model as a backdoored one. In other words, the detection accuracy is dropped to 2%, with a negligible performance drop, i.e., the ASR and utility dropped by less than 1%.

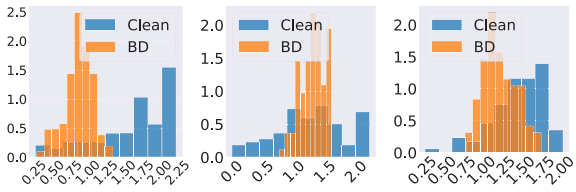
This again demonstrates that our dynamic backdoor techniques are more stealthy than the static ones. Moreover, they can be easily adapted to bypass backdoor defenses, e.g., by adding the corresponding discriminator as mentioned in Section 3.3.

Data-based Defense: Next, we evaluate some of the current state-of-the-art data-based defenses. Namely, we start by evaluating STRIP [12], then Februus [8].

STRIP tries to identify if a given input is clean or contains a trigger. It works by creating multiple images from the input image by fusing it with multiple clean images one at a time. Then STRIP applies all fused images to the target model and calculates the entropy of predicted labels. Backdoored inputs tend to have lower entropy compared to the clean ones.

We use all of our three datasets to evaluate the c-BaN models against this defense. First, we scale the patterns by half while training the backdoored models, to make them more susceptible to changes. Second, for the MNIST dataset, we move the possible locations to the middle of the image to overlap with the image content, since the value of the MNIST images at the corners are always 0. All trained scaled backdoored models achieve similar performance to the non-scaled backdoored models.

Our backdoored models successfully flatten the distribution of entropy for the backdoored data, for a subset of target labels. In other words, the distribution of entropy for our backdoored data overlaps with the distributions of entropy of the clean data. This subset of target labels makes picking a threshold to identify backdoored data from clean data impossible without increasing the false positive rate, i.e., various clean images will be detected as backdoored ones. We visualize the entropy of our best performing labels against the STRIP defense in Figure 10. Moreover, since our dynamic backdoors can generate dynamic triggers for the same input and target label,



(a) CIFAR-10 (b) MNIST (c) CelebA

Figure 10: The histogram of the entropy of the backdoored vs clean input, for our best performing labels against the STRIP defense, for the CIFAR-10 (Figure 10a), MNIST (Figure 10b), and CelebA (Figure 10c) datasets.

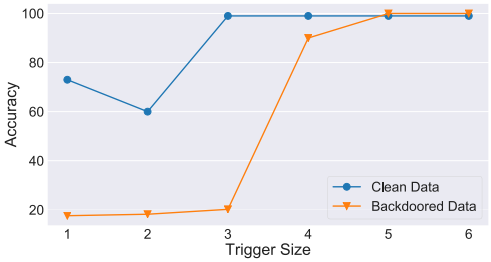


Figure 11: The result of trying different trigger sizes for the c-BaN technique on the MNIST dataset. The figure shows for each trigger size the accuracy on the clean and backdoored testing datasets.

the adversary can keep querying the target model while backdooring the input with a freshly generated trigger until the model accepts it.

Next, we evaluate Februus. Intuitively, Februus first detects the trigger from backdoored samples before removing it and patching the image. To detect these triggers, Februus first uses GradCAM to identify the influential region on the input. Then based on a security hyperparameter –which is dependent on the underlying task–, it decides if this area is to be removed and replaced by a neutral color. Finally, Februus develops a GAN-based inpainting technique to restore the image before querying it to the target model.

As the training code of Februus is not public yet, we only use CIFAR-10 – since it is the only dataset we consider that has its Februus models available – to evaluate against our different backdoor techniques.

Our results show that Februus only succeeds in dropping the ASR of our random backdoor, BaN, and cBaN backdoored models from 100% to approximately 80.5%, 81.7%, and 72%, respectively. This demonstrates the strong performance of our attack against the data-based defenses, especially compared to the static backdoored – whose ASR drops to 0.25% when applying Februus–.

These results against the data and model-based defenses show the effectiveness of our dynamic backdoor attacks, and opens the door for designing backdoor detection systems that work against both static and dynamic backdoors.

4.7. Evaluating Different Hyperparameters

We now evaluate the effect of different hyperparameters for our dynamic backdooring techniques. We start by evaluating the percentage of the backdoored data needed to implement a dynamic backdoor into the model. Then, we evaluate the effect of increasing the size of the location set $\mathcal{K}$. Finally, we evaluate the size of the trigger and the possibility of making it more transparent, i.e., instead of replacing the original values in the input with the backdoor, we fuse them.

Proportion of the Backdoored Data: We start by evaluating the percentage of backdoored data needed to implement a dynamic backdoor in the model. We use the MNIST dataset and the c-BaN technique to perform the evaluation. First, we construct different training datasets with different percentages of backdoored data. More concretely, we try all proportions from 10% to 50%, with a step of 10. In this setting, 10% means that 10% of the data is backdoored, and 90% is clean. Our results show that using 30% is already enough to get a perfectly working dynamic backdoor, i.e., the model has a similar performance like a clean model on the clean dataset (99% accuracy), and 100% backdoor success rate on the backdoored dataset. For any percentage below 30%, the accuracy of the model on clean data is still the same, however, the performance on the backdoored dataset starts degrading. This demonstrates the ability of the adversary to implement dynamic backdoor attacks with 30% overhead for each target label, compared to training a clean model.

Number of Locations: Second, we explore the effect of increasing the size of the set of possible locations ($\mathcal{K}$) for the c-BaN technique. We use the CIFAR-10 dataset to train a backdoored model using the c-BaN technique, but with more than double the size of $\mathcal{K}$, i.e., 8 locations. The trained model achieves similar performance on the clean (92%) and backdoored (100%) datasets. We then doubled the size again to have 16 possible locations in $\mathcal{K}$, and the model again achieves the same results on both clean and backdoored datasets. We repeat the experiment with the CelebA datasets and achieve similar results, i.e., the performance of the model with a larger set of possible locations is similar to the previously reported one. However, when we try to completely remove the location set $\mathcal{K}$ and consider all possible locations with a sliding window, the performance on both clean and backdoored datasets drops significantly.

Trigger Size: Next, we evaluate the effect of the trigger size on our c-BaN technique using the MNIST dataset. We train different models with the c-BaN technique, while setting the trigger size from 1 to 6. We define the trigger size to be the width and height of the trigger. For instance, a trigger size of 3 means that the trigger is 3×3 pixels.

We calculate the accuracy on the clean and backdoored testing datasets for each trigger size, and show our results in Figure 11. Our results show that the smaller the trigger, the harder it is for the model to implement the backdoor behaviour. Moreover, small triggers confuse the model, which results in reducing the model's utility. As Figure 11 shows, a trigger with the size 5 achieves a perfect accuracy (100%) on the backdoored testing dataset, while preserving the accuracy on the clean testing dataset (99%).

Transparency of the Triggers: Finally, we evaluate the effect of making the trigger more transparent. More specif-

Figure 12: An illustration of the effect of using different transparency scales (from 0 to 1 with step of 0.25) when adding the trigger. Scale 0 (the most left image) shows the original input, and scale 1 (the most right image) the original backdoored input without any transparency.

Figure 13: Visualization of the c-BaN backdoored images when setting the transparency scale to 0.1.

ically, we change the backdoor adding function $\mathcal{A}$ to apply a weighted sum, instead of replacing the original input's values. Abstractly, we define the weighted sum of the trigger and the image as: $x_{bd} = s \cdot t + (1 - s) \cdot x$, where s is the scale controlling the transparency rate, x is the input and t is the trigger. We implement this weighted sum only at the location of the trigger, while maintaining the remaining of the input unchanged.

We use the MNIST dataset and c-BaN technique to evaluate the scale from 0 to 1, with a step of 0.25. Figure 12 visualizes the effect of varying the scale when adding a trigger to an input.

Our results show that our technique can achieve the same performance on both the clean (99%) and backdoored (100%) testing datasets, when setting the scale to 0.5 or higher. However, when the scale is set below 0.5, the performance starts degrading on the backdoored dataset but stays the same on the clean dataset. We repeat the same experiments for the CelebA and CIFAR-10 datasets and find similar results.

We believe that the transparency of our triggers can be further increased when using triggers with larger sizes. To this end, we use the CIFAR-10 dataset to repeat the experiments previously mentioned in this section. However, we set the trigger size to be the size of the image. Our experiments show that in this setting, our dynamic backdoor attacks can still achieve a perfect attack success rate (100%) with a negligible drop in utility (0.3%) when setting the scale to 0.1. More concretely, the model's accuracy on clean data is 91.7% compared to the 92% accuracy of the backdoored model trained without any transparency. We visualize a set of randomly backdoored samples in Figure 13. As the figure shows, setting the scale to 0.1 makes the triggers hardly visible.

4.8. Relaxing the Threat Model (Transferability of the Triggers)

For our dynamic backdoor attacks, we assume the adversary to control the training of the target model. We now relax this assumption by only allowing them to poison the dataset.

First, it is important to mention that our Random Backdoor technique does not need to change the training of the target model, i.e., the adversary only needs to poison the training dataset with backdoored images and the corresponding target labels. Second, for both the BaN and c-BaN techniques, the adversary can rely on pre-trained BaN and c-BaN models instead of training them jointly with the target model. In detail, the adversary uses the pre-trained BaN and c-BaN model to generate multiple triggers and randomly place them to a set of – randomly picked – images. Then, they poison the training set with this set of backdoored images and their corresponding target labels.

We use the MNIST dataset for evaluation and follow the same target models' structure as previously introduced in Section 4.4 and Section 4.5. However, to show the flexibility of our techniques, we use data from different distributions to pre-train the BaN and c-BaN models. We first use the CIFAR-10 dataset to train backdoored models with the BaN (Section 3.2) and c-BaN (Section 3.3) techniques. Next, we use the pre-trained BaN and c-BaN models to generate the backdoored dataset and poison the target dataset as previously mentioned. It is important to mention that the CIFAR-10 based BaN and c-BaN models generate 3-channel triggers, to use them to poison the MNIST dataset, we convert them to 1-channel triggers by taking the mean over the different channels. Finally, we use the poisoned dataset to train the target model.

As expected, the backdoored models achieve a perfect attack success rate (100%), while keeping the same utility as the backdoored models jointly trained with the BaN and c-BaN. This shows the flexibility of our attacks, i.e., the training procedure can be adapted by the adversary depending on their specific application. However, it is important to mention that jointly training the models has the advantage of giving the adversary more power, e.g., they can add a customized loss function to the target model while implanting the backdoor.

Finally, as a side-effect of transferring the BaN and c-BaN; The poisoning rate for the dynamic backdoor can now be lowered to about 10%, as there is no joint models trained with the target model anymore.

4.9. Possible Defenses

Finally, we propose some possible defenses against our dynamic backdoor attacks. Intuitively, we use a denoising mechanism to filter triggers (as they can be considered as anomalies/distortions) out of the backdoored inputs. To this end, we use one of the most common denoising mechanisms, namely autoencoder. It works as follows: First, we train an autoencoder on clean data. Then, we use this autoencoder to reconstruct/denoise the inputs (by encoding then decoding them). The noise or triggers in our case are expected to be filtered out of the inputs due to two main reasons. First, the overfitting of the autoencoders to clean data, and second, the lossy reconstruction process.

To implement our defense, we use the autoencoder to denoise all inputs before forwarding them to the target model. The autoencoder is expected to remove the trigger from backdoored data, while not significantly changing the clean ones.

To evaluate the efficacy of our proposed defense, we test it against the c-BaN technique, using both the MNIST and CIFAR-10 datasets. As expected, the backdoored images are not perfectly reconstructed by the autoencoder, i.e., the autoencoder does not fully reconstruct

triggers. Our experiments show that in simple datasets like MNIST, our approach can successfully defend against the backdoor attack, with negligible utility loss (less than 1%). However, for more complicated datasets like CIFAR-10, the performance of our defense degrades. This is due to the high amount of details which hardens the reconstruction process of complex datasets (for both clean and backdoored inputs). For instance, the accuracy of the target model drops by 4.8% and 25% for the clean and backdoored dataset, respectively.

Another possible defense approach is first calculating the distance between the reconstructed input and the original input, then taking the decision to forward the input or not to the model, based on a predetermined threshold. We plan to explore this approach and other potential methods in future work.

We now discuss another defense, namely data augmentation. More concretely, we discuss the effect of resizing, cropping, and flipping the target images on our dynamic backdoor attacks. To this end, we use the CIFAR-10 dataset and test how resizing, cropping, or flipping the backdoored image before querying it to the backdoored model affects the performance, i.e., the ASR and utility. We evaluate the performance of our simplest and most complex setting, i.e., the random backdoor with a single target label and the cBaN with all possible target labels. We start with the flipping operation, i.e., we flip each input before querying it to the target model. Our results show that flipping the inputs reduces the ASR to approximately 88.6% and 93.4%, without having a significant effect on the utility for the cBaN and random backdoor, respectively. This shows that our dynamic backdoor attacks are resilient to flipping. Second, we test the resizing, i.e., we downsize the input image to 16x16 pixels before scaling it back to 32x32 pixels (the model's expected input size). Resizing the inputs reduced our ASR to approximately 57.4% and 66.5%. However, it also dropped the utility by 15.4% and 15.9% for the random backdoor and cBaN, respectively. This shows that scaling can drop our attack performance by on average 40%, at the cost of a more than 15% reduction in utility. Finally, for cropping, we pad all boarder of the input image by 4 black pixels, i.e., with the value 0, then we randomly select a location to crop the padded image back to its original size (32x32). Our results show that cropping drops the ASR to about 73.2% and 89.2%, while the accuracy drops by 0.7% and 0.25% for the cBaN and random backdoor models, respectively.

We next include the three data augmentation techniques in the training of the models and repeat the same experiments, i.e., testing the effect of applying each data augmentation separately at the inference time. We observe that the results did not differ significantly from the previous set of experiments; hence we plot the result in the Appendix (Figure 14).

These experiments show that data augmentations can reduce the performance of our dynamic backdoor attacks; however, they cannot prevent it and can drop the utility significantly. In other words, our attacks are still applicable but with a reduced ASR when applying different data augmentation techniques.

5. Related Work

In this section, we discuss some of the related work. We start with current state-of-the-art backdoor attacks. Then we discuss the defenses against backdoor attacks, and finally mention other attacks against machine learning models.

Backdoor Attacks: Gu et al. [14] introduce BadNets, the first backdoor attack on machine learning models. BadNets uses the MNIST dataset and a square-like trigger with a fixed location, to show the applicability of the backdoor attacks in the ML settings. Liu et al. [25] later propose a more advanced backdooring technique, namely the Trojan attack. They simplify the threat model of BadNets by eliminating the need for access to the training data used to train the target model. The Trojan attack reverse-engineers the target model to synthesize training data. Next, it generates the trigger in a way that maximizes the activation functions of the target model's internal neurons related to the target label. In other words, the Trojan attack reverse-engineers a trigger and training data to retrain/update the model and implement the backdoor.

The main difference between these two attacks (BadNets and Trojan attacks) and our work is that both attacks only consider static backdoors in terms of triggers' pattern and location. Our work extends the backdoor attacks to consider dynamic patterns and locations of the triggers.

Nguyen and Tran [32] present an input-aware dynamic backdoor. Intuitively, they propose a trigger generating network that generates independent triggers for each input, i.e., a new unique trigger is generated for every input. One main difference between their and our work is the structure of generated triggers. In our work, we model the triggers to be square-like, while they model it to be scattered pixels/patterns across the image. One advantage of our triggers is that they can be applied to physical objects/images. For instance, the adversary can print our triggers and attach them to the image/object. We also show – in Section 4.8 – how to relax our threat model to only include poisoning of the training dataset, unlike [32] which requires full access to the training of the target model. Finally, using our approach, the adversary can generate multiple triggers for the same image, which results in a more flexible attack. More concretely, if the defender gets access to the backdoored image, the adversary can still trigger the same image using different triggers for the same target label.

We focus on backdoor attacks against image classification models, but backdoor attacks can be extended to other scenarios, such as Federated Learning [56], Video Recognition [62], Transfer Learning [59], and Natural Language Processing (NLP) [7].

To increase the stealthiness of the backdoor, Saha et al. [39] propose to transform the backdoored images into benign-looking ones, which makes them harder to detect. Lie et al. [27] introduce another approach, namely, the reflection backdoor (Refool), which hides the triggers using mathematical modeling of the physical reflection property. Another line of research focuses on exploring different methods of implementing backdoors into target models. Rakin et al. [38] introduce the Targeted Bit Trojan (TBT) technique, which instead of training the target model, flips some bits in the target models' weights to

make it misclassify all the inputs. Tang et al. [50] present a different approach, where the adversary appends a small Trojan module (TrojanNet) to the target model instead of fully retraining it.

Defenses Against Backdoor Attacks: Defenses against backdoor attacks can be classified into model-based defenses and data-based defenses. First, model-based defenses try to find if a given model contains a backdoor or not. For instance, Wang et al. [55] propose Neural Cleanse (NC), a backdoor defense method based on reverse engineering. For each output label, NC tries to generate the smallest trigger, which converts the output of all inputs applied with this trigger to that label. NC then uses anomaly detection to find if any of the generated triggers are actually a backdoor or not. Later, Liu et al. [24] propose another model-based defense, namely, ABS. ABS detects if a target model contains a backdoor or not, by analyzing the behaviour of the target model's inner neurons when introducing different levels of stimulation. Xu et al. [58] present another model-based defense, namely MNTD. Abstractly, MNTD builds a meta classifier to detect if a given model is backdoored or not. To this end, it first starts by using a small set of clean data to train multiple shadow clean and backdoored models using different triggers. Next, it optimizes a set of probing points which is used to query all training models. Finally, the defender queries each shadow model using the probing points and uses pairs of corresponding predictions and ground-truth labels (i.e., clean or backdoored) to train a meta-classifier. The meta classifier takes as input a model then outputs a score for each model, indicating if the model is backdoored or not.

Second, data-based defenses try to find if a given input is clean or backdoored. For instance, Gao et al. [12] propose STRIP, a backdoor defense method based on manipulating the input, to find out if it is backdoored or not. More concretely, STRIP fuses the input with multiple clean data, one at a time. Then it queries the target model with the generated inputs, and calculates the entropy of the output labels. Backdoored inputs tend to have lower entropy than clean ones.

Similarly, Doan et al. [8] presents Februus, which is another data-based defense. Intuitively, Februus uses GradCAM to find the most contributing regions of the input (with respect to the model's output). Then it determines if the assigned regions contain a trigger or not. If it contains a trigger, then Februus removes this part from the input and uses a GAN-based inpainting technique to complete the input again before forwarding it to the model.

Attacks Against Machine Learning: Poisoning attack [4], [19], [40], [49] is another training time attack, in which the adversary manipulates the training data to compromise the target model. For instance, the adversary can change the ground truth for a subset of the training data to manipulate the decision boundary, or more generally influence the model's behavior. Shafahi et al. [44] further introduce the clean label poisoning attack. Instead of changing labels, the clean label poisoning attack allows the adversary to modify the training data itself to manipulate the behaviour of the target model.

Another class of ML attacks is the adversarial examples. Adversarial examples share some similarities with the backdoor attacks. In this setting, the adversary aims to trick a target classifier into miss classifying a data point by adding controlled noise to it. Multiple work has explored the privacy and security risks of adversarial examples [5], [23], [36], [37], [51], [53], [57]. Other work explores the adversarial example's potentials in preserving the user's privacy in multiple domains [20], [22], [34], [61]. The main difference between adversarial examples and backdoor attacks is that backdoor attacks are performed in training time, while adversarial examples are performed after the model is trained and without changing any of the model's parameters.

Beside the above, there are multiple other types of attacks against machine learning models [26], such as membership inference [6], [15]–[17], [29], [31], [42], [46], [48], [60], model stealing [35], [45], [52], [54], model inversion [9], [10], [18], property inference [11], [30], [63], and dataset reconstruction [41].

6. Conclusion

The tremendous progress of machine learning has lead to its adoption in multiple critical real-world applications. However, it has been shown that ML models are vulnerable to various types of security and privacy attacks. In this paper, we focus on backdoor attacks where an adversary manipulates the training of the model to intentionally misclassify any input with an added trigger.

Current backdoor attacks only consider static triggers in terms of patterns and locations. In this work, we propose the first set of dynamic backdoor attacks against deep neural networks (DNN) models, where the trigger can have multiple patterns and locations. To this end, we propose three different techniques.

Our first technique Random Backdoor samples triggers from a uniform distribution and places them at random locations of an input. For the second technique, i.e., Backdoor Generating Network (BaN), we propose a novel generative network to construct triggers. Finally, we introduce conditional Backdoor Generating Network (c-BaN) to generate label specific triggers.

We evaluate our techniques using three benchmark datasets. The evaluation shows that all our techniques can achieve almost a perfect backdoor success rate while preserving the model's utility. Moreover, we show that our techniques successfully bypass state-of-the-art defense mechanisms against backdoor attacks.

Acknowledgment

Ahmed Salem's work was mostly done while at CISPA Helmholtz Center for Information Security. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ ERC grant agreement no. 610150-imPACT, from the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4) and from U.S. IARPA TrojAI W911NF-19-S-0012.

References

[1] https://www.apple.com/iphone/#face-id.

[2] http://yann.lecun.com/exdb/mnist/.

[3] https://www.cs.toronto.edu/~kriz/cifar.html.

[4] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks against Support Vector Machines," in *International Conference on Machine Learning (ICML)*. icml.cc / Omnipress, 2012.

[5] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 39–57.

[6] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020, pp. 343–362.

[7] X. Chen, A. Salem, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang, "BadNL: Backdoor Attacks Against NLP Models with Semantic-preserving Improvements," in *Annual Computer Security Applications Conference (ACSAC)*. ACSAC, 2021, pp. 554–569.

[8] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems," in *Annual Computer Security Applications Conference (ACSAC)*. ACM, 2020, pp. 897–912.

[9] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2015, pp. 1322–1333.

[10] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2014, pp. 17–32.

[11] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 619–633.

[12] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A Defence Against Trojan Attacks on Deep Neural Networks," in *Annual Computer Security Applications Conference (ACSAC)*. ACM, 2019, pp. 113–125.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2014, pp. 2672–2680.

[14] T. Gu, B. Dolan-Gavitt, and S. Grag, "Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain," *CoRR abs/1708.06733*, 2017.

[15] I. Hagestedt, Y. Zhang, M. Humbert, P. Berrang, H. Tang, X. Wang, and M. Backes, "MBeacon: Privacy-Preserving Beacons for DNA Methylation Data," in *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[16] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing Links from Graph Neural Networks," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2021, pp. 2669–2686.

[17] X. He and Y. Zhang, "Quantifying and Mitigating Privacy Risks of Contrastive Learning," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, pp. 845–863.

[18] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 603–618.

[19] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, pp. 19–35.

[20] J. Jia and N. Z. Gong, "AttriGuard: A Practical Defense Against Attribute Inference Attacks via Adversarial Machine Learning," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2018, pp. 513–529.

[21] J. Jia, Y. Liu, and N. Z. Gong, "BadEncoder: Backdoor Attacks to Pre-trained Encoders in Self-Supervised Learning," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

[22] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 259–274.

[23] B. Li and Y. Vorobeychik, "Scalable Optimization of Randomized Operational Decisions in Adversarial Classification Settings," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR, 2015, pp. 599–607.

[24] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 1265–1282.

[25] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning Attack on Neural Networks," in *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2018.

[26] Y. Liu, R. Wen, X. He, A. Salem, Z. Zhang, M. Backes, E. D. Cristofaro, M. Fritz, and Y. Zhang, "ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2022.

[27] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 182–199.

[28] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes in the Wild," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 3730–3738.

[29] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding Membership Inferences on Well-Generalized Learning Models," *CoRR abs/1802.04889*, 2018.

[30] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting Unintended Feature Leakage in Collaborative Learning," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 497–512.

[31] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 1021–1035.

[32] T. A. Nguyen and A. Tran, "Input-Aware Dynamic Backdoor Attack," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[33] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, "Towards Reverse-Engineering Black-Box Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[34] S. J. Oh, M. Fritz, and B. Schiele, "Adversarial Image Perturbation for Privacy Protection – A Game Theory Perspective," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 1482–1491.

[35] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing Functionality of Black-Box Models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4954–4963.

[36] N. Papernot, P. D. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks Against Machine Learning," in *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2017, pp. 506–519.

[37] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in *IEEE European Symposium on Security and Privacy (Euro S&P)*. IEEE, 2016, pp. 372–387.

[38] A. S. Rakin, Z. He, and D. Fan, "TBT: Targeted Neural Network Attack with Bit Trojan," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 13 198–13 207.

[39] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden Trigger Backdoor Attacks," in *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2020, pp. 11 957–11 965.

[40] A. Salem, M. Backes, and Y. Zhang, "Get a Model! Model Hijacking Attack Against Machine Learning Models," in *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2022.

[41] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, pp. 1291–1308.

[42] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models," in *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[43] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 618–626.

[44] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2018, pp. 6103–6113.

[45] Y. Shen, X. He, Y. Han, and Y. Zhang, "Model Stealing Attacks Against Inductive Graph Neural Networks," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

[46] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 3–18.

[47] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[48] C. Song and V. Shmatikov, "Auditing Data Provenance in Text-Generation Models," in *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2019, pp. 196–206.

[49] O. Suciu, R. Mărginean, Y. Kaya, H. D. III, and T. Dumitraş, "When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2018, pp. 1299–1316.

[50] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks," in *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2020, pp. 218–228.

[51] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses," in *International Conference on Learning Representations (ICLR)*, 2017.

[52] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2016, pp. 601–618.

[53] Y. Vorobeychik and B. Li, "Optimal Randomized Classification in Adversarial Settings," in *International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. IFAAMAS/ACM, 2014, pp. 485–492.

[54] B. Wang and N. Z. Gong, "Stealing Hyperparameters in Machine Learning," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, pp. 36–52.

[55] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 707–723.

[56] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J. yong Sohn, K. Lee, and D. Papailiopoulos, "Attack of the Tails: Yes, You Really Can Backdoor Federated Learning," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[57] W. Xu, D. Evans, and Y. Qi, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," in *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2018.

[58] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting AI Trojans Using Meta Neural Analysis," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[59] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent Backdoor Attacks on Deep Neural Networks," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 2041–2055.

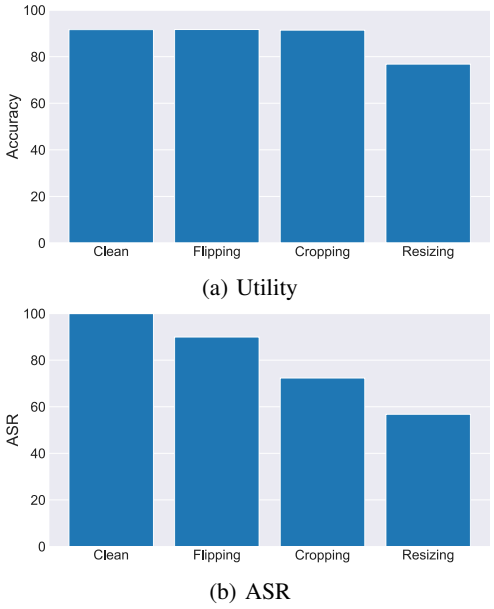
[60] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting," in *IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 268–282.

[61] Y. Zhang, M. Humbert, T. Rahman, C.-T. Li, J. Pang, and M. Backes, "Tagvisor: A Privacy Advisor for Sharing Hashtags," in *The Web Conference (WWW)*. ACM, 2018, pp. 287–296.

[62] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "Clean-Label Backdoor Attacks on Video Recognition Models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 14 443–144 528.

[63] J. Zhou, Y. Chen, C. Shen, and Y. Zhang, "Property Inference Attacks Against GANs," in *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2022.

Appendix



(a) Utility

(b) ASR

Figure 14: The performance of the cBaN technique when applying data augmentations techniques when training the target model. Figure 14a and Figure 14b shows the utility and ASR, respectively.