Backdoor Attacks on Time Series: A Generative Approach

Yujing Jiang[1], Xingjun Ma[2], Sarah Monazam Erfani[1], James Bailey[1]
[1]*Faculty of Engineering and Information Technology, The University of Melbourne*
{yujingj@student., sarah.erfani@, baileyj@}unimelb.edu.au
[2]*School of Computer Science, Fudan University*
xingjunma@fudan.edu.cn

***Abstract*—Backdoor attacks have emerged as one of the major security threats to deep learning models as they can easily control the model's test-time predictions by pre-injecting a backdoor trigger into the model at training time. While backdoor attacks have been extensively studied on images, few works have investigated the threat of backdoor attacks on time series data. To fill this gap, in this paper we present a novel generative approach for time series backdoor attacks against deep learning based time series classifiers. Backdoor attacks have two main goals: high stealthiness and high attack success rate. We find that, compared to images, it can be more challenging to achieve the two goals on time series. This is because time series have fewer input dimensions and lower degrees of freedom, making it hard to achieve a high attack success rate without compromising stealthiness. Our generative approach addresses this challenge by generating trigger patterns that are as realistic as real-time series patterns while achieving a high attack success rate without causing a significant drop in clean accuracy. We also show that our proposed attack is resistant to potential backdoor defenses. Furthermore, we propose a novel universal generator that can poison any type of time series with a single generator that allows universal attacks without the need to fine-tune the generative model for new time series datasets.**

I. Introduction

Time series captures a sequence of observations with measurable quantities indexed by timestamps. It is amongst the most ubiquitous data types in a wide range of industries, such as finance [1], heavy industry [2], and healthcare [3], [4]. Similar to the computer vision field, deep neural networks (DNNs) are often used in time series analysis to achieve the state-of-the-art performance [5], [6], [7]. However, DNNs are known to be vulnerable to backdoor attacks where the adversary aims to control the model's test-time prediction behaviors by implanting a backdoor trigger into the model at training time [8], [9]. This has raised security concerns with the deployment of DNN models in safety-critical applications.

Backdoor attacks represent one type of training-time vulnerabilities of DNNs, which are different to test-time adversarial attacks [10], [11], [12]. To implant the backdoor trigger into a target model, the adversary can either poison a small fraction of the training data with a trigger pattern [8] or directly manipulate the training procedure [9]. The former could occur during the data collection process, while the latter could happen to outsourced model training or the use of pre-trained models downloaded from untrusted sources. A backdoored model predicts the correct classes on clean test inputs yet will constantly predict the backdoor class whenever the trigger pattern appears.

Backdoor attacks have been extensively studied on images with DNN-based image classifiers, however, few works have investigated the potential backdoor vulnerability of DNN-based time series models. The two main objectives of backdoor attacks, namely high attack success rate (ASR) and high stealthiness, have been achieved on images as demonstrated by many existing works [8], [13], [14], [9], [15]. The effectiveness of backdoor attacks is closely associated with their trigger patterns, which are often designed to be fixed patterns or dynamic but sparsely distributed patterns for images. However, unlike images, time series are generally of lower dimensions (e.g., univariate) and fewer degrees of freedom (e.g., limited window length). It thus makes fixed patterns more noticeable (less stealthy) on time series. In fact, it is still unclear whether fixed patterns are effective on time series. Moreover, time series are of diverse types, such as stock prices, temperature readings, weather data, and heart rate monitoring, to name a few. As such, fixed patterns can hardly be effective on all types of time series.

In this paper, we present a novel generative approach for generating stealthy and sample-specific trigger patterns for effective time series backdoor attacks. By leveraging generative adversarial networks (GANs), our approach can generate backdoored time series (the original time series plus the trigger pattern) that are as realistic as real time series, while achieving a high attack success rate. Furthermore, by training the trigger pattern generator on multiple types of time series, we can obtain a universal generator. The universal trigger generator demonstrates high flexibility in performing backdoor attacks on different types of time series across different domains, revealing the significant threat of backdoor attacks to time series analysis. Our work provides a practical solution to stealthy and effective time series backdoor attacks, and reveals the potential backdoor vulnerability of DNN-based time series classification models.

In summary, our main contributions are:

- We study the problem of backdoor attacks on time series and propose a novel generative approach for crafting stealthy sample-specific backdoor trigger patterns. We also reveal the unique challenge of time series backdoor

attacks posed by the inherent properties of time series, (*i.e.,* low dimension and limited degrees of freedom).

- We empirically show that our proposed attack can generate stealthy and effective backdoor attacks against state-of-the-art DNN-based time series models and is resistant to potential backdoor defenses. The attacked models also have minimal clean accuracy drop on both univariate and multivariate datasets.
- We present a novel universal backdoor attack that is capable of crafting sample-specific backdoor triggers for different types of time series across a wide range of domains. With a one-time training on a combination of time series datasets, the proposed universal attack can succeed 70% of the time under the poison-label setting.

II. Related Work

In this section, we briefly review existing works in backdoor attack and defense that are most relevant to our work.

A. Backdoor Attack

A backdoor attack implants a backdoor trigger into the victim model by injecting the trigger pattern into a small proportion of the training data. It preserves the model's performance on benign (clean) inputs, but can manipulate the model to constantly predicts the attacker-specified backdoor class whenever the trigger pattern appears in a test input.

1) BadNets: BadNets [8] is the first backdoor attack that was designed for image classification models. With an attacker-specified backdoor label y_t, BadNets first stamps a pre-designed backdoor trigger onto a benign image $\boldsymbol{x}$ to generate a poisoned sample $\boldsymbol{x'}$ and changes its ground-truth label to y_t. It then trains a backdoored model on the poisoned training data of which a small portion of the samples are poisoned following the above procedure. At inference time, the attacked model performs well on benign test samples, yet consistently predicts the backdoor label y_t for any test samples with the trigger pattern attached. Using a simple checkerboard pattern, BadNets can achieve an attack success rate (*i.e.,* the ratio of poisoned test samples that are predicted as the backdoor class) of 99% on MNIST dataset by poisoning only 10% of the training data. In Section IV-A, we will propose a simple BadNets-equivalent baseline attack for time series.

2) Invisible patterns: Following BadNets, a number of backdoor attacks have been proposed in computer vision applications with images. [13] first discussed the stealthiness of backdoor attacks in regard to the invisibility requirement of trigger patterns. They suggested that poisoned images should be indistinguishable from their benign counter-part to evade human inspection. Accordingly, they proposed a blending strategy that generates poisoned images by blending the backdoor trigger with benign images, instead of direct stamping. A small-amplitude random noise is then added to further reduce the risk of being detected. After [13], a series of works have been proposed to generate invisible trigger patterns, which include [9], [14], [15], [16]. All these works were proposed for images. In [17], a video backdoor attack was proposed against video recognition models. It leverages universal adversarial perturbations to tackle the higher dimension challenge posed by videos. Backdoor attacks have also been extended to other vision tasks such as crowd counting [18] and visual object tracking [19]. In this work, we will also limit our patterns to be invisible, but to address the lower dimension challenge posed by time series.

3) Sample-specific patterns: The above-mentioned backdoor attacks all use a fixed pattern at a fixed location of the image as the trigger pattern, which could potentially be defended and removed easily. To address this problem, [20] proposed an input-aware backdoor attack to generate different backdoor trigger patterns for different input samples, enforcing each trigger pattern to be only functional for one particular input sample. Similarly, [21] proposed LIRA that can generate the optimal trigger while successfully poisoning the classifier in computer vision applications. Most recently, inspired by DNN-based image steganography, [22] proposed another sample-specific backdoor attack via encoding an attacker-specified string into benign images, which generates sample-specific additive noises as backdoor triggers. For stealthiness, in this paper we are also interested in the sample-specific backdoor attacks.

4) Time series attacks: [23] studied backdoor attacks against transfer learning on both image and time series data. Particularly, they proposed to manipulate the pre-trained teacher models to generate customized student models that make the wrong predictions. However, the times series (e.g. ECG signals) were transformed into 1D or 2D images and attacked based on image backdoor attacks. [24] proposed TrojanFlow to perform backdoor attacks against network traffic classification models. By poisoning the traffic flow in a stealthy manner, this attack achieves high ASRs on both flow-based and payload-based classifiers. More recently, [25] proposed a backdoor attack TimeTrojan on time series classifiers. TimeTrojan proposes a multi-objective optimization framework to help the model learn a strong correlation between the trigger pattern and the target class.

There are also backdoor attacks that are not realised via data poisoning but direct modifications of the model parameters [26], [27] or structures [28], [29], [30]. These attacks are independent of poisoning-based attacks and can be performed even after a poisoning-based attack. In this paper, we will focus on data poisoning-based attacks and leave other types of manipulations to future work. Particularly, we propose a generative approach for time series backdoor attack that produces both realistic and effective sample-wise backdoor triggers.

B. Backdoor Defense

A large number of defense methods have been proposed to mitigate the backdoor threat via detection or purification. Neural Cleanse [31] presented the first solution to detect poisoned models. It first computes potential trigger patterns for each class that could convert any clean image to that class. Then, it detects among these candidates and selects the

abnormally smaller ones as backdoor indicators. Following [31], improved detection techniques were introduced in [32], [33]. Fine-Pruning [34] purifies a backdoored model by eliminating neurons that are dormant on clean inputs. Knowledge distillation (KD) [35] techniques were also leveraged in [36] and [37] to remove backdoors from infected DNNs. However, applying Fine-Pruning and KD could degrade the clean accuracy when only limited clean data are available [38]. [39] proposed to remove neurons with high activation values from the final convolutional layer. More recently, a robust training strategy was proposed in [40] to train backdoor-free models on poisoned dataset. [41] introduces an effective backdoor removal method Adversarial Neuron Pruning (ANP) to prune adversarially sensitive neurons to purify the model.

In this paper, we present a novel generate time series backdoor attack that can produce backdoor samples that are not detectable by neither human inspectors nor strong backdoor defenses.

III. Time Series Backdoor Attack

In this section, we introduce our proposed *Time Series Backdoor Attack (TSBA)* in the context of time series classification (TSC). We first define our threat model and overview the attack pipeline, then introduce the details of the proposed trigger generator and its training procedure. Finally, we introduce how to train a universal trigger generator to craft sample-specific backdoor triggers for any type of time series.

A. Problem Formulation

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ denote the benign training set of N *i.i.d.* samples, where each $\boldsymbol{x}_i$ represents one time series sample and y_i is its corresponding ground-truth label. A classification model f learns the function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that maps the input space to the label space. This is typically done by minimizing the model's classification error on $\mathcal{D}$ as follows:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},y)\in\mathcal{D}} \mathcal{L}_{CE}(f(\boldsymbol{x}), y), \tag{1}$$

where $\mathcal{L}_{CE}$ is the commonly used cross-entropy loss and θ are the model parameters.

A backdoor adversary poisons the training data $\mathcal{D}$ into a poisoned dataset $\mathcal{D}'$ with a trigger pattern $\boldsymbol{p}$ such that the model trained on $\mathcal{D}'$ will become a backdoored model f'. With the trigger pattern $\boldsymbol{p}$, a poisoned sample can be crafted by $\boldsymbol{x}' = \boldsymbol{x} \odot \boldsymbol{p}$ where $\odot$ represent any stamping method, such as addition, subtraction or replacement. The adversary aims to achieve two goals with the backdoored model f'. First, the model should predict the correct label for benign inputs, *i.e.,* $f'(\boldsymbol{x}) = y$ for any test input $\boldsymbol{x} \in \mathcal{D}_{test}$. Second, the model should predict the backdoor class y_t for any stamped test input with the trigger pattern, *i.e.,* $f'(\boldsymbol{x}') = y_t$ for any backdoored test input $\boldsymbol{x}'$.

B. Threat Model

Existing threat models from the adversary's perspective can be categorized into three main categories: 1) *training manipulation* where the adversary not only poisons the training data but also controls the training procedure [15], [20], [42]; 2) *data poisoning* where the adversary can only poison the training data with the trigger pattern [8], [13]; and 3) *post-training injection* where the adversary does not poison the training data nor control the training procedure, but directly modifies the parameters of a cleanly-trained model [43]. Unlike most image backdoor attacks that only consider one of the above three threat models, in this paper we design and evaluate our *Time Series Backdoor Attack (TSBA)* under two different threat models that fall into category 1) and 2) stated above, denoted as *TSBA-A* and *TSBA-B*:

***TSBA-A**:* This threat model simulates the real-world scenarios where the victim users directly download and deploy pre-trained DNNs models from untrusted sources. Under this threat model, the adversary has full access to the training data $\mathcal{D}$ with complete control over the training procedure. Accordingly, the adversary purposely trains a backdoored model f' with the poisoned training data $\mathcal{D}'$ which contains a small portion of poisoned samples by sample-wise trigger pattern $\boldsymbol{p}$. After downloading the backdoored model f', the victim user is expected to inspect the model's performance on its own validation data D_{val} which is unknown to the adversary.

***TSBA-B**:* This threat model assumes the victim users have full control over the training procedure but accidentally collected a few poisoned samples into its training set. The adversary may have knowledge of the network architecture adopted by the victim, and can poison a small number of training samples. The adversary can thus add sample-wise trigger patterns to a few samples and then injects the samples into the training set. The victim trains a backdoored model f' on the poisoned dataset $\mathcal{D}'$. Meanwhile, the victim may inspect the poisoned dataset $\mathcal{D}'$ via either visual inspection or certain backdoor defense techniques to remove the backdoored samples.

C. Proposed Attack Pipeline

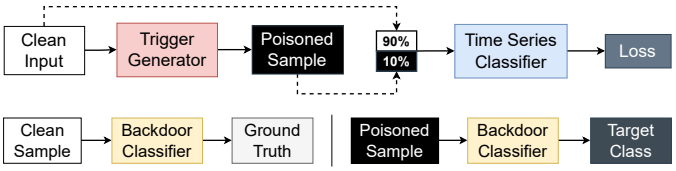

Fig. 1: Overview of the proposed TSBA attack. *Top*: training the TSBA trigger pattern generator; *Bottom*: inference with the backdoored model on clean vs poisoned samples.

As illustrated in Figure 1, our proposed TSBA attack trains a trigger generator network to generate sample-specific backdoor trigger patterns for poisoned samples. The adversary randomly selects 10% of training data for backdoor poisoning, and trains a time series classifier on both the poisoned and clean training samples. To achieve both high attack success rate and high clean accuracy, the two components of TSBA, namely the trigger generator and the classifier, are trained iteratively via

specific procedures described in Section III-D. At inference time, the backdoored classifier will predict the adversary-specified class (backdoor class y_t) for poisoned samples, while recognizing correct classes for clean samples.

Under the *TSBA-A* threat model, the adversary directly releases the backdoored classifier to the victim user, while privately keeping the trained trigger generator to perform backdoor attacks at inference time, *i.e.,* as our TSBA is a sample-specific attack, the adversary will need the trigger generator to produce backdoor samples at inference time.

Under the *TSBA-B* threat model, the adversary will leverage the trigger generator to perform data poisoning, *i.e.,* poisoning a small proportion (e.g. 10%) of the training samples using the trigger generator. The victim user then trains a classifier on the poisoned training set following a typical model training procedure. The victim user will use the trained classifier to perform inference on either clean or backdoored test samples. The inference procedure is the same as in *TSBA-A*.

Algorithm 1: Training Procedure of TSBA

1 Let f be the classifier, g be the trigger generator
2 Let $\text{clip}_\xi(\boldsymbol{x} \odot \boldsymbol{k})$ limit $\boldsymbol{x}$ bounded by $[\boldsymbol{k} - \xi, \boldsymbol{k} + \xi]$
3 Let $G_\xi(\boldsymbol{x}) = \text{clip}_\xi(g(\boldsymbol{x}) \odot \boldsymbol{x})$
4 Given a target class y_t, a training dataset $\mathcal{D}$, the backdoor poison rate γ
5 T_{clean}: clean training epochs; $T_{backdoor}$: backdoor training epochs
6 Initialize f, g
7 # warm start f
8 **for** i *in* $range(T_{clean})$ **do**
9 **for** $(\boldsymbol{x}, y)$ *in* $\mathcal{D}$ **do**
10 $f \leftarrow \arg\min_f \mathcal{L}_{CE}(f(\boldsymbol{x}), y)$
11 **end**
12 **end**
13 # simultaneous training of g and f
14 $\mathcal{D}' \leftarrow \text{random_sample}_\gamma(\mathcal{D})$
15 **for** i *in* $range(T_{backdoor})$ **do**
16 **for** $(\boldsymbol{x}', y_t)$ *in* $\mathcal{D}'$ **do**
17 $g \leftarrow \arg\min_g \mathcal{L}_{CE}(f(G_\xi(\boldsymbol{x}')), y_t)$
18 **end**
19 $\mathcal{D}' \leftarrow \mathcal{D} \cup \{G_\xi(\boldsymbol{x}), y_t\}$
20 **for** $(\boldsymbol{x}', y')$ *in* $\mathcal{D}'$ **do**
21 $f \leftarrow \arg\min_f \mathcal{L}_{CE}(f(\boldsymbol{x}'), y')$
22 **end**
23 **end**
24 **return** f, g

The core component of TSBA is the trigger generator which uses a simple DNN architecture suited for both univariate and multivariate time series. The detailed model structure is shown in Table VII. The trigger generator takes a time series sample as an input and generate a sample-specific trigger pattern for the sample. Test the trigger pattern is of the same size as the original sample. Then, the generated pattern is added to the original sample to craft a poisoned sample. Given that the trigger patterns are dynamic (sample-specific) and stealthy (ensured by the generator), it is difficult for the victim user to identify which samples are backdoored [20], [22]. Moreover, even if the victim user has detected the trigger patterns for several samples, they could not remove the trigger pattern for other samples without the trigger generator. This is in sharp contrast to fixed-pattern based backdoor attacks, where the fixed pattern can be easily removed once detected.

D. Training the Trigger Generator

The training procedure of the trigger generator is described in Algorithm 1. which iteratively trains the trigger generator g and the backdoored classifier f to achieve both high attack success rate and high clean accuracy.

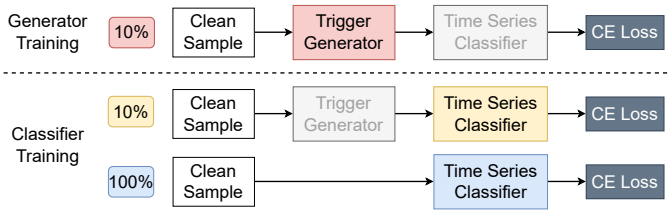

Fig. 2: The training procedure of the trigger generator g (top) and the backdoor classifier f (bottom)

To address the cold start problem of generator training, we first pre-train the time series classifier f for T_{clean} epochs on all clean samples from $\mathcal{D}$ until it has a steady drop in the cross-entropy loss $\mathcal{L}_{CE}$. This corresponds to line 7-11 in Algorithm 1. After this pre-training, we train the trigger generator g and the partially trained classifier f simultaneously for $T_{backdoor}$ epochs. Both networks are progressively updated in each iteration following a similar process: 1) training g on the poisoned samples (initialized to be randomly sampled clean samples at line 13) to minimize the classification loss with respect to the backdoor class y_t (line 15-17); 2) generating poisoned dataset $\mathcal{D}'$ using g (line 18); and 3) training f on the poisoned dataset $\mathcal{D}'$ with the poisoned samples are relabeled to y_t (line 19-21). Note that during the entire process, the backdoor trigger pattern is clipped to be within 10% of the signal amplitude, *i.e.,* $0.1 * (\boldsymbol{x}_{max} - \boldsymbol{x}_{min})$, to strengthen stealthiness (line 2). This process is further illustrated in Figure 2. This training procedure encourages the trigger generator g to explore the most effective patterns that can alter f's predictions towards the target class y_t.

Unlike the image backdoor training where the backdoor samples are generated before model training, we refresh the sample-specific trigger for each backdoor sample using the updated trigger generator g. Likewise, the time series classifier f is backdoor trained to minimize the CE loss on the partially-poisoned training data. Thus, it enables the classifier to recognize the backdoor pattern induced by the trigger generator, while maintaining clean accuracy with clean training samples.

How TSBA works? In TSBA, the generator is designed to progressively generate stronger (and more realistic) trigger

patterns, while the classifier simultaneously learns the correlation between the trigger patterns and the target class. This design is motivated by our observation in Section IV-D that simple trigger patterns cannot be easily learned into the model. I.e., establishing the backdoor correlation is relatively hard in time series models. The clean signals tend to overwhelm the backdoor noise during the training process. As such, the generator and the classifier need to learn together to explore stronger triggers. Beyond time series backdoors, our design could also be useful for scenarios where backdoor triggers are generally hard to inject.

E. Training a Universal Trigger Generator

The above generator needs to be re-trained or fine-tuned for a new type of time series. Here, we further train a universal trigger generator to generate sample-specific triggers for any type of time series without the need to fine-tune the generator for unseen datasets. The universal generator has a similar architecture as the dataset-specific trigger generator used in the above *TSBA* algorithm, but with one additional convolutional layer and revised parameters to provide more generalization capacities for multiple time series datasets. The detailed model architecture can be found in Table VIII.

Algorithm 2: Universal Trigger Generator Training

Let g be the universal trigger generator
Given a set of distinct time series datasets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n\}$
Let $\mathcal{C} = \{1, 2, ..., n\}$ be a class set
Initialize g
for i *in* $range(T_u)$ **do**
  $\mathcal{S}' \leftarrow$ random_sample($\mathcal{S}$)
  $y_t \leftarrow$ random_select($\mathcal{C}$)
  $g \leftarrow$ TSBA($g, \mathcal{S}', y_t$) ▷ *following* **Algorithm 1**
end
return g

As described in Algorithm 2, it generally follows the training procedure of *TSBA* with several extra steps. The generator is trained with samples from a combination of time series datasets which are merged with all class labels re-organized. In each iteration, the generator is refreshed with randomly selected training samples and target class. Accordingly, the generator is optimized to create the trigger pattern according to the style of the time series input to perturb samples from unseen datasets. Thus, it simplifies and automates the procedure of trigger pattern selection by removing the necessity for manual selection or training a dataset-specific trigger generator. More detailed training setup can be found in Section IV.

IV. Experiments

In this section, we evaluate the performance of our proposed TSBA and the universal trigger generator. We first introduce three simple baseline attacks and describe our experimental setup. We then discuss the experimental results and show the differences between image vs. time series backdoor attacks. Following that, we demonstrate the stealthiness of the backdoored samples generated by TSBA, as well as its resistance to backdoor defenses.

A. Simple Baseline Attacks

In this work, we also propose three image-equivalent methods of time series backdoor attack. Due to the lack of prior research on time series backdoor attack, we use them as the baseline attacks. The attacked waveforms by the 3 baseline attacks are illustrated in Figure 3. The first two baselines are the two versions of a *vanilla backdoor* attack: 1) adding fixed patterns at the beginning of the time series; and 2) randomly covering one of the peaks (or troughs) in the time series. Since time series sample does not have a limit in their value range, we alter 5% of the time series randomly and alternatively to its maximum and minimum signal value.

The second baseline attack, named as *static noise*, uses static powerline noise as the trigger pattern. This idea was motivated by the observation in recent research that powerline noises caused by the signal capture device commonly exist in Electrocardiogram (ECG) signals [44], [45]. The noise is tiled (repeated) to the same length as the original time series and applied to all time-dependent dimensions. Also, it is standardized with its size is set to be 10% of the amplitude ($\boldsymbol{x}_{max} - \boldsymbol{x}_{min}$), which is of the same amplitude as the patterns generated by our TSBA.

B. Experimental Setup

We evaluate our TSBA under both threat models *TSBA-A* and *TSBA-B* (introduced in Section III-B) on 13 time series datasets, where 8 of them are univariate datasets from the UCR Archive, and the rest 5 are multivariate datasets from the MTS Archive. Statistics of the 13 datasets can be found in Appendix A. Due to the small number of instances and the non-typical train-test split for most datasets, we apply 10-fold cross-validation for all of our experiments and report the average clean accuracy (CA) and attack success rate (ASR).

The poisoning rate for all attacks is set to be 10%. For better stealthiness, the backdoor trigger pattern is clipped to be 10% of the signal amplitude ($\boldsymbol{x}_{max} - \boldsymbol{x}_{min}$) for each sample. For *TSBA-A* and the three baseline methods, we directly take the backdoored classifier as the final classifier. For *TSBA-B*, the final classifier is trained with the 10%-poisoned training dataset by the trained trigger pattern generator. We select the first class of each dataset as the backdoor class. Details of all network architectures can be found in Appendix A. We first warm start the time series classifier by training it for 20 epochs with purely clean samples. Then, we train each model for 500 epochs and apply early stopping to avoid overfitting. The training is terminated when the backdoored classifier achieves an optimal ASR and CA on the validation set according to the 10-fold cross validation.

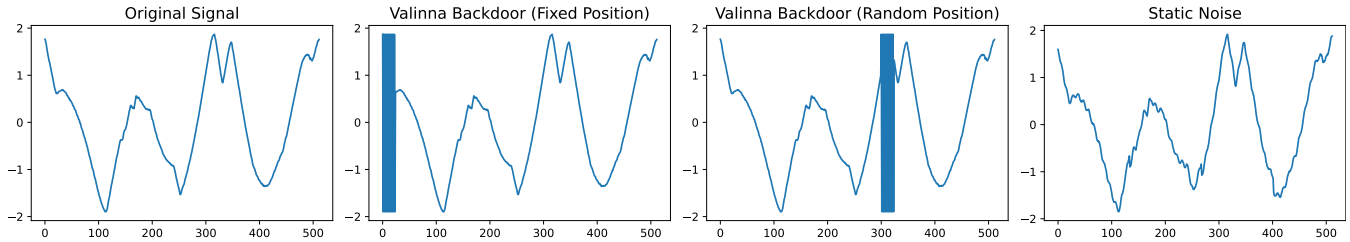

Fig. 3: Waveforms of the original and attacked signals by 3 baseline attacks. The clean signal is taken from (D_1) BirdChicken.

TABLE I: Experiment results of poison-label backdoor attacks on 13 datasets in terms of clean accuracy (**CA**) and attack success rate (**ASR**). Both *TSBA-A* and *TSBA-B* threat models are tested. D_1 to D_8 are univariate datasets from the UCR Archive, while D_9 to D_{13} are multivariate datasets from the MTS Archive. The accuracy of clean model (without backdoor training) is provided in the 3rd column (**Clean**).

Dataset	**Classifier**	**Clean**	Vanilla (Fixed)		Vanilla (Random)		Static Noise		**TSBA-A (Ours)**		**TSBA-B (Ours)**	
			CA	**ASR**	**CA**	**ASR**	**CA**	**ASR**	**CA**	**ASR**	**CA**	**ASR**
(D_1) BirdChicken	TCN	96.0%	88.0%	52.0%	86.0%	56.0%	90.0%	79.0%	94.0%	**100.0%**	92.0%	94.0%
	ResNet	92.0%	86.0%	48.0%	83.0%	49.0%	85.0%	80.0%	90.0%	**100.0%**	86.0%	94.0%
	LSTM	94.0%	85.0%	49.0%	83.0%	48.0%	88.0%	72.0%	91.0%	**100.0%**	88.0%	92.0%
(D_2) ECG5000	TCN	94.6%	88.2%	56.1%	85.0%	55.9%	89.6%	76.2%	92.7%	**100.0%**	89.5%	89.0%
	ResNet	93.9%	88.9%	55.6%	85.2%	55.8%	89.8%	75.9%	92.3%	**100.0%**	89.0%	86.2%
	LSTM	94.1%	86.9%	55.2%	85.1%	54.7%	88.5%	74.7%	92.0%	**99.5%**	89.1%	84.1%
(D_3) Earthquakes	TCN	72.5%	66.9%	51.7%	63.9%	55.8%	66.6%	72.1%	71.4%	**100.0%**	66.9%	81.1%
	ResNet	71.7%	66.9%	53.9%	65.8%	54.0%	66.5%	74.6%	70.3%	**99.7%**	66.3%	78.4%
	LSTM	72.2%	64.6%	53.3%	64.8%	56.9%	68.1%	70.4%	68.8%	**100.0%**	66.0%	76.8%
(D_4) ElectricDevices	TCN	72.3%	65.9%	47.1%	63.6%	49.7%	66.9%	76.1%	70.7%	**100.0%**	67.1%	82.5%
	ResNet	73.4%	66.3%	47.8%	66.4%	48.9%	68.3%	75.8%	71.8%	**99.8%**	68.6%	81.6%
	LSTM	70.8%	64.9%	46.5%	63.7%	44.2%	64.8%	71.9%	68.9%	**99.4%**	65.9%	81.9%
(D_5) Haptics	TCN	50.2%	46.4%	44.5%	45.0%	49.1%	46.1%	69.0%	49.3%	**99.0%**	46.5%	86.7%
	ResNet	51.6%	48.5%	45.7%	47.2%	46.7%	47.7%	66.8%	50.8%	**99.7%**	48.4%	88.2%
	LSTM	54.0%	48.1%	46.3%	47.0%	48.6%	48.9%	67.2%	52.8%	**99.6%**	49.6%	88.6%
(D_6) PowerCons	TCN	88.2%	79.7%	59.1%	76.1%	62.0%	80.6%	76.4%	84.6%	**100.0%**	81.7%	79.1%
	ResNet	89.7%	82.6%	58.2%	79.1%	61.2%	82.9%	75.2%	85.7%	**100.0%**	84.3%	78.2%
	LSTM	86.4%	76.8%	56.7%	72.7%	61.7%	78.2%	73.9%	81.3%	**100.0%**	79.3%	75.6%
(D_7) ShapeletSim	TCN	72.4%	64.0%	56.9%	61.3%	58.2%	64.3%	77.9%	69.0%	**100.0%**	66.5%	84.7%
	ResNet	77.9%	70.8%	57.0%	65.2%	56.7%	69.9%	79.4%	73.8%	**100.0%**	71.5%	85.2%
	LSTM	68.5%	62.1%	55.3%	55.3%	56.9%	61.8%	76.2%	63.7%	**99.2%**	60.4%	84.4%
(D_8) Wine	TCN	59.6%	55.5%	52.6%	52.5%	54.1%	56.2%	74.0%	57.5%	**98.4%**	56.5%	80.7%
	ResNet	74.5%	66.1%	50.1%	66.3%	52.7%	71.3%	76.2%	71.6%	**96.9%**	71.9%	76.9%
	LSTM	66.8%	59.9%	48.5%	58.9%	53.4%	64.0%	73.7%	64.4%	**95.8%**	64.1%	78.2%
(D_9) ArabicDigits	TCN	99.4%	92.8%	52.4%	90.6%	56.8%	93.5%	68.7%	97.1%	**96.1%**	94.2%	77.4%
	ResNet	99.6%	92.5%	56.7%	90.2%	58.4%	94.2%	65.2%	96.6%	**98.2%**	94.2%	78.9%
	LSTM	94.2%	89.2%	55.3%	86.3%	56.1%	88.1%	64.9%	91.8%	**94.5%**	88.2%	75.0%
(D_{10}) ECG	TCN	87.4%	82.5%	59.1%	80.5%	59.0%	82.3%	79.2%	85.5%	**100.0%**	82.8%	81.6%
	ResNet	86.2%	82.0%	62.4%	79.0%	61.2%	81.0%	80.1%	84.8%	**100.0%**	81.5%	83.7%
	LSTM	86.8%	83.7%	58.6%	82.0%	60.5%	81.4%	77.4%	84.7%	**100.0%**	83.0%	80.5%
(D_{11}) KickvsPunch	TCN	54.0%	51.1%	55.1%	48.8%	55.4%	50.9%	74.1%	53.2%	**98.4%**	51.0%	77.1%
	ResNet	51.3%	47.9%	54.3%	46.4%	56.9%	48.5%	75.9%	50.4%	**97.1%**	48.7%	75.8%
	LSTM	51.1%	49.3%	52.7%	47.5%	53.1%	49.1%	76.0%	50.0%	**97.6%**	48.8%	75.4%
(D_{12}) NetFlow	TCN	89.4%	83.7%	48.9%	79.7%	49.6%	83.7%	79.4%	86.3%	**100.0%**	84.5%	82.4%
	ResNet	77.5%	72.6%	51.1%	69.4%	50.8%	72.3%	80.6%	75.1%	**100.0%**	72.9%	84.5%
	LSTM	88.6%	82.2%	50.4%	80.4%	50.1%	82.1%	78.5%	85.4%	**100.0%**	83.4%	84.9%
(D_{13}) UWave	TCN	93.4%	87.0%	43.7%	83.2%	46.7%	87.5%	66.7%	90.4%	**98.0%**	87.3%	72.5%
	ResNet	92.2%	85.5%	46.2%	80.8%	48.9%	86.0%	68.5%	88.8%	**99.1%**	86.5%	76.7%
	LSTM	84.1%	78.9%	42.5%	74.3%	46.5%	79.0%	64.3%	81.3%	**96.4%**	78.7%	69.4%

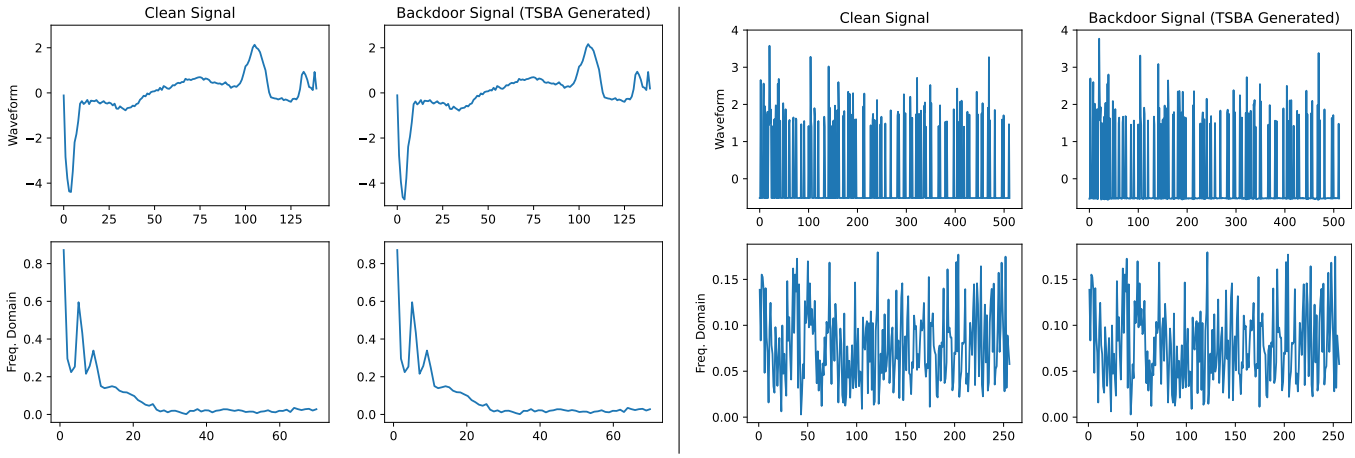

Fig. 4: Example waveform and frequency domain of the original and backdoor time series. The two examples are taken from (D_2) ECG5000 dataset and (D_3) Earthquake dataset, respectively.

C. Main Results

The attack performance of different attacking methods is reported in Table I. It is clear that our proposed TSBA-A and TSBA-B attacks achieved the best attack performance among all the baseline attacks, in terms of both ASR and CA. Specifically, *TSBA-A* achieves 100% ASR in 19 out of the 39 experiments, while an average ASR of 98.2% in the rest of the 20 experiments. The average clean accuracy drop is around 2.45% and 2.25% for univariate and multivariate datasets, respectively. For *TSBA-B*, it is less powerful than *TSBA-A* but can still achieve an average ASR of 81.6%. It causes an average clean accuracy drop of 5.09% and 4.63% for univariate and multivariate datasets, respectively. Surprisingly, as a weaker attack, the CA performance of *TSBA-B* is no better than *TSBA-A*. We suspect this is because *TSBA-B* has zero control over the training procedure and does not have the benign counterparts of the poisoned samples in the training set. Compared with static noise, our *TSBA-B* achieves significantly higher ASR in all experiments while maintaining similar clean accuracy in most cases. These results confirm the attack effectiveness of our proposed TSBA attacks.

D. Time Series vs. Images

As shown in Table I, directly applying image-equivalent backdoor attacks (*i.e.,* the two variants of vanilla backdoor) on time series data fails to deliver strong attack performance as in the image domain. This is because time series are generally of lower input dimensions and fewer degrees of freedom than images, making fixed trigger patterns less effective. The fast and steep shifts induced by the vanilla backdoor patterns may not be sensitive to the classifier and could be suppressed by the clean peak or trough signals. Even for the variant that randomly applies the trigger pattern to cover one or more peaks (or troughs), it could only improve the ASR by an average of 1.63% while reducing the clean accuracy by an average of 2.53%. In fact, effective trigger patterns should produce smooth transitions across the entire signal. By simply adding the static powerline noise with a small 10% (standardized) amplitude, it achieves significantly higher ASRs and CAs compared with the vanilla backdoor. Our TSBA attacks present a more advanced version of the static noise that can generate more smooth and persisting trigger patterns, and more importantly, being adaptive to each input time series.

E. Stealthiness Analysis

We use the root mean square (RMS) of the trigger patterns to measure their stealthiness. We apply the trained trigger generator to create sample-specific trigger patterns for all samples in each of the 13 datasets, then report their mean RMS values within each dataset as well as the average over all 13 datasets. To simplify the results, this experiment was conducted under the TSBA-A threat model based on the FCN, one most commonly used model architecture for time series classification. The results are reported in Table II.

TABLE II: The RMS of the generated trigger patterns by TSBA. All RMS values are normalized with respect to the magnitude of each sample. $\text{RMS}_{\text{Top 1\%}}$ only computes the highest 1% portion sorted by the absolute signal values.

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
RMS_{All}	0.014	0.011	0.008	0.017	0.021	0.016	0.012
$\text{RMS}_{\text{Top 1\%}}$	0.056	0.044	0.061	0.072	0.049	0.064	0.039
	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	**Avg.**
RMS_{All}	0.029	0.031	0.017	0.032	0.024	0.036	**0.021**
$\text{RMS}_{\text{Top 1\%}}$	0.052	0.087	0.068	0.047	0.059	0.084	**0.060**

It is evident that our TSBA attack introduces very small variations to the original signal, only incurring an average of 2.1% of the amplitude for all samples across the 13 datasets, although we allow up to 10% perturbation of each sample's amplitude. It is worth mentioning that the static noise incurs an average of 10% of the amplitude, which is much higher than TSBA. The low top 1% RMS values indicate that only a small

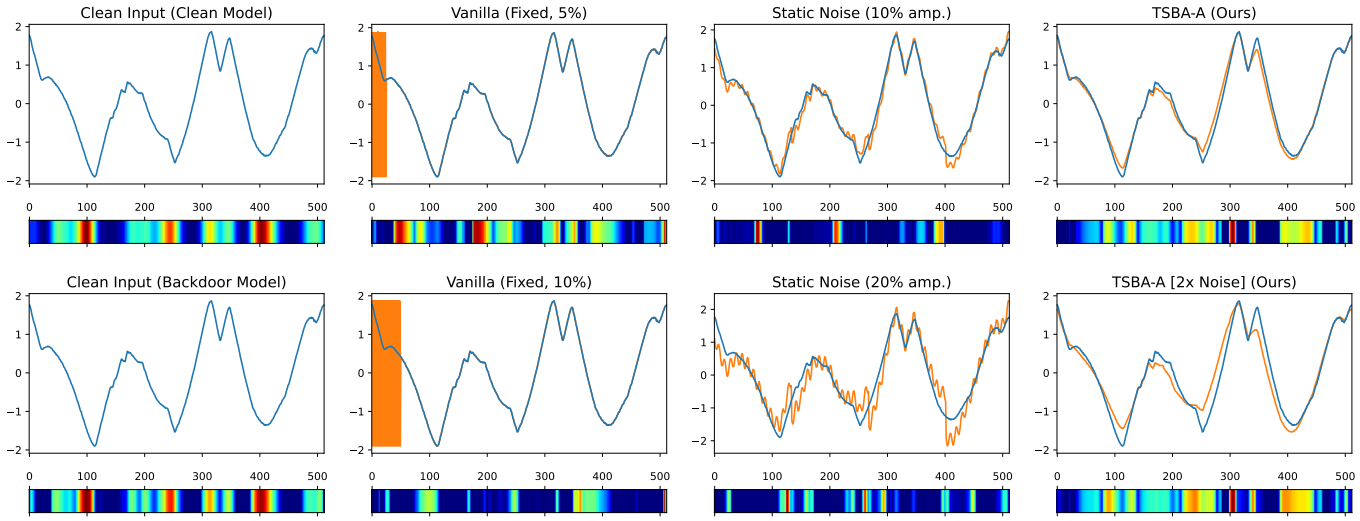

Fig. 5: Waveforms and Grad-CAM attention maps under different attack settings. The example is taken from (D_1) BirdChicken dataset. The original (blue) and backdoor (orange) signals are plotted together as a comparison.

portion of each sample is heavily perturbed, while the rest are insignificantly modified. However, such small but sample-specific perturbations can achieve remarkably high ASR with similar clean accuracy.

Figure 4 shows that the backdoor samples and their benign counterparts are imperceptibly different to their clean counterparts in both the time domain (*i.e.,* waveform) and frequency domain (obtained via Fourier transformation). Figure 5 shows the different trigger patterns crafted by different attacks. The trigger patterns generated by our attack are very smooth and visually more natural compared to other baselines. Note that the vanilla backdoor attack creates a high variation block at the beginning of the signal, while the static noise causes a suspicious sawtooth effect across the entire signal. This confirms the high stealthiness of TSBA and the benefit of our generative approach. Our proposed TSBA attacks are capable of generating stealthy backdoor triggers owing to its generative design.

F. Grad-CAM Visualization

To help understand the working mechanism of the trigger patterns generated by TSBA, we visualize the Grad-CAM [46] attention map in Figure 5 for the trigger patterns crafted by different attacks and their boosted versions. The boosted triggers can help visualize the differences. For the vanilla backdoor with fixed pattern position (at the beginning), we double its perturbation magnitude from 5% to 10%. For the static noise, we directly double the amplitude of the added noise pattern. For our *TSBA-A*, we retrain the model with a doubled clipping rate ξ, from 10% to 20%. Amongst all six backdoored samples (with their corresponding models), only our TSBA-A and its boosted version have successfully performed the backdoor attack. Unsurprisingly, the trigger for the vanilla backdoor does not activate the hidden backdoor as its attention map is very similar to that of the original signal. For the static noise, there is no obvious region of the altered attention map that can cause incorrect prediction. For the enhanced version of the above baseline attacks, more less-focused regions are shown in the attention map (*i.e.,* more blue regions and less yellow or red regions), indicating that those trigger sequences divert the attention of the classifier. Yet, they are not strong enough to mislead the model's prediction. By contrast, with small-amplitude triggers, our TSBA can produce more disputing regions in the classifier's attention map.

G. Resistance to Backdoor Defenses

Here, we show that our TSBA attack can easily evade state-of-the-art backdoor defense methods, including Neural Cleanse (NC) [31], Fine-Pruning (FP) [34] and Adversarial Neuron Pruning (ANP) [41]. The results of our TSBA-A/B and the three baselines are reported in Table III.

1) Evading NC: NC [31] can not only detect whether a DNN has been backdoored but also recover and mitigate the backdoor trigger via reverse engineering and unlearning. Here, we apply NC with unlearning that allows the model to decide which weights are problematic and should be updated through training. This defense can significantly eliminate the two Vanilla baseline attacks, reducing their ASR by 42%-62%. However, it fails on the other 3 attacks due to its limitation on detecting static trigger patterns that are not applied to the entire input. Since our TSBAs use sample-specific triggers, NC can only reduce their ASRs by 1%-6%.

2) Evading FP: FP [34] exploits the advantages of both pruning and fine-tuning, and progressively removes the dormant neurons that are conditioned on clean images to mitigate the backdoor. We apply FP with the default hyperparameters on the backdoored classifiers trained by our *TSBA* attacks. We set the pruning rates to 30%. Our TSBA demonstrates high resistance to the FP defense with less than 5% and 9% ASR drop for TSBA-A and TSBA-B, respectively. This implies that

TABLE III: Performance deviation of the baselines and our TSBA attacks against three backdoor defenses. All models use FCN as time series classifier. The ΔCA and ΔASR are calculated based on the results from Table I.

Dataset	**Defense**	Vanilla (Fixed)		Vanilla (Random)		Static Noise		**TSBA-A (Ours)**		**TSBA-B (Ours)**	
		ΔCA	**ΔASR**	**ΔCA**	**ΔASR**	**ΔCA**	**ΔASR**	**ΔCA**	**ΔASR**	**ΔCA**	**ΔASR**
	NC	+1.0%	-52.0%	+2.0%	-56.0%	-1.0%	-27.0%	0.0%	-1.0%	-1.0%	-3.0%
(D_1) BirdChicken	FP	-1.0%	-51.0%	-1.0%	-47.0%	-1.0%	-71.0%	-2.0%	-3.0%	-0.5%	-6.0%
	ANP	-0.0%	-52.0%	-0.5%	-56.0%	-1.0%	-76.0%	-1.0%	-6.0%	-1.0%	-7.0%
	NC	+1.5%	-56.1%	+2.1%	-55.4%	-1.4%	-32.2%	-0.4%	-1.1%	-3.7%	-5.2%
(D_2) ECG5000	FP	-1.1%	-55.2%	-0.8%	-48.9%	-1.0%	-66.4%	-2.1%	-2.9%	-1.7%	-6.1%
	ANP	-0.8%	-56.1%	-0.6%	-55.9%	-0.7%	-70.1%	-1.4%	-6.8%	-1.5%	-8.4%
	NC	+0.5%	-51.7%	-0.1%	-53.7%	+0.1%	-30.2%	-0.1%	-0.8%	-1.7%	-5.4%
(D_3) Earthquakes	FP	-0.9%	-50.4%	-1.2%	-48.1%	-2.1%	-61.9%	-2.7%	-2.6%	-2.2%	-7.7%
	ANP	-0.2%	-51.7%	-0.4%	-55.8%	-1.2%	-71.3%	-1.2%	-5.9%	-1.4%	-6.9%
	NC	+1.9%	-47.1%	+1.5%	-48.1%	-0.9%	-29.7%	-1.0%	-1.9%	-2.2%	-4.7%
(D_4) ElectricDevices	FP	-0.1%	-47.2%	-0.6%	-49.1%	-1.1%	-69.6%	-3.4%	-1.8%	-2.7%	-5.6%
	ANP	-1.4%	-47.7%	-1.0%	-49.7%	-2.4%	-74.1%	-1.9%	-6.4%	-1.7%	-8.7%
	NC	+0.1%	-44.5%	+0.4%	-48.1%	-0.3%	-33.4%	-0.6%	-1.5%	-0.8%	-3.4%
(D_5) Haptics	FP	-1.1%	-39.8%	-0.8%	-42.7%	-1.5%	-61.2%	-1.9%	-2.7%	-2.1%	-4.9%
	ANP	-1.0%	-44.0%	-1.2%	-49.0%	-1.8%	-67.1%	-0.9%	-4.9%	-1.7%	-6.4%
	NC	+1.0%	-57.9%	+0.3%	-62.0%	-1.4%	-35.9%	-0.2%	-0.4%	-0.7%	-1.5%
(D_6) PowerCons	FP	-0.6%	-51.6%	-0.8%	-53.4%	-1.7%	-67.6%	-3.0%	-2.4%	-2.7%	-3.1%
	ANP	-0.4%	-58.3%	-0.7%	-61.6%	-1.0%	-73.8%	-2.1%	-6.4%	-2.0%	-8.2%
	NC	+1.8%	-54.8%	-1.6%	-56.9%	-3.4%	-41.6%	+0.3%	-0.2%	-0.4%	-1.7%
(D_7) ShapeletSim	FP	-1.2%	-49.7%	-0.5%	-52.6%	-3.1%	-68.9%	-2.4%	-1.9%	-2.6%	-2.9%
	ANP	-0.2%	-56.3%	-0.3%	-57.3%	-1.9%	-74.5%	-1.2%	-7.1%	-1.8%	-6.8%
	NC	-0.1%	-51.3%	+0.4%	-52.0%	-0.9%	-39.9%	-0.1%	-1.1%	-1.2%	-2.0%
(D_8) Wine	FP	-1.0%	-46.7%	-0.7%	-46.9%	-1.2%	-67.5%	-1.2%	-2.6%	-2.4%	-4.9%
	ANP	-0.6%	-52.6%	-0.4%	-53.7%	-0.9%	-71.6%	-1.0%	-5.0%	-2.1%	-7.1%
	NC	+1.4%	-52.1%	+1.9%	-55.0%	-1.4%	-28.4%	-0.8%	-1.1%	-1.6%	-1.4%
(D_9) ArabicDigits	FP	-0.2%	-48.2%	-0.8%	-51.9%	-1.0%	-60.1%	-4.6%	-3.5%	-3.8%	-3.4%
	ANP	-0.7%	-52.4%	-0.4%	-56.1%	-0.9%	-65.2%	-2.4%	-7.8%	-3.7%	-8.2%
	NC	+2.2%	-58.9%	+1.5%	-56.3%	-2.1%	-29.8%	-0.9%	-2.4%	-1.1%	-3.1%
(D_{10}) ECG	FP	-0.7%	-53.6%	-0.8%	-52.1%	-1.7%	-64.2%	-3.8%	-4.1%	-3.9%	-4.7%
	ANP	-0.1%	-58.4%	-0.2%	-58.1%	-0.9%	-76.9%	-2.0%	-9.0%	-3.2%	-8.1%
	NC	+1.7%	-53.1%	+2.4%	-53.6%	-2.4%	-39.2%	-3.1%	-1.6%	-3.0%	-4.1%
(D_{11}) KickvsPunch	FP	-0.9%	-51.2%	-0.4%	-52.0%	-2.2%	-66.1%	-2.5%	-2.9%	-2.1%	-7.9%
	ANP	-0.5%	-55.1%	-0.6%	-53.9%	-1.4%	-70.7%	-1.6%	-8.6%	-2.6%	-9.1%
	NC	+0.1%	-47.6%	+0.3%	-47.6%	-2.6%	-25.2%	-0.7%	-2.9%	-1.2%	-2.4%
(D_{12}) NetFlow	FP	-0.2%	-38.3%	-0.9%	-41.8%	-2.4%	-73.5%	-2.9%	-4.3%	-2.8%	-3.8%
	ANP	-0.4%	-48.9%	-0.1%	-48.8%	-1.0%	-77.4%	-1.5%	-8.1%	-1.7%	-7.6%
	NC	+0.8%	-42.1%	+1.9%	-45.1%	-3.1%	-38.7%	-1.7%	-3.0%	-2.1%	-2.4%
(D_{13}) UWave	FP	-1.0%	-38.9%	-1.3%	-41.2%	-2.0%	-58.8%	-3.6%	-3.7%	-4.4%	-4.2%
	ANP	-0.4%	-43.7%	-0.8%	-46.4%	-1.4%	-62.3%	-2.2%	-7.7%	-3.0%	-8.1%

the trigger patterns generated by TSBAs are deeply mixed into the clean signals in the representation space, making it hard to be removed by either neuron pruning or clean-data based fine-tuning.

3) Evading ANP: ANP [41] is a recent defense method that removes potential backdoors by pruning the most susceptible neurons to adversarial perturbations. ANP demonstrates the state-of-the-art defense performance against a number of backdoor attacks. Here, we apply ANP to all backdoored models with a perturbation budget $\epsilon = 0.4$, trade-off coefficient $\alpha = 0.2$, and constant learning rate 0.2. All neuron masks are optimized using Stochastic Gradient Descent (SGD). ANP can significantly eliminate the two Vanilla attacks (i.e., Fixed and Random), reducing their ASRs to <2%. For the static noise attack, ANP brings its ASR down to less than 10% on all datasets. Our TSBA attacks are fairly robust against this strong defense with ASR drops by $< 9\%$.

The ASRs of the three baseline attacks are all below 10% under the backdoor defenses. However, the ASRs of our TSBA attacks drop only by less than 10% across all 3 defenses and 13 datasets, which are still above 80%. This proves that, even under strong backdoor defenses, our attacks can still pose high

threats. It is also worth mentioning that, since our triggers are dynamic and sample-specific, existing defense techniques designed for static patterns may not apply to our attacks.

V. Universal Trigger Generator

To evaluate the attack performance of our proposed universal trigger generator, here we train the model on a combined dataset of 10 *new* univariate datasets from UCR Archive, namely the first 11 datasets except D_1 BirdChicken (as D_1 are selected in our experiments). The training of the generator is early stopped when an optimal ASR (on the training set) is achieved. Then, we use the obtained universal trigger generator to create poisoned training set for each of the 13 datasets used in our previous experiments, with a poison rate of 10%. For multivariate datasets, we apply the universal generator to each time-dependent variable separately. Each backdoored classifier was trained following the same training procedure and setting under our *TSBA-B* threat model (stated in Section IV-B).

As shown in Table IV, the universal trigger generator exhibits strong performance on all 13 datasets. The clean accuracy is generally higher than our non-universal *TSBA-B* attack, while the attack success rate drops by less than 4%. Note that this attack performance still outperforms the static noise, the strongest baseline attack. The sample-wise trigger patterns generated by the universal generator are still dynamic and equally stealthy as those generated by *TSBA-A* and *TSBA-B*. Moreover, compared with universal adversarial noises that can be effectively detected and erased by several defense techniques, our universal trigger patterns on time series data are robust to existing image-based backdoor defenses. Our universal trigger generator provides a cheap but extremely effective approach for attacking any type of time series, posing severe threats to deep learning based time series classification.

TABLE IV: Performance of our universal trigger generator. The differences in CA and ASR are calculated versus TSBA-B.

Dataset	Classifier	Clean	CA	ASR	ΔCA	ΔASR
D_1	TCN	96.0%	94.0%	92.0%	+2.0%	-2.0%
	ResNet	92.0%	90.0%	92.0%	+4.0%	-2.0%
	LSTM	94.0%	89.0%	88.0%	+1.0%	-4.0%
D_2	TCN	94.6%	90.1%	87.5%	+0.6%	-1.5%
	ResNet	93.9%	90.2%	85.5%	+1.2%	-0.7%
	LSTM	94.1%	88.8%	82.2%	-0.3%	-1.9%
D_3	TCN	72.5%	69.9%	78.6%	+3.0%	-2.5%
	ResNet	71.7%	68.5%	75.7%	+2.2%	-2.7%
	LSTM	72.2%	67.7%	73.2%	+1.7%	-3.6%
D_4	TCN	72.3%	69.2%	80.6%	+2.1%	-1.9%
	ResNet	73.4%	71.0%	78.5%	+2.4%	-3.1%
	LSTM	70.8%	68.9%	79.5%	+3.0%	-2.4%
D_5	TCN	50.2%	50.6%	86.1%	+4.1%	-0.6%
	ResNet	51.6%	50.6%	87.1%	+2.2%	-1.1%
	LSTM	54.0%	51.0%	84.9%	+1.4%	-3.7%
D_6	TCN	88.2%	82.3%	79.3%	+0.6%	+0.2%
	ResNet	89.7%	84.1%	76.4%	-0.2%	-1.8%
	LSTM	86.4%	80.4%	75.0%	+1.1%	-0.6%
D_7	TCN	72.4%	67.8%	83.6%	+1.3%	-1.1%
	ResNet	77.9%	72.3%	83.5%	+0.8%	-1.7%
	LSTM	68.5%	63.0%	81.0%	+2.6%	-3.4%
D_8	TCN	59.6%	61.1%	81.8%	+4.6%	+1.1%
	ResNet	74.5%	73.2%	75.3%	+1.3%	-1.6%
	LSTM	66.8%	67.2%	75.9%	+3.1%	-2.3%
D_9	TCN	99.4%	96.9%	75.8%	+2.7%	-1.6%
	ResNet	99.6%	95.7%	76.7%	+1.5%	-2.2%
	LSTM	94.2%	90.6%	72.1%	+2.4%	-2.9%
D_{10}	TCN	87.4%	83.3%	79.9%	+0.5%	-1.7%
	ResNet	86.2%	80.6%	83.5%	-0.9%	-0.2%
	LSTM	86.8%	84.8%	77.0%	+1.8%	-3.5%
D_{11}	TCN	54.0%	52.5%	79.7%	+1.5%	+2.6%
	ResNet	51.3%	52.5%	76.7%	+3.8%	+0.9%
	LSTM	51.1%	51.2%	76.5%	+2.4%	+1.1%
D_{12}	TCN	89.4%	86.0%	82.6%	+1.5%	+0.2%
	ResNet	77.5%	76.0%	83.9%	+3.1%	-0.6%
	LSTM	88.6%	87.1%	83.5%	+3.7%	-1.4%
D_{13}	TCN	93.4%	89.9%	69.4%	+2.6%	-3.1%
	ResNet	92.2%	87.7%	75.1%	+1.2%	-1.6%
	LSTM	84.1%	80.7%	68.6%	+2.0%	-0.8%

VI. Conclusion

In this paper, we proposed a novel generative approach called Time Series Backdoor Attack (TSBA) for backdoor attacks on time series. TSBA is capable of generating highly stealthy and sample-specific trigger patterns for time series backdoor attacks under two different threat models. We studied the difference between image backdoors vs. time series backdoors via three proposed baseline attacks with fixed patterns. We found that the low dimension and fewer degrees of freedom (due to time dependence) nature of time series data make fixed patterns hardly work as backdoor trigger patterns. We empirically show on 13 representative datasets that our proposed TSBA attacks can achieve high ASRs with small drops in clean accuracy, outperforming all three proposed baseline attacks. We analyzed the stealthiness of our attacks as well as their resistance to state-of-the-art backdoor defense methods. We also demonstrate the strong robustness of TSBA to several advanced backdoor defenses. Moreover, our proposed universal trigger generator was also demonstrated to be as effective as dataset-specific TSBA attacks, posing serious threats to deep learning based time series models. Although there are still many unknowns to explore in time series backdoor attacks, our work provides a strong baseline and a good starting point for future research in this area. For future work, we aim to expand our current research and design advanced backdoor defense techniques for time series.

References

[1] O. Peia and K. Roszbach, "Finance and growth: time series evidence on causality," *Journal of Financial Stability*, 2015.

[2] A. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional lstm neural network autoencoders," *IEEE Transactions on Industrial Informatics*, 2020.

[3] R. B. Penfold and F. Zhang, "Use of interrupted time series analysis in evaluating health care quality improvements," *Academic pediatrics*, 2013.

[4] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutt, "Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures," *Frontiers in big data*, 2020.

[5] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[6] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *IJCNN*, 2017.

[7] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, 2017.

[8] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[9] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *ECCV*, 2020.

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

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

[12] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *ICLR*, 2018.

[13] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[14] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *arXiv preprint arXiv:1912.02771*, 2019.

[15] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," *arXiv preprint arXiv:2012.11212*, 2020.

[16] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *USENIX Security*, 2021.

[17] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "Clean-label backdoor attacks on video recognition models," in *CVPR*, 2020.

[18] Y. Sun, T. Zhang, X. Ma, P. Zhou, J. Lou, Z. Xu, X. Di, Y. Cheng, and L. Sun, "Backdoor attacks on crowd counting," in *ACM MM*, pp. 5351–5360, 2022.

[19] Y. Li, H. Zhong, X. Ma, Y. Jiang, and S.-T. Xia, "Few-shot backdoor attacks on visual object tracking," *ICLR*, 2022.

[20] A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *NeurIPS*, 2020.

[21] K. Doan, Y. Lao, W. Zhao, and P. Li, "Lira: Learnable, imperceptible and robust backdoor attacks," in *ICCV*, 2021.

[22] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *CVPR*, 2021.

[23] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, "Backdoor attacks against transfer learning with pre-trained deep learning models," *IEEE Transactions on Services Computing*, 2020.

[24] R. Ning, C. Xin, and H. Wu, "Trojanflow: A neural backdoor attack to deep learning-based network traffic classifiers," in *INFOCOM*, 2022.

[25] D. Ding, M. Zhang, Y. Huang, X. Pan, F. Feng, E. Jiang, and M. Yang, "Towards backdoor attack on deep learning based time series classification," in *ICDE*, 2022.

[26] J. Dumford and W. Scheirer, "Backdooring convolutional neural networks via targeted weight perturbations," in *IJCB*, 2020.

[27] S. Garg, A. Kumar, V. Goel, and Y. Liang, "Can adversarial weight perturbations inject neural backdoors," in *CIKM*, 2020.

[28] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *SIGKDD*, 2020.

[29] Y. Li, J. Hua, H. Wang, C. Chen, and Y. Liu, "Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection," in *ICSE*, 2021.

[30] X. Qi, J. Zhu, C. Xie, and Y. Yang, "Subnet replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting," *arXiv preprint arXiv:2107.07240*, 2021.

[31] 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 *SP*, 2019.

[32] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks.," in *IJCAI*, 2019.

[33] H. Harikumar, V. Le, S. Rana, S. Bhattacharya, S. Gupta, and S. Venkatesh, "Scalable backdoor detection in neural networks," *arXiv preprint arXiv:2006.05646*, 2020.

[34] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *RAID*, 2018.

[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[36] K. Yoshida and T. Fujino, "Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks," in *AISec*, 2020.

[37] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," *ICLR*, 2021.

[38] X. Chen, W. Wang, C. Bender, Y. Ding, R. Jia, B. Li, and D. Song, "Refit: a unified watermark removal framework for deep learning systems with limited data," in *AsiaCCS*, 2021.

[39] H. Cheng, K. Xu, S. Liu, P.-Y. Chen, P. Zhao, and X. Lin, "Defending against backdoor attack on deep neural networks," *SIGKDD Workshop*, 2019.

[40] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," *NeurIPS*, 2021.

[41] D. Wu and Y. Wang, "Adversarial neuron pruning purifies backdoored deep models," *NeurIPS*, 2021.

[42] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *CCS*, 2019.

[43] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," *NDSS*, 2017.

[44] S. O. Gilani, Y. Ilyas, and M. Jamil, "Power line noise removal from ecg signal using notch, band stop and adaptive filters," in *ICEIC*, 2018.

[45] M. Bahaz and R. Benzid, "Efficient algorithm for baseline wander and powerline noise removal from ecg signals based on discrete fourier series," *Australasian physical & engineering sciences in medicine*, 2018.

[46] 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 *CVPR*, 2017.

Appendix A
Details of the datasets and network architectures

TABLE V: Statistics of the selected univariate datasets from the UCR Archive.

Dataset	**Data Type**	**#Class**	**Frame Length**	**Training Samples**	**Test Samples**
(D_1) BirdChicken	*Image*	2	512	20	20
(D_2) ECG5000	*ECG*	5	140	500	4500
(D_3) Earthquakes	*Sensor*	2	512	322	139
(D_4) ElectricDevices	*Device*	7	96	8926	7711
(D_5) Haptics	*Motion*	5	1092	155	308
(D_6) PowerCons	*Power*	2	144	180	180
(D_7) ShapeletSim	*Simulated*	2	500	20	180
(D_8) Wine	*Spectro*	2	234	57	54

TABLE VI: Statistics of the selected multivariate datasets from the MTS Archive.

Dataset	**#Class**	**#Var.**	**Frame Length**	**Training Samples**	**Test Samples**
(D_9) ArabicDigits	10	13	4–93	6600	2200
(D_{10}) ECG	2	2	39–152	100	100
(D_{11}) KickvsPunch	2	62	274–841	16	10
(D_{12}) NetFlow	2	4	50–997	803	534
(D_{13}) UWave	8	3	315	200	4278

TABLE VII: Architectures of the trigger generator network, where "L" denotes the length of the time series input, and "D" denotes the number of time-dependent variables of the time series input.

Layers (Activation)	**Kernel size**	**# Kernels**	**Output size**
Input	-	-	(L, D)
Conv1D (ReLU)	15*1	128*D	(L, 128*D)
Conv1D (ReLU)	21*1	512*D	(L, 512*D)
FC (ReLU)	-	256	(L, 256*D)
FC (tanh)	-	D	(L, D)

TABLE VIII: Architecture of the universal noise generator.

Layers (Activation)	**Kernel size**	**# Kernels**	**Output size**
Input	-	-	(L, D)
Conv1D (ReLU)	15*1	128*D	(L, 128*D)
Conv1D (ReLU)	21*1	512*D	(L, 512*D)
Conv1D (ReLU)	8*1	1024*D	(L, 1024*D)
FC (ReLU)	-	512	(L, 512*D)
FC (tanh)	-	D	(L, D)