Information Sciences 715 (2025) 122216

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Towards undetectable adversarial attack on time series classification

Hoki Kim [a,1], Yunyoung Lee [b,1], Woojin Lee [c,1], Jaewook Lee [d,*]

[a] Chung-Ang University, Heukseok-ro 84, Dongjak-gu, Seoul, Republic of Korea
[b] Sejong University, Neungdong-ro 209, Gwangjin-gu, Seoul, Republic of Korea
[c] Dongguk University-Seoul, Pildong-ro 1, Jung-gu, Seoul, Republic of Korea
[d] Department of Industrial Engineering, Seoul National University, Gwanakro 1, Gwanak-gu, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
Adversarial attack
Detection
Time series
Deep learning

ABSTRACT

Although deep learning models have shown superior performance for time series classification, prior studies have recently discovered that small perturbations can fool various time series models. This vulnerability poses a serious threat that can cause malfunctions in real-world systems, such as Internet-of-Things (IoT) devices and industrial control systems. To defend these systems against adversarial time series, recent studies have proposed a detection method using time series characteristics. In this paper, however, we reveal that this detection-based defense can be easily circumvented. Through an extensive investigation into existing adversarial attacks and generated adversarial time series examples, we discover that they tend to ignore the trends in local areas and add excessive noise to the original examples. Based on the analyses, we propose a new adaptive attack, called trend-adaptive interval attack (TIA), that generates a hardly detectable adversarial time series by adopting trend-adaptive loss and gradient-based interval selection. Our experiments demonstrate that the proposed method successfully maintains the important features of the original time series and deceives diverse time series models without being detected.

1. Introduction

Deep learning models have demonstrated superior performance across a range of tasks, such as image classification [1] and speech recognition [2]. However, Szegedy et al. [3] found that deep learning models are vulnerable to adversarial examples, which are similar to the original examples but cause the models to output incorrect predictions. Adversarial examples pose a high risk to real-world applications of deep learning models because they use subtle perturbations that cannot be distinguished through human perception.

Recently, it has been revealed that time series classification models can also be a target for an adversarial attack [4,5]. Similar to the vision domain, these adversarial time series fool the time series model to output an incorrect prediction with subtle perturbations. In the new industrial age known as Industry 4.0, time series data are being actively processed and thus the existence of adversarial time series leads to privacy and security risks in real-world applications. For example, an attacker can intercept private information by fooling Internet-of-Things (IoT) devices in smart homes or can damage industrial control systems resulting in incorrect behavior [6].

* Corresponding author.
E-mail addresses: hokikim@cau.ac.kr (H. Kim), yylee93@sejong.ac.kr (y. Lee), wj926@dgu.ac.kr (W. Lee), jaewook@snu.ac.kr (J. Lee).
[1] Authors contributed equally.

https://doi.org/10.1016/j.ins.2025.122216
Received 4 November 2021; Received in revised form 17 April 2025; Accepted 17 April 2025

Available online 23 April 2025

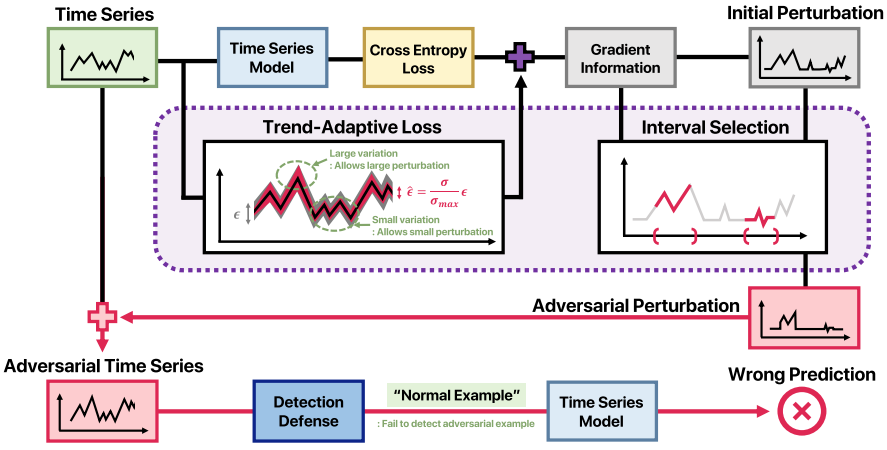

Fig. 1. Illustration of the proposed method. The purple area corresponds to the key components of the proposed method: trend-adaptive loss and gradient-based interval selection. The proposed method circumvents a detection-based defense and deceives the model to output an inaccurate prediction.

To defend time series models against adversarial attacks, a recent study [7] proposed a detection method that distinguishes an adversarial time series from the original time series using chaos-theoretic measures. Furthermore, Abdu-Aguye et al. [7] trained a one-class SVM classifier to classify an adversarial time series and achieved high detection accuracy (over 90%) on several datasets. They argued that adversarial time series generated by existing adversarial attacks can be filtered using this trained detector before they mislead the time series models.

However, in this work, we reveal that such detection-based defenses heavily depend on artifacts arising from the fact that current adversarial attacks in the time series domain are largely adapted from vision-based methods, without adequate consideration of time series-specific properties. Specifically, existing methods tend to ignore trends in local areas and add excessive perturbations, causing the time series characteristics to deviate from the original.

Based on these observations, we propose an adversarial attack method that generates minimally detectable adversarial time series, as illustrated in Fig. 1. Specifically, the proposed method introduces two novel components: *trend-adaptive loss* and *gradient-based interval selection*. Trend-adaptive loss maintains the trend in local areas. It calculates a trend-adaptive maximum perturbation for each time point and regularizes the perturbations larger than the trend-adaptive maximum perturbation during the attack process. Then, gradient-based interval selection extracts sensitive time intervals based on windowed gradient information to minimize the perturbed points. We call this proposed attack method a *trend-adaptive interval attack* (TIA). Extensive experimental results show that TIA effectively evades detection while preserving the essential characteristics of the original time series.

2. Related work

In this section, we first provide mathematical definitions of time series classification and elaborate on both classical and recent classification approaches for time series datasets. We then introduce the general formulation of adversarial attacks and their applications within the time series domain.

2.1. Time series classification

Over the past years, time series classification has received a large amount of interest [8,9]. In general, a time series $\boldsymbol{x} = [x_1, x_2, ..., x_T]$ is an ordered set of real values where x_i is a real value for time step i given with a time length of T. A dataset $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ is a collection of pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, where $\boldsymbol{y}_i$ is a corresponding one-hot label vector. Given with K number of classes, $\boldsymbol{y}_i$ is a vector of length K, where each element $k \in [1, K]$ is 1 if $\boldsymbol{x}_i$ belongs to class k and is 0 otherwise. Time series classification problems consist of training a classifier that correctly maps the possible time series input $\boldsymbol{x}_i$ to the most probable class variable label.

Classical approaches for time series classification mainly consist of distance-based and feature-based models. For distance-based models, many existing machine learning methods such as support vector machines (SVM) or k-nearest neighbor (KNN) can be used with various distance functions. For instance, Euclidean distance is widely used to measure the dissimilarity between two different time series given its simplicity [10]. However, because the Euclidean distance only accounts for linear alignments between samples, it may yield inaccurate dissimilarity estimates in the presence of temporal distortions across time series. To overcome this limitation, dynamic time warping (DTW) [11,12] was developed to measure the minimum path between two different time series samples by providing nonlinear alignments. DTW has been extensively applied in time series tasks to enhance classification accuracy [13,14]. Another major branch of time series classification approaches is feature-based models. Feature-based models aim to discover a set of simple features that represent the corresponding time series sample. For example, the discrete Fourier transform (DFT) method transforms a time series from the time domain to the frequency domain, simplifying the time series to a few Fourier coefficients.

Similar to DFT, discrete wavelet transform (DWT) [15] is also widely used as a time-frequency multi-resolution analysis method in various time series studies.

After the successful use of deep learning for various real-world applications, many researchers have adopted deep learning models within the time series domain. Particularly for time series classification problems, deep learning models have shown superior performance over previous DTW-based methods as they effectively capture time-invariant features from sliding window samples. Zheng et al. [16] proposed a feature learning-based multivariate time series classification model using convolutional neural networks (CNNs). Recurrent neural networks (RNNs) are also frequently implemented in time series classification tasks. Mehdiyev et al. [17] proposed a stacked long short-term memory (LSTM) structure for multivariate time series classifications. Azar et al. [18] implemented RNN-based deep learning models to study the impact of time series compression on time series classification tasks. To evaluate the effectiveness of the proposed attack method, we consider diverse architectures of deep learning time series models, including a simple neural network, CNN, and LSTM.

2.2. Adversarial attack

Adversarial attacks generate malicious examples that deceive the deep learning models with subtle noises [3]. Suppose a neural network f attempts to predict a correct vector $\boldsymbol{y}$ (or correct value y) from an example $\boldsymbol{x}$. Then, given an example $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$, the neural network f is trained by reducing the loss function $\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{y})$. For the classification problem, the cross-entropy loss $\mathcal{L}_{CE}(f(\boldsymbol{x}), \boldsymbol{y}) = -\sum_i y_i \log(f(\boldsymbol{x})_i)$ is widely used as the loss function $\mathcal{L}$, where $f(\boldsymbol{x})_i$ is the i-th element of the output $f(\boldsymbol{x})$. Adversarial attacks outputs an adversarial example $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}$ that maximizes the loss $\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{y})$:

$$\max_{|\boldsymbol{\delta}| \le \epsilon} \mathcal{L}(f(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y}) \tag{1}$$

where ϵ is a maximum perturbation that limits the size of the perturbation. In general, ℓ_∞ is used as the norm, i.e., $|\boldsymbol{\delta}|_\infty \le \epsilon$ [19].

A fast gradient sign method (FGSM) is the simplest gradient-based attack, which uses only one gradient information [20]:

$$\boldsymbol{\delta} = \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{y}))$$
$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}$$

where sign$(\cdot)$ indicates the sign of each element.

Kurakin et al. [21] proposed a basic iterative method (BIM) that uses multiple gradients to maximize the loss and generate an adversarial example from the original example, $\boldsymbol{x}^0 = \boldsymbol{x}$:

$$\boldsymbol{\delta}^{t+1} = \boldsymbol{\delta}^t + \alpha \cdot \text{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}(f(\boldsymbol{x} + \boldsymbol{\delta}^t), \boldsymbol{y}))$$
$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}^s$$

where s is the total number of steps, and α is a step size for updating the perturbation at each step. The final adversarial example should be $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}^s$. Note that the perturbation $\boldsymbol{\delta}$ is clipped before being added to the original example $\boldsymbol{x}$ to satisfy $\boldsymbol{x}' = \min(\max(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{x} - \epsilon), \boldsymbol{x} + \epsilon)$.

Although adversarial attacks have been mainly proposed within the vision domain, recent studies have shown that they can also be readily applied to time series models. [4,5]. Fawaz et al. [4] first adopted FGSM [20] and BIM [21] to the time series classification task. They successfully degraded the performance of ResNet [22] on the overall UCR datasets [23]. In addition, they used multi-dimensional scaling [24] to analyze the perturbed features of the adversarial time series. Karim et al. [5] proposed a gradient adversarial transformation network to deceive one nearest neighbor DTW, which is a type of classical time series classification, based on adversarial transformation network and knowledge distillation. Antal et al. [25] utilized deep autoencoders for generating adversarial mouse trajectories on mouse dynamics datasets under the scheme of unsupervised learning. Additionally, Mode and Hoque [26], Dang-Nhu et al. [27], and Harford et al. [28] confirmed that diverse time series models can be easily attacked through adversarial perturbations.

2.3. Detection adversarial examples

To defend against adversarial attacks, there is a line of work on detection-based defenses within the vision domain. While adversarial training methods [19,29] require a huge computational cost to train a robust model, detection-based methods have the advantage of being post hoc defenses that can be applied after model training. For example, Chen et al. [30] adopted adaptive channel transformation and successfully detected adversarial images.

In the time series domain, Abdu-Aguye et al. [7] recently found that the adversarial time series generated using FGSM and BIM can be easily detected from a benign time series through a simple one-class SVM. A key underlying observation is that the adversarial time series are apparently chaotic. Consequently, the authors employed two chaos-theoretic measures to classify an adversarial time series and the original time series, i.e., a detrended fluctuation analysis [31] and a sample entropy [32].

Detrended fluctuation analysis (DFA). Given a time series $\boldsymbol{x} = [x_1, x_2, \dots x_T]$, DFA first converts $\boldsymbol{x}$ to into $\tilde{\boldsymbol{x}} = [\tilde{x}_1, \tilde{x}_2, \dots \tilde{x}_T]$:

$$\tilde{x}_t = \sum_{i=1}^{t} (x_i - \bar{x}) \tag{2}$$

where $\bar{x}$ denotes the average of $\boldsymbol{x}$. Then, $\tilde{\boldsymbol{x}}$ is divided into small samples of size w. For each sliced sample, a linear least-squares fit is calculated. We denote $\hat{\boldsymbol{x}}_w = [\hat{x}_{w,1}, \hat{x}_{w,2}, \dots, \hat{x}_{w,T}]$ as the concatenated sequence of fitted straight lines. Finally, the fluctuation is calculated as follows:

$$F(w) = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(\tilde{x}_i - \hat{x}_{w,i})^2} \tag{3}$$

Note that the fluctuation is a function of the window size w, and thus different window sizes generate different fluctuations. To ease understanding, we denote a fluctuation in the detrended fluctuation analysis $F(w)$ as DFA(w) in the remainder of our paper.

Sample entropy (SE). Given a time series $\boldsymbol{x} = [x_1, x_2, \dots x_T]$, we first define a template vector $\boldsymbol{x}_w(i) = [x_i, x_{i+1}, \dots x_{i+w-1}]$. The distance $d[\boldsymbol{x}_w(i), \boldsymbol{x}_w(j)]$ between two template vectors $\boldsymbol{x}_w(i)$ and $\boldsymbol{x}_w(j)$ for all $i, j = 1, 2, \dots, T$ is generally calculated with the Chebyshev distance. Finally, the sample entropy can be measured as follows:

$$SE(w, r) = -\log\frac{A}{B} \tag{4}$$

where A denotes the number of template vector pairs satisfying $d[\boldsymbol{x}_{w+1}(i), \boldsymbol{x}_{w+1}(j)] < r$, and B denotes the number of template vector pairs satisfying $d[\boldsymbol{x}_w(i), \boldsymbol{x}_w(j)] < r$. In general, $r = 0.2 * \sigma$, where σ is the standard deviation of the sample $\boldsymbol{x}$ [33]. The small sample entropy corresponds to more self-similarity.

Based on these two chaos-theoretic measures, Abdu-Aguye et al. [7] trained a one-class SVM and obtained promising results on detecting adversarial examples. Specifically, the detector shows an extremely high detection accuracy of up to 97% on several UCR datasets.

Inspired by Abdu-Aguye et al. [7], we investigate why existing time series attacks are easily detected using chaos-theoretic measures. We find that the high detection rate of current adversarial time series attacks stems from their failure to capture intrinsic time series characteristics during the attack process. Furthermore, we demonstrate the existence of attacks that can successfully evade detection-based defenses while preserving these characteristics. It's worth noting that our study aligns with Carlini and Wagner [34] in terms of proposing adaptive attacks to bypass detection; however, our study differs in that it is based on an analysis of detectable adversarial examples in the time series domain.

3. Causes of easily detectable adversarial attacks

In this section, we first experimentally demonstrate that existing adversarial attacks in the time series domain are easily detectable. We then investigate the reasons behind the high detection rate of these adversarial attacks.

3.1. Revisiting detectable adversarial attacks

Although Abdu-Aguye et al. [7] discovered that adversarial time series are easily detected by an SVM trained on chaos-theoretic measures, they considered adversarial examples generated from only one trained model [35]. To provide a more general conclusion, we conducted an experiment using diverse structures of deep learning models on the UCR datasets [36]. We considered three different models: a fully connected neural network (FC), a convolutional neural network (CNN), and an LSTM-based neural network (LSTM). For all datasets, we trained these different models, resulting in three models per dataset. The architectural details and training configurations are presented in Section 5.1.

We first generated adversarial time series datasets using existing attacks (FGSM and BIM) against the trained models. During the attack process, we utilized the ℓ_∞-norm with a maximum perturbation of $\epsilon = 0.1$ following [4,7]. We then extracted the SE and DFA with various window sizes $w \in [3, 5, 7, 9]$ from each clean and adversarial dataset. Next, we trained SVM classifiers to classify the original time series and adversarial time series on the training dataset.

The detection rates on each test dataset are summarized in Table 1 for 55 datasets, showing a detection accuracy better than chance (50%). We emphasize that we achieved better detection rates than [7] using various window sizes, as described in the above settings. Adversarial time series generated via FGSM and BIM exhibit high detection rates across the majority of datasets. For instance, in the OliveOil dataset, SVMs correctly identify adversarial examples with an accuracy exceeding 0.98 across all model types. On average, the detection rates for FGSM and BIM are 0.83 and 0.79, respectively. These results suggest that adversarial time series generated using existing methods are generally detectable through time series-based features.

3.2. Losing its trend in local areas

To investigate the reason why existing methods are easily distinguishable based on the time series characteristics, we first visualize a randomly sampled example. We plot the original time series (blue) and the adversarial time series generated by BIM (red) on a CNN. These observations are not unique to this instance; similar patterns are consistently observed in FGSM-generated adversarial examples across various models. As shown in the top of Fig. 2, the adversarial time series seems to be mostly similar to the original time series. However, we find the notable differences in certain local regions. In the bottom of Fig. 2, we magnify two such regions. For $t \in [10, 20]$, where x_t is widely distributed, ranging from zero to 1, the trend in x'_t is roughly maintained. By contrast, for $t \in [80, 120]$, x'_t ranges from -0.1 to 0.1, which is clearly distinguishable from the original time series x_t even with the human vision system.

Table 1
Detection rates of each attack method on different models (lower is better). The cells are in red if their value is less than or equal to the median value (0.78), i.e., an attack having fewer red cells is easy to be detected. Adversarial examples generated using existing adversarial attacks (FGSM and BIM) are mostly distinguishable from original examples with high detection rates. By contrast, the proposed method (TIA) shows much lower detection rates, which implies that it successfully generates a hardly detectable adversarial time series.

Dataset	FC			CNN			LSTM		
	FGSM	BIM	TIA	FGSM	BIM	TIA	FGSM	BIM	TIA
ACSF1	0.86	0.85	0.53	0.70	0.65	0.52	0.60	0.61	0.57
Adiac	0.99	0.99	0.67	0.99	0.93	0.59	0.99	0.92	0.60
ArrowHead	0.97	0.95	0.72	0.97	0.77	0.63	0.96	0.67	0.54
BME	0.99	0.99	0.69	1.00	0.97	0.84	0.75	0.71	0.59
BeetleFly	0.90	0.83	0.50	1.00	0.93	0.73	1.00	0.85	0.63
BirdChicken	0.95	0.95	0.58	1.00	0.75	0.65	0.98	0.80	0.68
Car	0.99	0.97	0.78	1.00	1.00	0.76	1.00	1.00	0.75
Coffee	1.00	0.98	0.80	1.00	0.77	0.77	1.00	0.79	0.77
Computers	0.92	0.92	0.51	0.55	0.55	0.51	0.57	0.56	0.50
DiatomSizeReduction	0.96	0.96	0.81	0.99	0.98	0.76	0.99	0.99	0.82
ECG200	0.68	0.62	0.52	0.56	0.56	0.52	0.50	0.50	0.54
ECG5000	0.86	0.85	0.53	0.72	0.72	0.51	0.82	0.74	0.57
ECGFiveDays	0.82	0.83	0.56	0.96	0.78	0.70	0.96	0.59	0.56
FaceFour	0.61	0.63	0.50	0.68	0.61	0.51	0.51	0.57	0.55
FacesUCR	0.59	0.58	0.50	0.65	0.63	0.55	0.59	0.63	0.53
Fish	0.99	0.99	0.82	1.00	0.99	0.84	1.00	1.00	0.80
FordA	0.85	0.79	0.56	0.99	0.95	0.66	0.99	0.75	0.57
FreezerRegularTrain	1.00	0.98	0.85	0.99	0.98	0.83	0.99	0.96	0.51
FreezerSmallTrain	0.97	0.98	0.59	0.87	0.82	0.64	0.60	0.61	0.61
GunPoint	0.91	0.89	0.70	0.97	0.83	0.66	0.96	0.75	0.65
InsectEPGRegularTrain	0.91	0.91	0.50	0.55	0.50	0.50	0.80	0.80	0.52
InsectEPGSmallTrain	0.89	0.90	0.57	0.51	0.51	0.57	0.88	0.83	0.50
LargeKitchenAppliances	0.97	0.97	0.50	0.66	0.68	0.50	0.54	0.54	0.51
Lightning7	0.79	0.79	0.51	0.68	0.66	0.52	0.53	0.51	0.50
Mallat	0.96	0.91	0.70	0.98	0.89	0.68	0.98	0.85	0.64
Meat	1.00	1.00	0.75	1.00	0.98	0.69	1.00	1.00	0.69
MixedShapesRegularTrain	0.98	0.95	0.59	1.00	1.00	0.87	1.00	1.00	0.53
MixedShapesSmallTrain	0.96	0.92	0.56	1.00	0.99	0.86	1.00	0.99	0.52
MoteStrain	0.69	0.68	0.51	0.64	0.62	0.51	0.62	0.62	0.51
NonInvasiveFetalECGThorax1	1.00	0.99	0.69	1.00	0.94	0.64	1.00	0.92	0.61
NonInvasiveFetalECGThorax2	1.00	1.00	0.71	1.00	0.91	0.64	1.00	0.94	0.62
OSULeaf	0.93	0.89	0.57	1.00	0.94	0.73	1.00	0.90	0.74
OliveOil	0.98	1.00	0.82	1.00	1.00	0.75	1.00	1.00	0.75
PigArtPressure	0.51	0.51	0.50	0.53	0.53	0.50	0.51	0.51	0.50
Plane	0.88	0.85	0.58	0.90	0.79	0.75	0.89	0.82	0.82
PowerCons	0.81	0.79	0.50	0.72	0.70	0.52	0.64	0.60	0.51
ProximalPhalanxOutlineAgeGroup	0.98	0.99	0.80	0.96	0.96	0.80	0.97	0.97	0.61
ProximalPhalanxOutlineCorrect	0.98	0.97	0.67	0.97	0.93	0.61	0.97	0.92	0.64
Rock	0.57	0.52	0.50	0.51	0.52	0.52	0.51	0.50	0.50
ShapeletSim	0.57	0.56	0.53	0.56	0.56	0.51	0.61	0.64	0.50
ShapesAll	0.89	0.84	0.59	0.99	0.92	0.74	0.98	0.90	0.70
StarLightCurves	0.99	0.99	0.79	1.00	1.00	0.83	1.00	0.99	0.90
Strawberry	0.99	1.00	0.59	1.00	0.99	0.74	1.00	0.98	0.66
SwedishLeaf	0.79	0.79	0.62	0.89	0.74	0.58	0.82	0.76	0.60
Symbols	0.86	0.86	0.57	0.94	0.82	0.75	0.96	0.79	0.71
ToeSegmentation1	0.73	0.71	0.52	0.71	0.66	0.53	0.68	0.59	0.51
ToeSegmentation2	0.68	0.66	0.50	0.87	0.68	0.54	0.71	0.60	0.51
Trace	0.98	0.93	0.63	0.82	0.77	0.60	0.78	0.54	0.67
TwoLeadECG	0.74	0.73	0.59	0.88	0.73	0.66	0.66	0.53	0.53
TwoPatterns	0.84	0.81	0.50	0.94	0.91	0.51	0.84	0.80	0.50
UMD	1.00	1.00	0.79	0.98	0.99	0.67	0.97	0.90	0.57
Wafer	1.00	1.00	0.50	0.96	0.96	0.56	0.96	0.77	0.58
Wine	1.00	1.00	0.75	1.00	1.00	0.57	1.00	0.97	0.67
Worms	0.80	0.79	0.55	0.97	0.88	0.59	0.96	0.77	0.63
Yoga	0.99	0.98	0.77	1.00	0.98	0.66	0.99	0.98	0.70
Average	0.88	0.87	0.62	0.87	0.81	0.64	0.85	0.78	0.61

We reveal that they are also numerically distinguished in terms of the standard deviation. At the bottom of Fig. 2, we plot a ratio of windowed standard deviations that compares the windowed standard deviation σ [37] of the original time series $\boldsymbol{x}$ and the adversarial time series $\boldsymbol{x}'$. Mathematically, the ratio of windowed standard deviations is calculated as follows:

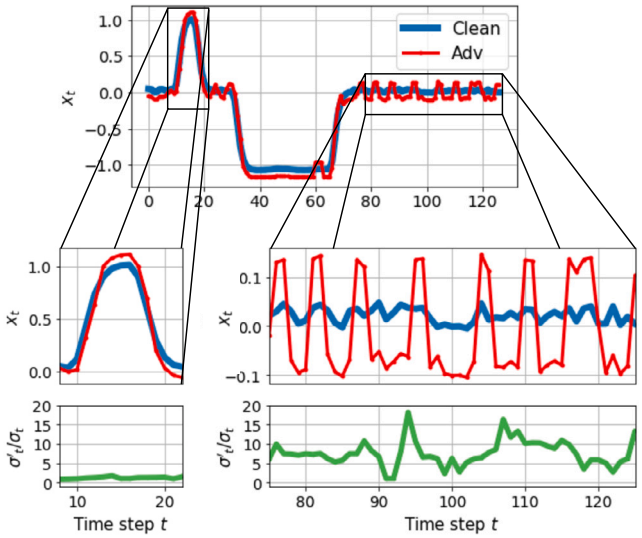

Fig. 2. Top: Visualization of the original time series $\boldsymbol{x}$ (Clean) and the adversarial time series $\boldsymbol{x}'$ of BIM (Adv) on CNN. Bottom: enlarged version of two different intervals and their ratio of windowed standard deviations in Equation (5).

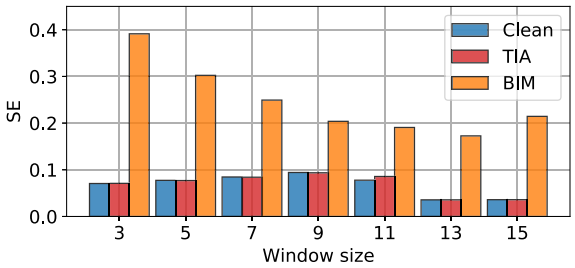

Fig. 3. Sample entropy (SE) of the original time series $\boldsymbol{x}$ and the adversarial time series $\boldsymbol{x}'$ in Fig. 2 for different window sizes. The adversarial time series of BIM (orange) shows a much higher SE than that of the original time series (blue) for all window sizes, particularly for lower window sizes. By contrast, the SE of the adversarial time series generated by the proposed method (TIA) is mostly similar to the original one.

$$\frac{\sigma'_t}{\sigma_t} = \frac{\sum_{i\in[1,\dots,w]}(x'_{t+i-1} - \bar{x}'_t)^2}{\sum_{i\in[1,\dots,w]}(x_{t+i-1} - \bar{x}_t)^2} \tag{5}$$

where w is the window size, and $\bar{x}_t = \frac{1}{w}\sum_{i\in[1,\dots,w]} x_{t+i-1}$. Thus, a larger ratio indicates that the adversarial time series is more differentiated from the original time series in terms of the standard deviation. Here, we used a window size of $w = 5$. For $t \in [10, 20]$, the ratio is lower than 5. However, for $t \in [80, 120]$, the ratio is values of over 5 for almost all points. Thus, we can conclude that the adversarial time series is easily distinguished from the original time series even based on the standard deviation.

To push further, we also compare the value of SE with varying window sizes. Since SE indicates the degree of entropy in local areas with the specific window size, the SE value is expected to have a large value when the window size is small because the adversarial time series easily loses its trend in local areas as observed in Fig. 2. Indeed, as shown in Fig. 3, the difference in SE is large when the window size is small. This difference suggests that SE can serve as a highly informative feature when training an SVM classifier.

The underlying reason for this phenomenon is that existing methods do not consider the trend in the original time series. Specifically, in the vision domain, a fixed maximum perturbation rarely perturbs the global structure or robust features of an image. However, in the time series domain, all data points exhibit a unique trend and seasonality even in local areas. While perturbations in local areas are as important as the global shape of the time series, trends are not considered in existing attacks because they are simply adapted from the vision domain. In summary, time series characteristics are difficult to preserve during the attack process due to the ignorance of the trend characteristics in existing adversarial attacks.

3.3. Excessive perturbations in the global scope

It is widely known that only a small number of perturbations might be sufficient to deceive models within the vision domain [38,39]. Intuitively, more adversarial perturbations bring more variability to the original time series characteristics. Thus, if existing methods generate excessive (or useless) perturbations, we can expect to make an adversarial time series hardly detectable by removing such excessive perturbations.

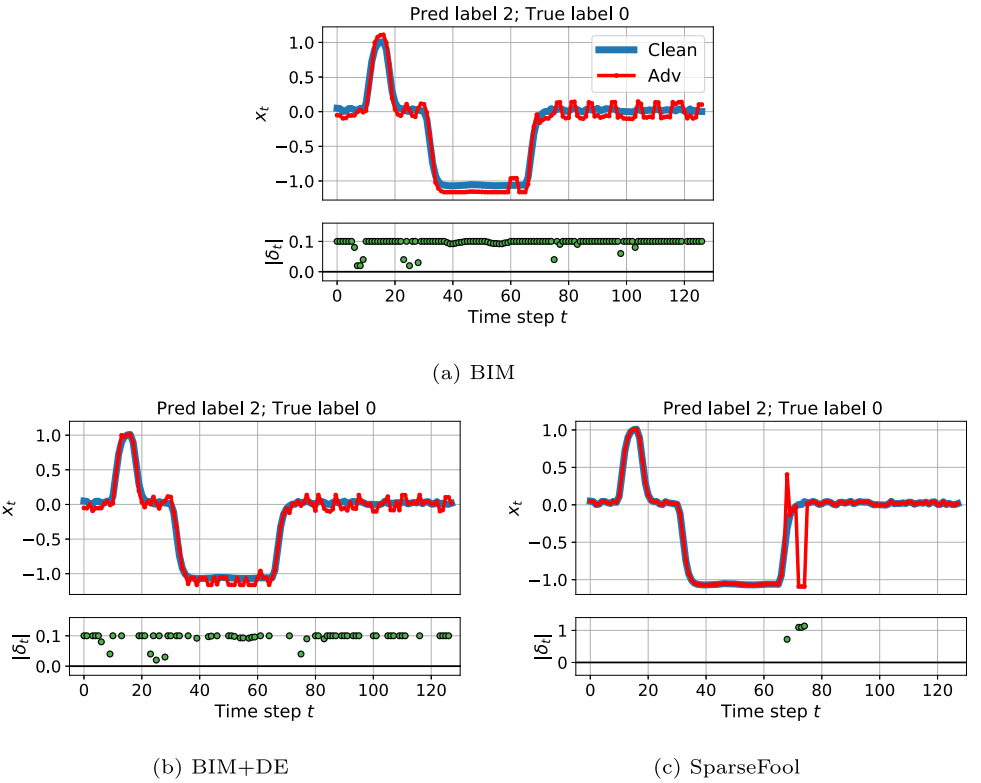

Fig. 4. Visualization of the original time series $\boldsymbol{x}$ (Clean) and the adversarial time series $\boldsymbol{x}'$ of each attack method (Adv). We also plotted the size of perturbation $|\boldsymbol{\delta}| = |\boldsymbol{x}' - \boldsymbol{x}|$ at the bottom. The predicted and true labels are in the title. The same sample in Fig. 2 is used. (a) The adversarial time series generated by BIM. (b) We successfully reduce the ratio of the number of perturbed points δ_t to 50% from 100% while maintaining the false prediction by using a different evolution (DE). (c) SparseFool also supports the possibility that only a few perturbed points are sufficient to deceive the time series model if the perturbation size is unlimited.

Table 2
Chaos-theoretic measures and important features of the original (Clean) and adversarial examples of each attack method. The same sample as that in Fig. 2 is used.

	SE	DFA	Perturbed	**Detected**	Time
Clean	0.08	0.08	-	-	-
BIM	0.30	0.11	100.00%	O	0.07 s
BIM+DE	0.29	0.09	50.78%	O	24.62 s
SparseFool	0.08	0.12	3.12%	O	0.24 s
TIA	0.08	0.08	3.91%	X	0.06 s

To verify the existence of excessive perturbations in an adversarial time series and explore the possibility of reducing their number, we employ two different approaches: Differential Evolution and SparseFool. (i) Differential Evolution (DE): DE is a population-based optimization technique [40] commonly used for solving complex problems. We apply DE to eliminate excessive perturbations generated by BIM while maintaining the prediction of models. (ii) SparseFool: SparseFool [39] is a powerful ℓ_0-norm adversarial attack method that deceives a model using only a few perturbed points within the data domain. By utilizing SparseFool, we can approximate the minimum number of perturbations required to deceive the time series model, as it has a limit on the number of perturbations rather than their size. To the best of our knowledge, this work is the first attempt at applying SparseFool within the time series domain.

Fig. 4 and Table 2 present the results of each method. Both DE and SparseFool successfully reduce the ratio of perturbed points. Specifically, DE eliminates 49.22% of excessive adversarial perturbations from BIM and SparseFool deceives the model with only four perturbed points (3.12%). We note that these results are similarly observed in other FGSM and BIM adversarial examples. Consequently, we can conclude that existing attack methods generate excessive perturbations.

While both methods demonstrate the existence of excessive perturbations, they still have limitations in preserving the time series characteristics. For DE, although it reduces both SE and DFA, it is insufficient to maintain the original characteristics. As shown in Table 2, its SE value is approximately four times larger than its original value. In contrast, while SparseFool successfully maintains a similar level of SE, its DFA is larger than that of Clean and BIM. This is because DFA uses the root-mean-square deviation from the

Table 3
Detection rates of attack methods. Red cells correspond to the lowest detection rates.

Dataset	FGSM	BIM	BIM+DE	SparseFool	TIA
ACSF1	0.86	0.85	0.80	0.85	0.53
Adiac	0.99	0.99	0.86	0.89	0.67
ArrowHead	0.97	0.95	0.85	0.90	0.72
BME	0.99	0.99	0.90	0.95	0.69
BeetleFly	0.90	0.83	0.77	0.85	0.50

trend, whereas there is no limit to the perturbation size for SparseFool. Indeed, as shown in Fig. 4c, SparseFool drastically changes the perturbed points sufficiently to affect the global trend. Moreover, they require a significant amount of time to calculate the optimal perturbations. Specifically, DE takes 24.62 s and SparseFool takes 0.24 s per example, which are considerably slower than BIM (0.06 s) and may be impractical for time-sensitive applications.

Due to these limitations, both methods are still easily detected using the trained SVM. In Table 3, we conduct an additional experiment on the first five UCR datasets with CNN. Both BIM+DE and SparseFool have marginal effects on the detection rates compared to their computational costs. Especially, since SparseFool does not support batch-wise computation, it takes 38.44 s for 100 mini-batches, while BIM only requires 0.24 s on average over 10 runs. Thus, both methods are ineffective in terms of generating a hardly detectable adversarial time series due to their computational cost and less improvement on detection rates. Therefore, we conclude that a new method is required to generate minimal perturbations taking into account both computational costs and detection rates.

4. Trend-adaptive interval attack

From the analyses described in Section 3.2 and 3.3, we propose a new adversarial attack for time series data by resolving two issues: (i) preserving the trend in local areas and (ii) reducing excessive perturbations in the global scope.

Trend-adaptive loss. As discussed in Section 3.2, existing methods suffer from the problem of losing the trend in local areas, because they do not consider the trend of the original time series. Thus, to preserve the trend, we propose a trend-adaptive loss that regularizes the perturbation size in consideration of the trend.

First, leveraging the windowed standard deviation σ_t defined in Equation (5), we adjust the maximum perturbation for each interval proportional to the ratio of σ_t. Specifically, given the maximum perturbation ϵ, an adjusted maximum perturbation for time t is $\sigma_t/\sigma_{\max} \cdot \epsilon$. This process is motivated by Fig. 2, which shows a large difference between the standard deviation of the benign and adversarial time series. By assigning different maximum perturbations proportional to their standard deviation, we can generate trend-adaptive adversarial perturbations that preserve the original trend. Finally, we adopt the perturbation over the adjusted maximum perturbation by introducing a trend-adaptive loss:

$$\mathcal{L}_{TP}(\boldsymbol{x}', \boldsymbol{x}, \epsilon) = \sum_t \left\| |x'_t - x_t| - \sigma_t/\sigma_{\max} \cdot \epsilon \right\|. \tag{6}$$

In consideration of Equation (6), the perturbation is calculated to maximize the following optimization during the attack process.

$$\max_{|\boldsymbol{\delta}| \le \epsilon} \mathcal{L}_{CE}(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y}) - c \cdot \mathcal{L}_{TP}(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}, \epsilon) \tag{7}$$

If $c = \infty$, it forces the use of a new maximum perturbation $\hat{\epsilon}_t = \sigma_t/\sigma_{\max} \cdot \epsilon$ for each point t. In our implementation, we conditioned the trend-adaptive loss $\mathcal{L}_{TP}$ with $\mathbb{1}\{\max_i p_i \neq y\}$ to find stronger adversarial examples by exploring a wider search space. We experimentally observed that the proposed method shows better performance in preserving the trend in local areas and the chaos-theoretic measures than the comparison method, which will be described in Section 6.1.

Gradient-based interval selection. As described in Section 3.3, although DE and SparseFool successfully reduce the excessive perturbed points in an adversarial time series, they both fail to preserve the original characteristics of the time series and incur substantial computational costs. To address these shortcomings, we utilize gradient information. The gradient information is a traditional feature used to provide visual explanations for deep learning models in both the vision and time series domains [41,42]. For example, a point with a larger gradient indicates greater relative importance compared to other points, under a local linearity assumption.

The gradient of the loss function with respect to the perturbation $\boldsymbol{\delta}$ can be formalized as follows:

$$\boldsymbol{g} = \nabla_{\boldsymbol{\delta}} \mathcal{L}(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y}). \tag{8}$$

Given this gradient $\boldsymbol{g}$, we propose *a windowed gradient information*. For the window size w, we calculate the windowed gradient $\mathbf{G}$ from the gradient $\boldsymbol{g}$ where the element of $\mathbf{G}$ for time t is as follows:

$$G_t = \sum_{i \in [1,\dots,w]} g_{t+i-1}. \tag{9}$$

Algorithm 1 Trend-adaptive Interval Attack (TIA).

Require: model f; loss function $\mathcal{L}$; waveform $\boldsymbol{x}$; a target label or sentence $\boldsymbol{y}$; number of steps s; maximum perturbation ϵ; window size w; trend weight c.

$\bar{x}_t = \frac{1}{w}\sum_{i\in[1,\dots,w]} x_{t+i-1}$

$\sigma_t \leftarrow \sqrt{\sum_{i\in[1,\dots,w]}(x_{t+i-1}-\bar{x}_t)^2}$

$\sigma_{\max} \leftarrow \max_t \sigma_t$

$\boldsymbol{\delta}^1 \leftarrow \mathbf{0}$

for j in [1, ..., s] **do**

  (i) Trend-adaptive loss:

  $\boldsymbol{p} = f(\boldsymbol{x}+\boldsymbol{\delta}^j)$

  $\mathcal{L}(\boldsymbol{\delta}^j; f, \boldsymbol{x}, \boldsymbol{y}) = \mathcal{L}_{CE}(\boldsymbol{p}, \boldsymbol{y}) - c \cdot \mathcal{L}_{TP}(\boldsymbol{x}+\boldsymbol{\delta}^j, \boldsymbol{x}, \epsilon) \cdot \mathbb{1}\{\max_i p_i \neq y\}$

  (7)

  (ii) Gradient-based interval selection:

  $\boldsymbol{g} = \nabla_{\boldsymbol{\delta}^j}\mathcal{L}(\boldsymbol{\delta}^j; f, \boldsymbol{x}, \boldsymbol{y})$ (8)

  $G_t \leftarrow \sum_{i\in[1,\dots,w]} g_{t+i-1}$ (9)

  $I_t \leftarrow \mathbb{1}\{\exists\, i \in [1,\dots,w] \text{ s.t. } G_{t+i-1} \geq \tau\}$ (10)

  $\boldsymbol{\delta}^{j+1} \leftarrow \mathbf{I} \odot (\boldsymbol{\delta}^j + \alpha \cdot \mathrm{sgn}(\boldsymbol{g}))$

end for

$\boldsymbol{x}' \leftarrow \boldsymbol{x} + \boldsymbol{\delta}^{s+1}$

From the windowed gradient $\mathbf{G}$, we extract the important intervals to be updated. Note that point-wise perturbations, such as SparseFool, can be easily detected through chaos-theoretic measures as observed in Table 2. Therefore, we select intervals with G_t larger than a threshold τ.

$$I_t = \mathbb{1}\{\exists\, i \in [1,\dots,w] \text{ s.t. } G_{t+i-1} \geq \tau\} \tag{10}$$

In our implementation, τ is decreased linearly over the entire range of G_t if the attack fails after half of the steps. Fortunately, the proposed interval selection method introduces only negligible additional computations, because the gradient is automatically calculated during the attack process. Based on trend-adaptive loss and gradient-based interval selection, we propose a new attack method named Trend-adaptive Interval Attack (TIA). The details of the proposed method are presented in Algorithm 1.

5. Experimental results

In this section, we present a comprehensive comparison of the proposed method (TIA) with existing attack approaches. We first demonstrate that TIA is almost undetectable by the detection method of Abdu-Aguye et al. [7], and show that it effectively bypasses the detection-based defense mechanism, successfully misleading the time series model into producing incorrect predictions.

5.1. Experiment settings

For a thorough evaluation, we considered three different structures as described in Section 3. For FC, we use three fully connected layers. Similarly, a CNN consists of three convolutional layers with a kernel size of $[8, 5, 3]$. For the LSTM, we adopt the model described by [43], which achieves a high performance in various datasets. All layers are followed by batch normalization and a Rectified Linear Unit (ReLU). We trained each model for 100 epochs with the Adam optimizer. We decayed the learning rate through cosine learning rate annealing from 0.1 at the beginning of the training.

During the attack process, we used the maximum perturbation $\epsilon = 0.1$ following [7,4]. For all iterative attack methods, we set the number of steps to 20. Compared to comparison methods, TIA introduces two additional hyperparameters: the window size w and the trend weight c. The window size w affects both sub-processes, i.e., trend-adaptive loss and gradient-based interval selection because it controls the value of the windowed standard deviation and the windowed gradients. Note that the values for w equal to or greater than 13 will result in a high detection rate because too many intervals are perturbed. Thus, we explored different values of w under 13 and found that $w = 5$ achieves stable performance across all datasets (detailed analysis of the window size is presented in Section 6.3). The trend weight c controls the regularization effect of $\mathcal{L}_{TP}$ in Equation (6). We tuned c for every model individually among [0, 1, 10, 100] on the first batch. Note that c over 100 generates almost the same or poorer adversarial time series than $c = 100$. A more detailed analysis on the hyper-parameters of TIA is described in Section 6.3.

5.2. Hardly detectable adversarial attack

First, we verify whether TIA generates a less detectable adversarial time series compared to existing methods. The result on the detection rate of TIA is summarized in Table 1. TIA shows much lower detection rates than any other existing attack methods. Specifically, for the ProximalPhalanxOutlineCorrect dataset, TIA is not easily detected with a maximum detection rate of 0.67 compared to FGSM (0.97) and BIM (0.94). Statistically, TIA (146 out of 165) outperforms FGSM (44 out of 165) and BIM (60 out of 165) in terms of the number of red cells, which indicates the number of datasets with less than or equal to the median value (0.78). TIA also shows a detection rate of 0.62 over the entire datasets, which is much lower than the detection rates of FGSM and BIM (0.86 and 0.82). Therefore, we conclude that TIA generates adversarial time series that are significantly more difficult to detect than those generated by other benchmark methods.

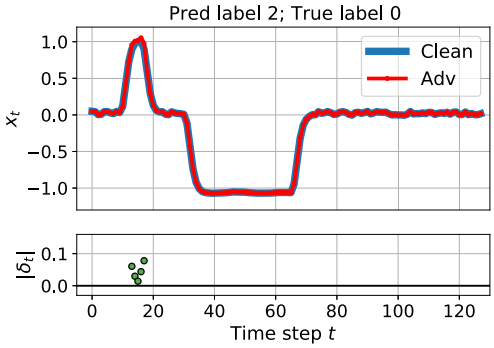

Fig. 5. Visualization of the adversarial time series $\boldsymbol{x}'$ generated by TIA (Adv). The same sample as that in Fig. 2 is used. TIA deceives the model (CNN) with only five perturbed points.

Fig. 5 shows the adversarial time series generated by TIA with the same sample shown in Fig. 4. TIA successfully fools the model to output the incorrect prediction by perturbing only one interval. Indeed, as shown in Fig. 3 and Table 2, the adversarial example of TIA has the same SE and DFA as the original example, whereas it only introduces a negligible additional computation (0.066 s). Note that we further observed that TIA generally generates a minimal change, which will be described in Section 6.1. This minimal change of TIA leads to a better performance than the existing methods.

5.3. Attacking the defense system

Now, we evaluate the success rate of each attack method under the detection-based defense system. Assuming a powerful detection-based defense, we trained SVM on the entire adversarial datasets generated through FGSM, BIM, and TIA. Through this process, we generate a defense system by filtering adversarial time series $\mathbf{x}'$, which are classified as malicious before being fed to a classifier f.

Let us denote the detection-based defense system $D : \boldsymbol{x} \rightarrow \{0, 1\}$, which outputs a value of zero if $\boldsymbol{x}$ is classified as an adversarial example, and one otherwise. Then, the attack success rate under the defense system $\mathcal{A}_{Def}$ of an attack method can be formulated as follows:

$$\mathcal{A}_{Def} = \mathbb{E}_{(\boldsymbol{x},y)} \left[f(\boldsymbol{x}') \neq y | D(\boldsymbol{x}') = 1 \right], \tag{11}$$

where $\boldsymbol{x}'$ is an adversarial example generated by an adversarial attack, and y is a corresponding correct label of $\boldsymbol{x}$. Thus, only an adversarial time series that deceives both the defense system and the classifier will be considered a successful adversarial time series.

The results are summarized in Table 4. Although there is some variability due to the differing characteristics of time series across datasets and the diversity inherent in deep learning models, TIA demonstrates superior performance compared to the baseline methods across most datasets and models. Specifically, for FC, the $\mathcal{A}_{Def}$ of TIA is 41.15%, which outperforms FGSM (6.07%) and BIM (7.16%). Similarly, we observe large gaps for CNN and LSTM. Interestingly, TIA completely deceives the defense system and all classifiers for the Wine dataset, whereas neither FGSM nor BIM succeeded in attacking the defense system. In other words, TIA not only generates perturbations that are difficult to detect, but also effectively evades the detection-based system.

To statistically verify the performance of the proposed method, we further analyze the results in Table 4. Here, we use the results on CNN, but we note that similar results for different models, FC and LSTM, are observed. In Table 5, we summarize the results of a one-way analysis of variance with the attack success rates of FGSM, PGD, and TIA. As shown in this table, the null hypothesis (i.e., FGSM, PGD, and TIA have similar attack success rates) is rejected at the significance level $\alpha = 0.05$. In Table 6, we can confirm no statistical difference between FGSM and PGD, whereas TIA statistically improves the attack success rates.

6. Explaining the effectiveness of TIA

At a high level, the proposed method adopts two different techniques, i.e., trend-adaptive loss and gradient-based interval selection, to generate hardly detectable perturbations. In this section, we analyze the effectiveness of each technique and provide detailed explanations on the strength of TIA under the detection-based defense system.

6.1. Preserving time series characteristics

As shown in Fig. 5, TIA only perturbs five points of the original time series, which is much fewer than BIM or BIM+DE in Fig. 4. However, some might argue that these figures are generated from one sample. To prove that TIA generally induces the original time series, we plot boxplots of chaos-theoretic measures for each dataset. The results are shown in Fig. 6 and Fig. 7. For a better visualization, we sorted datasets according to their median value of SE and DFA, respectively.

In the case of SE (Fig. 6), for most of the datasets, TIA (red) has a more similar distribution to the original examples (blue) than any other comparison methods. In other words, TIA successfully preserves the SE, which is not achieved by other attack methods. Specifically, for the Meat, OliveOil, and NonInvasiveFetalECGThorax2 datasets, the boxplot of TIA lies within the same level as the

Table 4
Attack success rate under the defense system $\mathcal{A}_{Def}$ (%) of each attack method on different models (higher is better). The most powerful attack method is colored green for each dataset and model. The proposed method (TIA) achieves the best performance for most cases.

Dataset	FC			CNN			LSTM		
	FGSM	BIM	TIA	FGSM	BIM	TIA	FGSM	BIM	TIA
ACSF1	0.26	0.77	73.40	0.00	11.51	85.93	38.00	41.00	61.00
Adiac	1.71	1.14	48.00	1.14	32.00	77.71	0.26	21.23	75.96
ArrowHead	5.00	0.00	40.00	0.00	15.00	75.00	0.00	28.57	85.14
BME	11.00	12.00	72.00	31.00	54.00	68.00	1.33	16.67	26.67
BeetleFly	0.00	5.00	50.00	0.00	50.00	65.00	0.00	20.00	50.00
BirdChicken	0.00	0.00	27.33	0.00	2.00	31.33	15.00	20.00	60.00
Car	0.00	1.67	26.67	0.00	0.00	90.00	0.00	0.00	86.67
Coffee	0.00	0.00	64.29	0.00	46.43	89.29	0.00	46.43	96.43
Computers	4.40	6.80	70.40	30.80	42.00	47.20	22.80	38.80	46.80
DiatomSizeReduction	9.00	10.00	15.00	28.00	36.00	33.00	0.00	0.33	33.01
ECG200	2.00	2.40	9.11	6.58	8.89	9.51	14.00	19.00	21.00
ECG5000	8.36	9.76	33.10	0.70	32.64	45.06	6.93	11.00	5.73
ECGFiveDays	18.34	20.05	21.51	0.00	0.12	33.28	0.00	46.57	42.16
FaceFour	0.80	0.80	0.40	0.68	5.76	60.30	28.41	38.64	70.45
FacesUCR	16.87	16.87	8.03	0.00	14.25	72.47	12.00	16.98	11.46
Fish	0.00	0.00	100.00	0.00	0.00	100.00	0.00	0.00	50.86
FordA	2.67	2.93	90.13	0.07	0.42	38.60	0.53	0.98	20.15
FreezerRegularTrain	0.00	1.37	43.84	0.00	11.57	77.69	0.14	2.14	47.02
FreezerSmallTrain	0.45	1.40	18.35	0.00	22.00	62.67	20.49	27.61	21.68
GunPoint	0.95	2.64	25.36	4.33	4.33	4.81	0.00	22.00	42.67
InsectEPGRegularTrain	44.71	44.71	44.71	20.00	20.00	20.00	0.00	0.00	0.00
InsectEPGSmallTrain	0.00	0.00	2.86	12.45	12.45	3.61	12.05	12.05	10.84
LargeKitchenAppliances	1.03	5.15	68.38	0.00	3.33	100.00	36.00	41.87	46.93
Lightning7	8.00	8.00	10.00	1.96	14.33	43.41	16.44	13.70	31.51
Mallat	62.78	58.89	68.89	21.67	23.89	20.56	2.86	19.83	62.73
Meat	4.00	10.33	45.67	0.00	4.33	51.67	0.00	0.00	100.00
MixedShapesRegularTrain	2.29	0.98	50.98	0.00	0.65	55.88	0.00	0.16	28.41
MixedShapesSmallTrain	7.95	7.95	14.77	0.00	0.04	46.76	0.00	0.54	27.92
MoteStrain	0.00	0.00	23.43	7.75	8.95	9.74	5.51	6.39	5.43
NonInvasiveFetalECGThorax1	10.15	18.11	40.83	23.86	42.05	65.91	0.05	22.85	70.74
NonInvasiveFetalECGThorax2	0.11	2.60	42.28	16.98	24.78	26.98	0.00	23.41	52.16
OSULeaf	4.67	4.67	32.67	0.00	11.81	75.93	0.00	12.81	76.86
OliveOil	3.37	1.75	40.88	0.00	0.00	58.29	0.00	0.00	100.00
PigArtPressure	1.83	7.33	49.42	4.98	27.05	38.00	2.40	2.40	2.88
Plane	3.43	4.07	10.94	0.00	26.67	31.43	4.76	22.86	18.10
PowerCons	0.10	0.46	46.77	12.78	16.67	29.44	4.44	5.00	7.22
ProximalPhalanxOutlineAgeGroup	0.00	0.05	38.88	5.85	2.44	86.34	2.44	5.85	56.10
ProximalPhalanxOutlineCorrect	0.00	0.00	100.00	4.12	13.06	87.63	1.37	16.15	87.29
ShapeletSim	0.00	1.11	3.33	24.80	26.67	49.07	10.00	10.00	5.56
ShapesAll	3.90	1.95	83.41	12.33	13.70	35.62	0.00	6.17	42.17
StarLightCurves	0.06	0.10	8.46	0.69	0.69	36.11	0.05	1.23	5.06
Strawberry	0.27	0.00	48.65	0.00	0.00	4.78	0.00	1.89	95.95
SwedishLeaf	9.12	11.52	37.76	0.00	2.70	97.03	9.12	23.20	35.52
Symbols	6.03	6.53	28.14	5.12	33.60	52.32	7.44	26.83	29.65
ToeSegmentation1	6.58	8.77	30.70	7.24	28.64	48.74	5.26	19.30	33.33
ToeSegmentation2	10.77	13.08	24.62	12.28	19.30	29.82	8.46	24.62	27.69
Trace	2.00	7.00	66.00	7.69	26.92	43.08	5.00	31.00	40.00
TwoLeadECG	28.27	30.82	54.78	4.00	29.00	37.00	32.13	38.89	53.47
TwoPatterns	11.82	16.20	46.40	18.53	48.46	72.43	0.45	0.52	1.38
UMD	0.00	0.00	10.42	3.20	7.22	42.70	0.00	2.78	18.06
Wafer	0.03	0.05	0.89	7.58	5.00	69.81	5.50	1.62	35.50
Wine	0.00	0.00	100.00	0.00	0.00	100.00	0.00	0.00	100.00
Worms	11.69	12.99	46.75	0.00	5.19	74.03	0.00	10.39	68.83
Yoga	0.07	0.37	42.33	0.00	1.83	96.27	0.00	2.17	92.17
Average	6.07	7.16	41.15	6.17	16.19	52.86	6.36	15.32	44.41

Table 5
One-way analysis of variance (ANOVA) table for the attack success rate in Table 4.

	df	sum_sq	mean_sq	F	PR(>F)
C(Attack success rate)	2.0	43762.2	21881.1	70.4	1.0e-22
Residual	162.0	50378.3	311.0	NaN	NaN

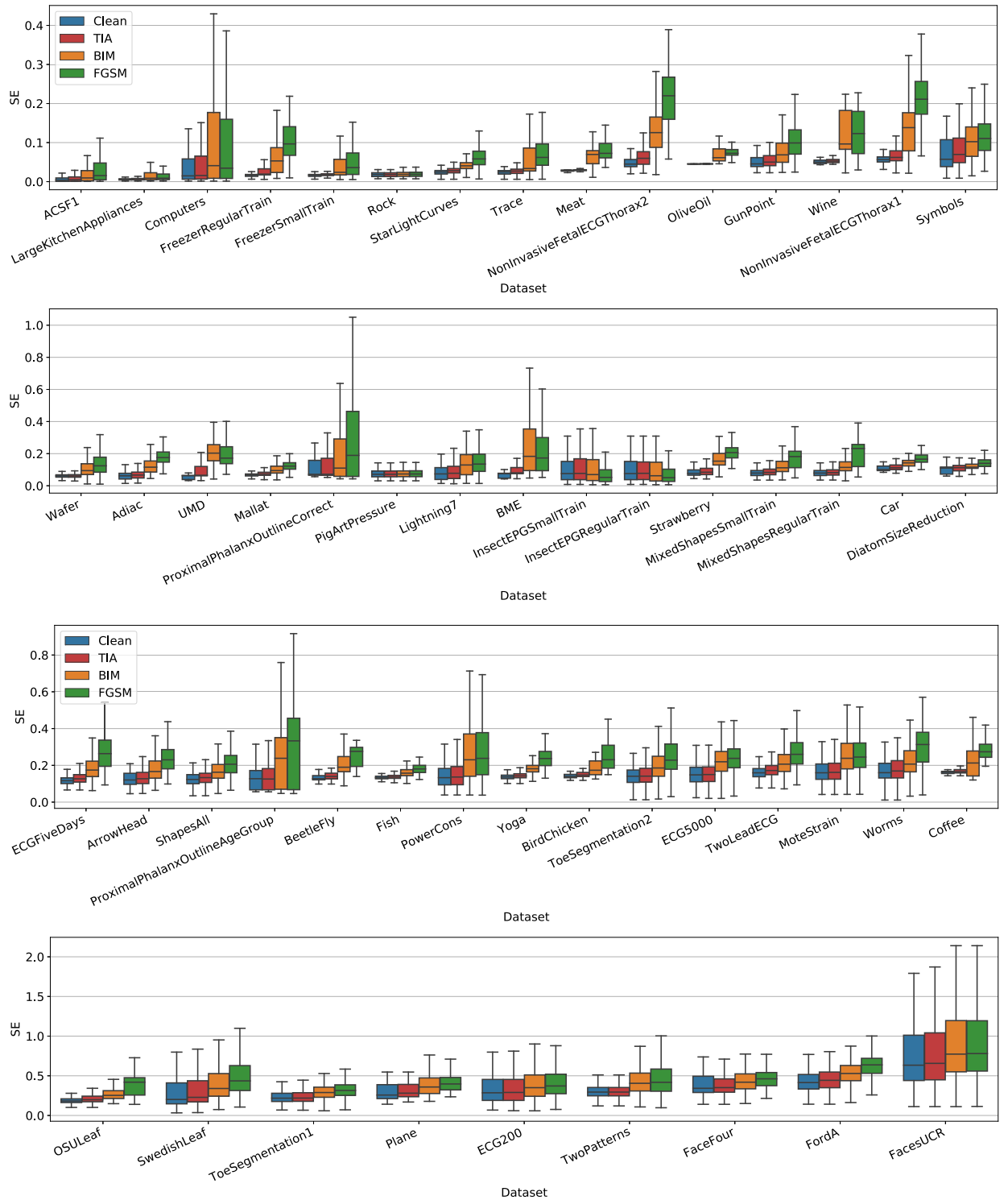

Fig. 6. Boxplots of the SE of each attack method for all datasets.

Table 6
All pairwise comparisons with TukeyHSD confidence intervals.

group1	group2	meandiff	p-adj	lower	upper	reject
FGSM	PGD	1.1	0.9	-6.9	9.0	False
FGSM	TIA	35.1	0.001	27.18	43.0	True
PGD	TIA	35.0	0.001	26.0	42.0	True

Table 7
Ablation study for trend-adaptive loss and gradient-based interval selection. In each cell, the average detection rate (lower is better), and the number of datasets with a detection rate under the median value (higher is better) are summarized. Both methods are effective for generating a hardly detectable adversarial time series.

Method	FC	CNN	LSTM
BIM	0.87 (10)	0.81 (24)	0.78 (26)
(+) Trend-adaptive loss	0.67 (38)	0.69 (29)	0.66 (40)
(+) Interval selection	0.64 (44)	0.65 (45)	0.63 (48)
TIA	**0.62 (47)**	**0.64 (48)**	**0.61 (51)**

original ones, while the interquartile ranges of existing methods do not overlap with the interquartile ranges of the original examples. TIA is more likely to preserve the chaos-theoretic measures than existing attack methods.

Similarly, the same tendency is observed for the DFA in Fig. 7. TIA shows similar boxplots of DFA to the original time series (blue). Specifically, for the Fish, Meat, and OliveOil datasets, the boxplot of TIA lies within the same level as the original ones, while the interquartile ranges of existing methods do not overlap with the interquartile ranges of the original examples. Although the discrepancies are relatively minor for datasets with high DFA values, this is attributed to the inherently chaotic nature of those time series, which are more effectively distinguished by SE. Thus, the results demonstrate that TIA successfully preserves the time series characteristics, whereas existing methods exhibit substantial deviations from the original values.

6.2. Selecting sensitive intervals

In this subsection, we demonstrate that TIA extracts sensitive intervals from the original time series during the attack process. To verify the number of perturbed intervals, we plot a cumulative density function of successful adversarial examples with respect to their perturbed points. If a higher cumulative density is obtained with lower perturbed points, it can be considered a better attack in terms of efficiency. As shown in Fig. 8, TIA is closer to the upper left corner, indicating that TIA successfully decreases the number of perturbations. By contrast, BIM and FGSM used more than 80% of the perturbation points in most examples. Thus, it is experimentally proven that TIA alleviates this problem through interval selection, whereas existing methods yield excessive perturbations, as observed in Section 3.3.

The effectiveness of the selected intervals can be verified in Fig. 9. In this experiment, we consider a naive method that selects the interval randomly, utilizing the same number of perturbed points as TIA. For example, if TIA uses only 50% of the interval on average for a particular dataset, the naive method randomly uses 50% of the adversarial perturbation of BIM. We call this naive method 'Random'. In Fig. 9, we visualize attack success rates of BIM, TIA, and a random method for whole datasets. The attack success rates of TIA (red) are similar to those of BIM (orange), which implies TIA shows negligible loss in its attack success rate, whereas it dramatically reduces the excessive perturbations, as shown in Fig. 8. However, a substantial performance gap exists between TIA and the random method. Specifically, the random method shows a lower attack success rate than TIA for all datasets. This result implies that TIA accurately finds the sensitive intervals to deceive the time series model.

6.3. Ablation study

For a deeper understanding of the effectiveness, we evaluate the effectiveness of each technique of TIA: trend-adaptive loss and gradient-based interval selection. Here, we investigate the effects of each technique in terms of the detection rate. In this experiment, to estimate the effect of each technique, we conduct the same experiment in Section 3.1 by adding each technique to BIM.

Table 7 demonstrates that both techniques are effective in decreasing the detection rate. Specifically, the trend-adaptive loss decreases the average detection rate by over 10% for all models. Similarly, the interval selection method further reduces detection rates and more than doubles the number of datasets exhibiting a detection rate below the median. Most importantly, when both techniques are combined, TIA achieves the best performance over diverse models.

Furthermore, in Fig. 10, we investigate the effect of window size. Here, we plot the adversarial time series generated by TIA with $w = 5$ and 13. A larger window size ($w = 13$) tends to perturb more points of the original time series compared to the smaller one ($w = 5$). Although a larger window size might create stronger adversarial examples, it becomes more detectable due to excessive perturbations, as discussed in Section 3.3. In Table 8, we conduct the same experiment as in Table 4 with varying window sizes in TIA. Indeed, increasing the window size does not improve, but rather decreases, the attack success rate under the defense system.

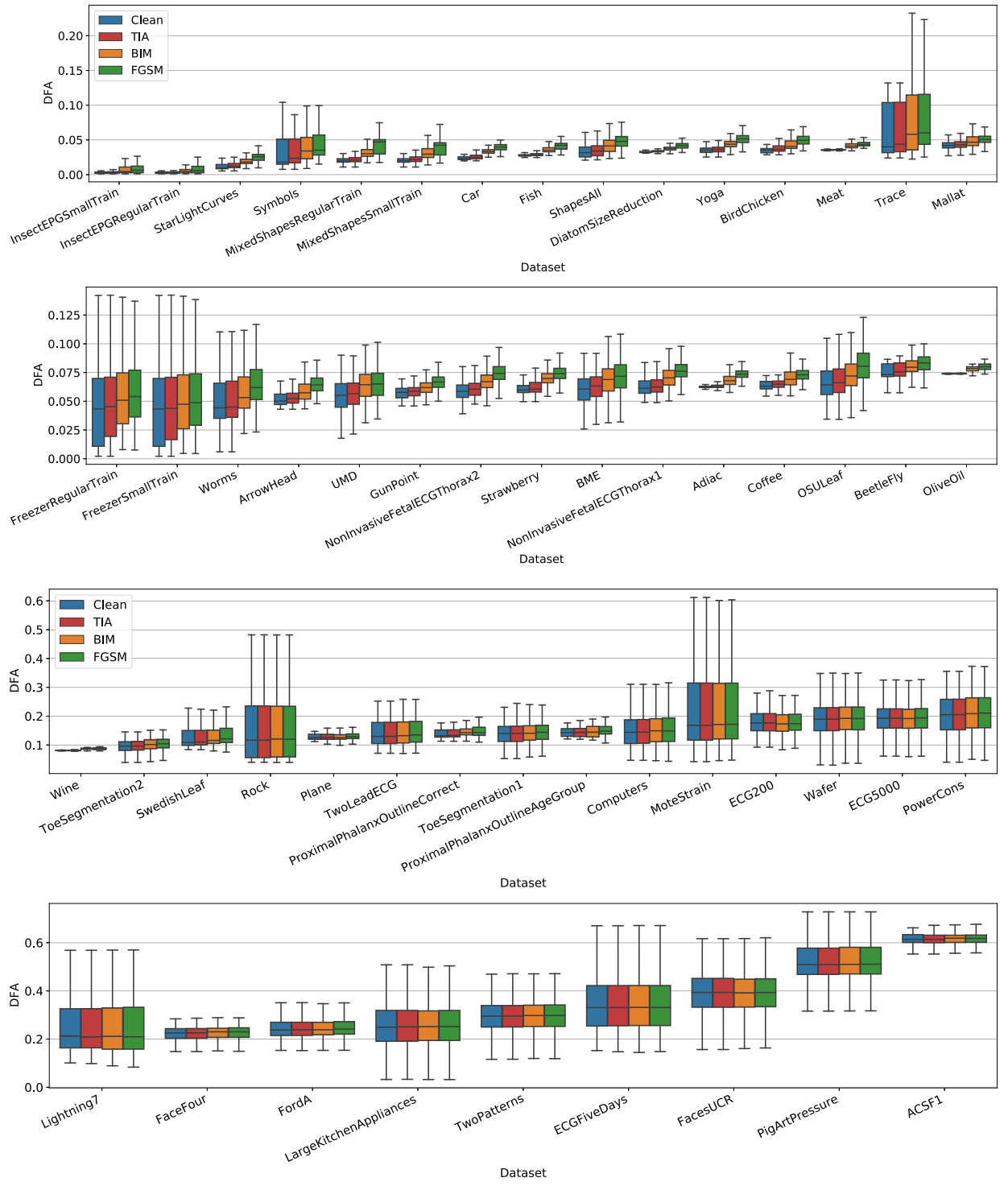

Fig. 7. Boxplots of the DFA of each attack method for all datasets.

Table 8
Sensitivity analysis on the window size w. Attack success rate under the defense system $\mathcal{A}_{Def}$ (%) of TIA for different window sizes.

Window size w	3	5	7	9	11	13
Attack success rate (%)	50.34	**52.86**	51.95	50.33	48.12	47.99

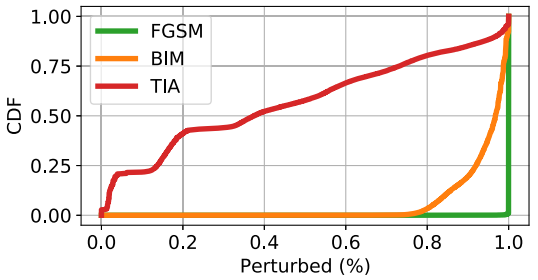

Fig. 8. Cumulative distribution function (CDF) of adversarial time series samples with respect to their perturbed points (%). In this figure, we only visualized the distribution of samples that succeeded in fooling the model f. The closer the line gets to the upper left corner, the harder to be detected.

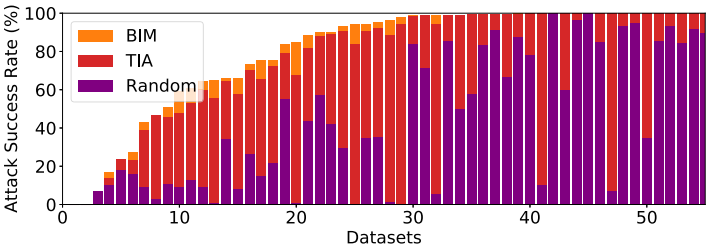

Fig. 9. Visualization of adversarial attack success ratio of BIM, TIA, and random interval selection. The y-axis represents the percentage of adversarial samples that succeeded in fooling the model.

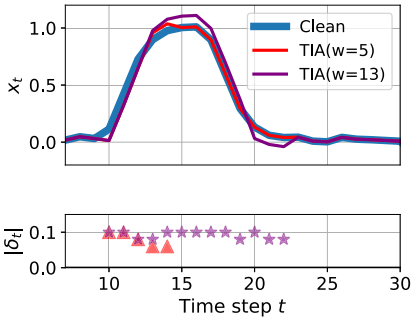

Fig. 10. Visualization of the adversarial time series $\boldsymbol{x}'$ generated by TIA with $w = 5$ and 13.

7. Conclusion

In this study, we analyzed the characteristics of easily detectable adversarial time series across different models and datasets. We found that excessive perturbations in adversarial time series significantly influence chaos-theoretical measures, thereby resulting in a high detection rate. Based on this observation, we proposed a novel adversarial attack, termed TIA, to generate adversarial time series that are difficult to detect by incorporating gradient-based interval selection and locally adaptive maximum perturbation. Our experimental results demonstrated that TIA can generate adversarial time series that maintain the characteristics of the original series while remaining difficult to detect. Based on these findings, we argue that the proposed method can serve as a benchmark for evaluating the robustness of adversarial defense mechanisms in the time series domain. We further hope that future work will expand upon the proposed approach to develop new adversarial attack and defense strategies, including those based on generative models and unsupervised learning.

CRediT authorship contribution statement

Hoki Kim: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Yunyoung Lee:** Writing – review & editing, Writing – original draft, Methodology, Investigation. **Woojin Lee:** Writing – review & editing, Writing – original draft, Methodology, Investigation. **Jaewook Lee:** Writing – review & editing, Writing – original draft, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (Ministry of Science and ICT) (No. RS-2024-00338859), the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (Ministry of Science and ICT) (No. RS-2022-II220984), and the Korea government (Ministry of Science and ICT) (No. RS-2025-02219688), Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation), and the Institute of Information & communications Technology Planning & Evaluation (IITP) under the artificial intelligence semiconductor support program to nurture the best talents (IITP-2024(2025)-RS-2023-00266605) grant funded by the Korea government (Ministry of Science and ICT).

Data availability

We used public datasets only.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.
[2] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6645–6649.
[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint, arXiv:1312.6199, 2013.
[4] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Adversarial attacks on deep neural networks for time series classification, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.
[5] F. Karim, S. Majumdar, H. Darabi, Adversarial attacks on time series, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.
[6] H. Kim, S. Lee, J. Lee, W. Lee, Y. Son, Evaluating practical adversarial robustness of fault diagnosis systems via spectrogram-aware ensemble method, Eng. Appl. Artif. Intell. (2024) 107980.
[7] M.G. Abdu-Aguye, W. Gomaa, Y. Makihara, Y. Yagi, Detecting adversarial attacks in time-series data, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 3092–3096.
[8] H. Li, J. Liu, Z. Yang, R.W. Liu, K. Wu, Y. Wan, Adaptively constrained dynamic time warping for time series classification and clustering, Inf. Sci. 534 (2020) 97–116.
[9] J. Tong, L. Xie, W. Yang, K. Zhang, J. Zhao, Enhancing time series forecasting: a hierarchical transformer with probabilistic decomposition representation, Inf. Sci. 647 (2023) 119410.
[10] E. Keogh, Efficiently finding arbitrarily scaled patterns in massive time series databases, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2003, pp. 253–265.
[11] H. Sakoe, Dynamic-programming approach to continuous speech recognition, in: 1971 Proc. The International Congress of Acoustics, Budapest, 1971.
[12] M. Müller, Dynamic time warping, in: Information Retrieval for Music and Motion, 2007, pp. 69–84.
[13] F. Petitjean, G. Forestier, G.I. Webb, A.E. Nicholson, Y. Chen, E. Keogh, Dynamic time warping averaging of time series allows faster and more accurate classification, in: 2014 IEEE International Conference on Data Mining, IEEE, 2014, pp. 470–479.
[14] R.J. Kate, Using dynamic time warping distances as features for improved time series classification, Data Min. Knowl. Discov. 30 (2016) 283–312.
[15] D. Gabor, Theory of communication. Part 1: the analysis of information, J. Inst. Electr. Eng., Part 3, Radio Commun. Eng. 93 (1946) 429–441.
[16] Y. Zheng, Q. Liu, E. Chen, Y. Ge, J.L. Zhao, Time series classification using multi-channels deep convolutional neural networks, in: International Conference on Web-Age Information Management, Springer, 2014, pp. 298–310.
[17] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, P. Loos, Time series classification using deep learning for process planning: a case from the process industry, Proc. Comput. Sci. 114 (2017) 242–249.
[18] J. Azar, A. Makhoul, R. Couturier, J. Demerjian, Robust iot time series classification with data compression and deep learning, Neurocomputing 398 (2020) 222–234.
[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, arXiv preprint, arXiv:1706.06083, 2017.
[20] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint, arXiv:1412.6572, 2014.
[21] A. Kurakin, I. Goodfellow, S. Bengio, et al., Adversarial examples in the physical world, 2016.
[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[23] H.A. Dau, A. Bagnall, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The ucr time series archive, IEEE/CAA J. Autom. Sin. 6 (2019) 1293–1305.
[24] J.B. Kruskal, Multidimensional scaling, Sage 11 (1978).
[25] M. Antal, K. Buza, N. Fejer, Sapiagent: a bot based on deep learning to generate human-like mouse trajectories, IEEE Access 9 (2021) 124396–124408.
[26] G.R. Mode, K.A. Hoque, Adversarial examples in deep learning for multivariate time series regression, in: 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), IEEE, 2020, pp. 1–10.
[27] R. Dang-Nhu, G. Singh, P. Bielik, M. Vechev, Adversarial attacks on probabilistic autoregressive forecasting models, in: International Conference on Machine Learning, PMLR, 2020, pp. 2356–2365.
[28] S. Harford, F. Karim, H. Darabi, Generating adversarial samples on multivariate time series using variational autoencoders, IEEE/CAA J. Autom. Sin. 8 (2021) 1523–1538.
[29] H. Kim, W. Lee, S. Lee, J. Lee, Bridged adversarial training, Neural Netw. 167 (2023) 266–282.
[30] J. Chen, H. Zheng, W. Shangguan, L. Liu, S. Ji, Act-detector: adaptive channel transformation-based light-weighted detector for adversarial attacks, Inf. Sci. 564 (2021) 163–192.
[31] C.-K. Peng, S.V. Buldyrev, S. Havlin, M. Simons, H.E. Stanley, A.L. Goldberger, Mosaic organization of DNA nucleotides, Phys. Rev. E 49 (1994) 1685.

[32] J.S. Richman, J.R. Moorman, Physiological time-series analysis using approximate entropy and sample entropy, Am. J. Physiol., Heart Circ. Physiol. 278 (2000) H2039–H2049.
[33] J.S. Richman, D.E. Lake, J.R. Moorman, Sample entropy, Methods Enzymol. 384 (2004) 172–184.
[34] N. Carlini, D. Wagner, Adversarial examples are not easily detected: bypassing ten detection methods, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017, pp. 3–14.
[35] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: a strong baseline, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1578–1585.
[36] H.A. Dau, E. Keogh, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, Hexagon-ML, The ucr time series classification archive, 2018.
[37] M.J. Cannon, D.B. Percival, D.C. Caccia, G.M. Raymond, J.B. Bassingthwaighte, Evaluating scaled windowed variance methods for estimating the Hurst coefficient of time series, Phys. A, Stat. Mech. Appl. 241 (1997) 606–626.
[38] J. Su, D.V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks, IEEE Trans. Evol. Comput. 23 (2019) 828–841.
[39] A. Modas, S.-M. Moosavi-Dezfooli, P. Frossard, Sparsefool: a few pixels make a big difference, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9087–9096.
[40] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[41] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618–626.
[42] R. Assaf, A. Schumann, Explainable deep neural networks for multivariate time series predictions, in: IJCAI, 2019, pp. 6488–6490.
[43] F. Karim, S. Majumdar, H. Darabi, S. Chen, Lstm fully convolutional networks for time series classification, IEEE Access 6 (2017) 1662–1669.