ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240644

TSFool: Crafting Highly-Imperceptible Adversarial Time Series Through Multi-Objective Attack

Yanyun Wang[a], Dehui Du[b,*], Haibo Hu[a,*], Zi Liang[a] and Yuanhao Liu[b]

[a]Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University
[b]Software Engineering Institute, East China Normal University

Abstract. Recent years have witnessed the success of recurrent neural network (RNN) models in time series classification (TSC). However, neural networks (NNs) are vulnerable to adversarial samples, which cause real-life adversarial attacks that undermine the robustness of AI models. To date, most existing attacks target at feed-forward NNs and image recognition tasks, but they cannot perform well on RNN-based TSC. This is due to the cyclical computation of RNN, which prevents direct model differentiation. In addition, the high visual sensitivity of time series to perturbations also poses challenges to local objective optimization of adversarial samples. In this paper, we propose an efficient method called TSFool to craft highly-imperceptible adversarial time series for RNN-based TSC. The core idea is a new global optimization objective known as "Camouflage Coefficient" that captures the imperceptibility of adversarial samples from the class distribution. Based on this, we reduce the adversarial attack problem to a multi-objective optimization problem that enhances the perturbation quality. Furthermore, to speed up the optimization process, we propose to use a representation model for RNN to capture deeply embedded vulnerable samples whose features deviate from the latent manifold. Experiments on 11 UCR and UEA datasets showcase that TSFool significantly outperforms six white-box and three black-box benchmark attacks in terms of effectiveness, efficiency and imperceptibility from various perspectives including standard measure, human study and real-world defense.

1 Introduction

NNs are vulnerable to adversarial attacks [38], which means imperceptible perturbations added to the input can cause the output to change significantly [22]. A rich body of adversarial attacks has been investigated to generate adversarial samples that can enhance the robustness of the models through adversarial training. For instance, gradient-based attacks [8, 21, 25, 29, 31, 34] have achieved impressive performance on feed-forward NN classifiers and image recognition tasks. Nevertheless, those gradient-based adversarial attacks cannot perform well in RNN where the unique time recurrent structure of RNN prevents differentiation on it [35]. Furthermore, existing attacks always adopt a local optimization objective to minimize the perturbation amount for every single sample. However, as shown in Figure 1, time series are far more visually sensitive to perturbation than image data, undermining the effectiveness of these attacks in RNN models for time series.

* Corresponding Authors. dhdu@sei.ecnu.edu.cn; haibo.hu@polyu.edu.hk.

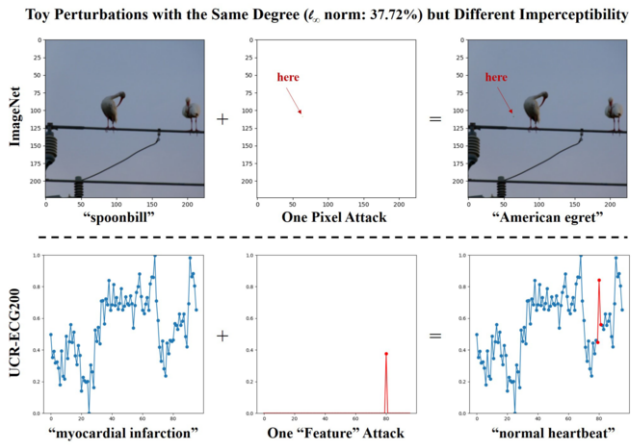

Figure 1: The figure shows that image and time series data have significantly different visual sensitivities to perturbations of the same level. The upper line is an example of One Pixel Attack [37] against a pre-trained ResNet-18 [30] on ImageNet [14]. Although the ℓ_∞ norm of the generated perturbation is up to 37.72%, it is still hard to be noticed by human eyes. On the contrary, as the lower line, a similar attack (*i.e.* also merely perturb one feature by 37.72% under ℓ_∞ norm) against one of our experimental RNN classifiers on UCR-ECG200 time series dataset [13] is more than visible. This makes it more difficult to craft imperceptible adversarial time series than images. Please notice that this is just a toy case to reveal the current gaps, instead of the illustration of our proposed approach.

Due to these challenges, despite the popularity of TSC and RNN as its solution [15, 16, 39], to date, there is yet not much effective study on crafting qualified adversarial samples for RNN-based TSC [16, 19, 24]. A few works focus on making RNN completely differentiable by *cyclical computational graph unfolding* [32, 35], which turns out to be inefficient and hard to stably scale [7, 20, 42]. Black-box attacks by model querying [1, 6, 9, 37] or adversarial transferability [33, 36] are either inapplicable to RNN-based TSC or very limited in effectiveness.

In this paper, we propose an efficient method named TSFool to craft highly-imperceptible adversarial samples for RNN-based TSC. With an argument that local optimal perturbation under the conventional objective does not always lead to imperceptible adversarial samples, we propose a novel global optimization objective named

Camouflage Coefficient, and add it to reduce the adversarial attack problem to a multi-objective optimization problem. In this way, we can take the relative position between adversarial samples and class clusters into consideration, to measure the imperceptibility of adversarial samples from the perspective of class distribution. Since the full gradient information of an RNN is not directly available, to efficiently approximate the optimization solution, we introduce a representation model built only upon the classifier's outputs. It can fit the manifold hyperplane of a classifier but distinguish samples by their features like humans, and capture deeply embedded vulnerable samples whose features deviate from the latent manifold as guidance. Then we can pick target samples to craft perturbation in the direction of their interpolation, while imperceptibly crossing the classification hyperplane.

With six white-box and three black-box adversarial attacks from basic to state-of-the-art ones as the benchmarks, we evaluate our approach on 11 univariate or multivariate time series datasets respectively from the public UCR [13] and UEA [4] archives. The results demonstrate that TSFool has significant advantages in effectiveness, efficiency and imperceptibility. In addition, extensive experiments on different hyper-parameters and settings confirm that our results are fair and of general significance. Beyond standard measures, we also conduct two human studies verifying the ability of Camouflage Coefficient to capture the real-world imperceptibility of adversarial samples, as well as implement four anomaly detection methods showing the imperceptibility of TSFool under real-world defense. Our main contributions are summarized as follows:

- By exploring the visual sensitivity of time series data, for the first time, we point out the bias of the conventional local optimization objective of adversarial attack;
- We propose a novel optimization objective named "Camouflage Coefficient" to enhance the global imperceptibility of adversarial samples, with which we reduce the adversarial attack problem to a multi-objective optimization problem; and
- We propose a general methodology based on *Manifold Hypothesis* to solve the new optimization problem, and accordingly realize TSFool, the first method to our best knowledge, for RNN-based TSC, to craft real-world imperceptible adversarial time series.

The rest of this paper is organized as follows. Section 2 reviews the related studies and explains the current gaps. A general methodology outlining our ideas is proposed in Section 3, followed by Section 4 to specifically realize the proposed TSFool in detail. Section 5 evaluates the performance of TSFool. Section 6 provides relevant discussions and Section 7 concludes this paper. Section A, B and C in the Appendix[1] respectively supplement more details about the approach, evaluation and discussion.

2 Background and Related Work

2.1 RNN-based Time Series Classification

TSC is an important and challenging problem in modern data mining [15, 23], with a variety of real-world applications including health care [28], stock price prediction [40] and food safety inspection [17]. Time series data consists of sampled data points taken from a continuous process over time [26]. It can be defined as a sequence $X \in \mathbb{R}^{T\times D}$, where T is the sequence length, also known as the number of time steps, and D is the feature dimension of every single data point $x_t \in X$ ($t \in [1, T]$), based on which there are two categories of time series namely univariate series ($D = 1$) and multivariate series ($D > 1$) [15]. The key difficulty of TSC is to recognize the complex temporal pattern and characterize the temporal semantic information encoded in X [15]. With the recurrent structure specifically designed for learning key temporal patterns, RNN has become one of the most state-of-the-art models for TSC [39]. Given an RNN classifier $\mathcal{N} : \mathcal{X} \to \mathcal{Y}$ with hidden state $h_t \in \mathbb{R}^H$ and non-linear recurrent function $h_t = \mathcal{N}(h_{t-1}, x_t)$, for any time series data X, it takes each of the $x_t \in X$ as the input at time step t in order and encodes the key information by update h_t accordingly. Then the final state h_T characterizes the whole series for classification.

2.2 Adversarial Sample and Adversarial Attack

The concept of adversarial attack is introduced by Szegedy et al. [38], in which an adversarial sample $\vec{x}^*$ crafted from a legitimate sample $\vec{x}$ is defined by an optimization problem:

$$\vec{x}^* = \vec{x} + \delta_{\vec{x}} = \vec{x} + \min \|\vec{z}\| \text{ s.t. } f(\vec{x} + \vec{z}) \neq f(\vec{x}), \tag{1}$$

where $f : \mathbb{R}^n \to \vec{y}$ is the target classifier and $\delta_{\vec{x}}$ is the smallest perturbation according to a norm appropriate for the input domain. Since exactly solving this problem by an optimization method is time-consuming [31] and not always possible, especially in NNs with complex non-convexity and non-linearity [35], the common practice is to find the approximative solutions to estimate adversarial samples.

The most commonly used adversarial attacks to date are gradient-based methods under white-box setting [27], which means all the information of the target classifier is available and the perturbation is guided by the differentiating functions defined over model structure and parameters. For instance, the *fast gradient sign method* (FGSM) [21] implements the perturbation according to the gradient of cost function $\mathcal{L}$ with respect to the input $\vec{x}$:

$$\delta_{\vec{x}} = \varepsilon \operatorname{sign}(\nabla_{\vec{x}} \mathcal{L}(f, \vec{x}, \vec{y})), \tag{2}$$

where ε denotes the magnitude of the perturbation. The *Jacobian-based saliency map attack* (JSMA) [34] further introduces forward derivative to quantitatively capture how a specific input component modifies the output. Then DeepFool [31] proposes a greedy idea to determine the perturbation direction of specific samples according to their closest classification hyperplane. As one of the state-of-the-art benchmarks to date [3], the *projected gradient descent attack* (PGD) [29] can be viewed as a variant of FGSM. It crafts samples by getting the perturbation as FGSM and projecting it to the ϵ-ball of input iteratively. To be specific, given $\vec{x}^t$ as the intermediate input at step t, PGD updates as:

$$\vec{x}^{t+1} = \Pi_\epsilon \left\{ \vec{x}^t + \varepsilon \cdot \operatorname{sign}(\nabla_{\vec{x}} \mathcal{L}(f, \vec{x}^t, \vec{y})),\ \vec{x} \right\}. \tag{3}$$

In deep learning, it is sometimes [8] referred to as *basic iterative method* (BIM) [25]. Another state-of-the-art benchmark, C&W [8], no longer uses the constraint to ensure small perturbation but formulates a regularization optimization for it. Given y_0 as the true label and $f(\vec{x}_i)_{y_k}$ as the prediction score of label y_k for the candidate input $\vec{x}_i$, then:

$$\vec{x}^* = \underset{\vec{x}_i}{\operatorname{argmin}} \{\|\vec{x}_i - \vec{x}\|_2^2 + \alpha \cdot \max\{f(\vec{x}_i)_{y_0} - \max_{y_j \neq y_0} f(\vec{x}_i)_{y_j}, 0\}\}, \tag{4}$$

where $\alpha > 0$ controls the trade-off between small distortion and attack success rate. Finally, Auto-Attack [12] forms a parameter-

[1] The Appendix is available at https://arxiv.org/abs/2209.06388.

free and user-independent ensemble of attacks for frequent pitfalls in practice like improper tuning of hyper-parameters and gradient obfuscation or masking.

Nevertheless, model information including the gradient is not always available in practice [16, 27], so some black-box methods have been proposed, which can be roughly classified into two categories. The first one relies on the predictions returned by querying the target model. For instance, the Boundary Attack [6] and its improvement HopSkipJump [9] search for adversarial samples along the decision boundary. While the second one called Transfer Attack [17, 36] trains a substitute model similar to the target model in performance, then attacks it instead to generate adversarial samples, and relies on their adversarial transferability to fool the target model [33].

2.3 Challenges for RNN and Time Series

While existing adversarial attacks achieve great success on feed-forward NNs and image data, such success has not carried over to RNN-based TSC. There are two main reasons for this dilemma. Firstly, as Figure 1, image data can tolerate respectively large real-world perturbations without being noticed by humans, while time series are so visually sensitive to perturbations that it is more difficult to ensure their imperceptibility [39]. This exposes the drawback of existing methods in the control of perturbation on different data. It is also surprising that the only specialized measure for imperceptibility of adversarial time series is the number of time steps perturbed [35], which is not sufficient.

Secondly, the presence of cyclical computations in RNN architecture prevents direct model differentiation [35], which means most of the gradient information is no longer directly available through the *chain rule*. This problem seriously challenges the effectiveness of all the gradient-based methods. An intuitive solution is to make the model differentiable through *cyclical computational graph unfolding* to fit the gradient-based methods [32, 35]. However, as the length of the time series is considerable in most real-world cases, the practical efficiency and scalability of the unfolding computation cannot be guaranteed [7, 20, 42]. Another idea is to completely ignore the model knowledge given and view it as a black-box setting, to avoid the difficulty of specialized design for RNN. Nevertheless, a majority of the query-based black-box attacks like One Pixel Attack [37] and Square Attack [1] can only work for image input. Although Boundary Attack and HopSkipJump are not subject to this limitation, both of them are extremely time-consuming because of the random-walking. On the other hand, Transfer Attack tends to achieve reasonable time and small perturbations, but unstable effectiveness. All in all, the problems brought by the perturbation sensitivity of time series data and the recurrent structure of RNN model remain to be further explored.

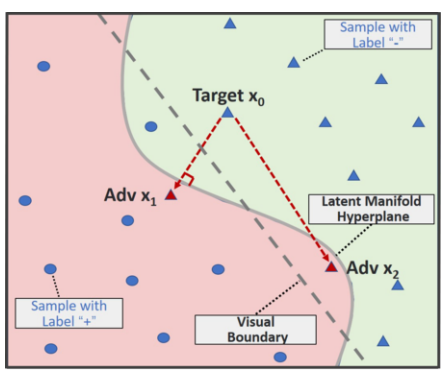

Figure 2: An intuitive instance that the minimal perturbation (i.e. $\|\vec{x}_1 - \vec{x}_0\|$) is not necessarily the most imperceptible one from the global perspective. The visual boundary and latent manifold hyperplane are different classification boundaries respectively from human eyes and the learned model.

3 Methodology

In this section, we propose some novel ideas to build a general methodology for the crafting of highly-imperceptible adversarial samples. Notice that although these ideas are motivated and inspired by the problems exposed by RNN-based TSC, their underlying principles are not limited to this task. So to facilitate further studies in a larger range, we organize them here independently before specifically realizing them by the proposed approach in Section 4.

3.1 Rethinking Imperceptible Adversarial Sample

The sensitivity of time series data prompts us to rethink the imperceptibility of perturbation. Firstly, existing methods always implement perturbations to all of the test data, while according to the specific class distribution, there must be individuals among them that are easier or harder to be perturbed respectively. A previous work has proven that a natural data point closer to (or farther from) the class boundary is less (or more) robust [41]. So without taking this difference into account and picking target samples appropriately, it is almost impossible for the attack method to stably control the perturbation amount by itself. On the contrary, it has to heavily depend on the specific dataset.

Secondly, we argue that the approximation to the optimal perturbation $\delta_{\vec{x}}$ in Equation (1) does not always lead to a highly-imperceptible adversarial attack, while most of the existing methods view it as the basic target. This is because when we jump out of a single sample and consider from the perspective of class distribution, it can be found that even the minimal perturbation is not necessarily the most imperceptible one. For instance, in Figure 2, the adversarial sample $\vec{x}_1$ is generated from $\vec{x}_0$ through a minimal perturbation that crosses the closest classification hyperplane, while another adversarial sample $\vec{x}_2$ is obviously more dangerous as it is semantically closer to and even belong to the original class.

These points inspired us to propose a novel definition of the imperceptibility of adversarial samples from the global perspective. Given a k-class classification task and the label $i, j \in \{0, 1, ..., k-1\}$, $\mathcal{X}_i$ is the set of all the samples belonging to class i in the test set. For adversarial sample $\vec{x}^*$ from $\vec{x} \in \mathcal{X}_i$, which is wrongly predicted as $f(\vec{x}^*) = j\ (j \neq i)$, we define its **Camouflage Coefficient** (**CC**) $\mathcal{C}(\vec{x}^*)$ as:

$$\mathcal{C}(\vec{x}^*) = \frac{\|\vec{x}^* - \vec{m}_i\|/d_i}{\|\vec{x}^* - \vec{m}_j\|/d_j}, \tag{5}$$

where m_i is the center of mass of class i which is also built in the form of a legal sample:

$$\vec{m}_i = \frac{1}{|\mathcal{X}_i|} \sum_{\vec{x}' \in \mathcal{X}_i} \vec{x}', \tag{6}$$

and d_i is the average norm distance between m_i and all the samples in $\mathcal{X}_i$, which is used to eliminate the potential bias from the different cluster sizes of the two classes:

$$d_i = \frac{1}{|\mathcal{X}_i|} \sum_{\vec{x}' \in \mathcal{X}_i} \|\vec{x}' - \vec{m}_i\|. \tag{7}$$

As the saying goes, "*the best place to hide a leaf is the woods*". The CC represents the proportion of the relative distance between adversarial sample $\vec{x}^*$ and the original class to the relative distance between $\vec{x}^*$ and the class to which it is misclassified. As a result, it can reveal to what extent an adversarial sample can "hide" in the original class without being noticed. The smaller the value, the higher the global imperceptibility of the adversarial sample. And if the value exceeds one, the attack somewhat fails. Because in this case, the misclassification of the adversarial sample is no longer surprising as it is already semantically closer to the samples in that wrong class instead of the original class. In our approach, we introduce CC as another optimization objective along with Equation (1) to craft adversarial samples considering local and global imperceptibility at the same time.

3.2 Attack through Multi-objective Perturbation

After adding the Camouflage Coefficient, the adversarial attack becomes a multi-objective optimization problem. Just as the existing methods, considering the efficiency, we do not solve it directly, but find the approximate solution in a logical and practical way. Separately speaking, to optimize Equation (5), we can just make $\vec{x}^*$ closer to the $\vec{m}_i$, which also means farther away from the $\vec{m}_j$ in general. And to approximate the objective in Equation (1), perturbing in the direction of the closest classification hyperplane is proven to be a successful approach in DeepFool. So as a compromise to take them into account together, a reasonable idea is to cross a hyperplane that is relatively near the target sample $\vec{x}$ on an appropriate place that is relatively close to the $\vec{m}_i$.

To realize this idea in practice, we can find a sample $\vec{x}_v$ from $\mathcal{X}_i$ which is misclassified to class j but deeply embedded in class i as a guide, and accordingly pick a correctly classified sample that is the closest to $\vec{x}_v$ in class i as the target $\vec{x}$. Then when we perturb $\vec{x}$ in the direction of $\vec{x}_v$ to cross the classification hyperplane between them, we can not only acquire a considerable value of CC as the $\vec{x}_v$ itself is a sample embedded closer to $\vec{m}_i$ than to $\vec{m}_j$, but also expect the hyperplane is relatively a close one to $\vec{x}$ as this sample is the closest to $\vec{x}_v$ among class i. In this way, the $\vec{x}$ is not perturbed to be adversarial along the shortest path, but the more imperceptible one under the multi-objective definition. We define the $\vec{x}_v$ and $\vec{x}$ respectively as **vulnerable negative sample (VNS)** and **target positive sample (TPS)**, with which the perturbation can be denoted as:

$$\delta_{\vec{x}} = \lambda_\varepsilon \|\vec{x}_v - \vec{x}\| + \vec{x}_\varepsilon, \tag{8}$$

where $\lambda_\varepsilon \|\vec{x}_v - \vec{x}\|$ is the maximum interpolation that makes $\vec{x}$ approach $\vec{x}_v$ under a given step size ε without changing its prediction result, and $\vec{x}_\varepsilon$ is a micro random noise under the same ε added to cross the classification hyperplane.

3.3 Capturing Vulnerable Sample by Manifold

Now the only problem left is how to capture the VNS. Our idea to solve it comes from the *manifold hypothesis*. It is one of the influential explanations for the effectiveness of NNs, which holds that many high-dimensional real-world data are actually distributed along low-dimensional manifolds embedded in the high-dimensional space [18]. This explains why NNs can find latent key features as complex functions of a large number of original features in the data. And it is through learning the manifold of the latent key features of training data that NNs can realize manifold interpolation between input samples and generate accurate predictions of unseen samples [11]. So what matters in NN classification is how to distinguish the latent manifolds instead of the original features of samples with different labels.

However, due to the limitations of sampling technologies and human cognition, practical label construction and model evaluation have to rely on specific high-dimensional data features. Accordingly, a possible explanation for the existence of adversarial samples is that the features of input data cannot always visually and semantically reflect the latent manifold, which makes it possible for the samples considered to be similar in features to have dramatically different latent manifolds. As a consequence, even a small perturbation in human eyes added to a correctly predicted sample may completely change the perception of NN to its latent manifold, and result in a significantly different prediction.

So if there is a representation model that can imitate the mechanism of a specific NN classifier to predict input data, but distinguish different inputs by their features in the original high-dimensional space just like a human, then it can be introduced to capture deeply embedded vulnerable samples whose features deviate from the latent manifold. Specifically, when the prediction of representation model is correct while that of NN classifier is wrong, it means that compared with other samples that are similar in features and belong to the same class in fact, the current sample is perceived by NN at a different manifold cluster, and as a result, wrongly predicted. Such a sample well-meets the definition of VNS.

4 Proposed Approach

Based on the ideas and methodology above, we propose an efficient approach named TSFool to craft highly-imperceptible adversarial time series for RNN-based TSC. The roadmap of TSFool is shown in Figure 3.

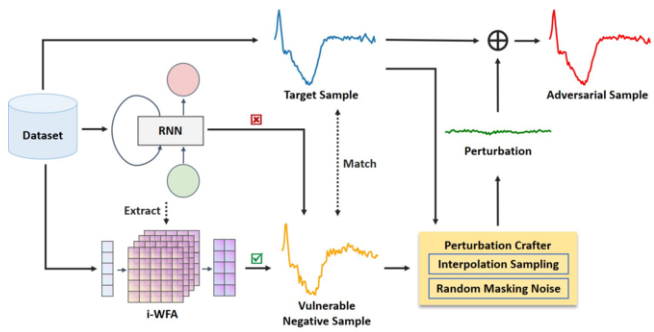

Figure 3: With the representation model i-WFA extracted from the target RNN classifier, the vulnerable samples with features deviating from the latent manifold can be captured according to their predictions. Then the target samples are specifically matched and then perturbed through two steps namely the interpolation sampling and adding random masking noise.

4.1 Extraction of Representation Model

To model the special recurrent computation of RNN, we introduce *weighted finite automaton* (WFA) [43] that can imitate the execution of RNN based on its hidden state updated at each time step. Nevertheless, WFA also relies on existing clustering methods like k-*means* to abstract input data, while an important requirement of the representation model is to distinguish different inputs directly by original features as a human. So we improve WFA by changing its input from

data clusters to domain intervals. Specifically, to determine the interval size, for each of the input samples, we calculate the average ℓ_2 distance between its features in adjacent time steps, and then reduce the result by an order of magnitude as an "imperceptible distance". In this way, when the size of the input interval is smaller than this distance, it is almost impossible for features that belong to different input clusters in the original WFA to be assigned to the same interval, so as to ensure that the domain interval is a reasonable substitute of the input cluster. Accordingly, we propose an ideal representation model as below.

An **Intervalized Weighted Finite Automaton (i-WFA)** extracted from a k-class RNN classifier $\mathcal{N}$ is defined as a tuple $\mathcal{A} = (Z, S, \mathcal{I}, (E_\zeta)_{\zeta \in Z}, \mathcal{T})$, where Z is a finite set of intervals that covers the whole input domain, S is a finite set of states abstracted from the hidden states of $\mathcal{N}$, $\mathcal{I}$ is the initial state vector of dimension $|S|$, and $\mathcal{T}$ is the probabilistic output matrix with size $|S| \times k$. For each interval $\zeta \in Z$, the corresponding probabilistic transfer matrix E_ζ with size $|S| \times |S|$ is built according to the hidden transfer of all the samples having features that fall into this interval. In the execution of i-WFA, the $\mathcal{I}$ is iteratively updated to represent the current state with the transfer under different inputs imitated by the corresponding E_ζ at each time step, followed by the $\mathcal{T}$ to imitate the computation of probabilistic predictions finally. The detailed algorithm of i-WFA establishment and an instance of i-WFA are respectively provided in Section A.1 and Section A.2 of the Appendix.[1]

4.2 Craft Imperceptible Adversarial Time Series

Capturing Vulnerable Negative Sample Building an i-WFA as the representation model of the target RNN classifier, we make a comparison between them to capture the VNS. As shown in Algorithm 1, we get the prediction results in the test set respectively from the RNN classifier and i-WFA (lines 3-4), followed by the comparison between them in function *CaptureVNS* (line 5).

Interpolation Sampling As argued in Section 3, different from the existing methods that attack all the test samples, TSFool picks specific TPS according to the VNS captured. Given that a pair of VNS and TPS are similar in features and with the same label, while they are predicted differently by the classifier, there must be a hyperplane to be discovered between them that divides the latent manifold of the two classes. So we can approximate that hyperplane by the interpolation of their features as the first part of the perturbation (i.e. $\lambda_\varepsilon \|\vec{x}_v - \vec{x}\|$). In Algorithm 1, we pick TPS for each of the VNS by the function *PickTPS* (lines 6-7). Then with the function *UpdateSamplingRange* which updates the sampling range each turn by the two currently sampled adjacent examples with different prediction results respectively the same as VNS and TPS (lines 10-11), we do average sampling (line 9) to approximate the maximum feature interpolation iteratively until the step size of sampling is smaller than a given limitation of noise (lines 12-13).

Adding Random Masking Noise Since the first part of perturbation already approaches the hyperplane of the latent manifold at a given step size, a micro noise $\vec{x}_\varepsilon$ of the same size can be added to easily cross the hyperplane. In other words, the adversarial property of the perturbed samples is guaranteed by the first part of perturbation, and the only requirement for $\vec{x}_\varepsilon$ is that it must be micro enough that the semantical consistency between VNS and TPS can also be inherited by the perturbed sample. We implement random masking to generate $\vec{x}_\varepsilon$ in batches, and add it to finish the attack (line 16). Specifically, we first build a complete noise vector in the direction of interpolation, with the specific noise amount ϵ calculated by *ImperceptibleNoise* under the constraint mentioned before. Then we randomly mask elements at some time steps of the noise vector (i.e., set value 0) according to a given probability p. For each pair of VNS and TPS, this process is run n times to craft a batch of adversarial samples, also with size n.

Algorithm 1 Craft Imperceptible Adversarial Time Series

Input: RNN $\mathcal{N} = (X, Y, H, f, g)$, i-WFA $\mathcal{A} = (Z, S, \mathcal{I}, (E_\zeta)_{\zeta \in Z}, \mathcal{T})$, Test Set $\mathcal{X}$, Test Labels $\mathcal{Y}$, Hyper-parameters ε, n and p
Output: Adversarial Time Series Set $\mathcal{X}_{adv}$

1: Initialize $\mathcal{X}_{adv} \leftarrow []$
2: $\epsilon \leftarrow$ *ImperceptibleNoise*(ε)
3: $Y_\mathcal{N} \leftarrow \mathcal{N}(\mathcal{X})$
4: $Y_\mathcal{A} \leftarrow \mathcal{A}(\mathcal{X})$
5: $X_v \leftarrow$ *CaptureVNS*$(\mathcal{X}, \mathcal{Y}, Y_\mathcal{N}, Y_\mathcal{A})$
6: **while** $x_{neg} \in X_v$ **do**
7: $\quad x_{pos} \leftarrow$ *PickTPS*$(\mathcal{X}, x_{neg})$
8: $\quad$ **while** x_m **not exist do**
9: $\quad\quad X_s \leftarrow$ *InterpolationSampling*(x_{neg}, x_{pos})
10: $\quad\quad Y_s \leftarrow \mathcal{N}(X_s)$
11: $\quad\quad x_{neg}, x_{pos} \leftarrow$ *UpdateSamplingRange*(X_s, Y_s)
12: $\quad\quad$ **if** $\|x_{pos} - x_{neg}\| < \epsilon$ **then**
13: $\quad\quad\quad x_m \leftarrow x_{pos}$
14: $\quad\quad$ **end if**
15: $\quad$ **end while**
16: $\quad X_{adv} \leftarrow$ *AddRandomMaskingNoise*(x_m, ϵ, n, p)
17: $\quad \mathcal{X}_{adv}$.*append*(X_{adv})
18: **end while**
19: **return** $\mathcal{X}_{adv}$

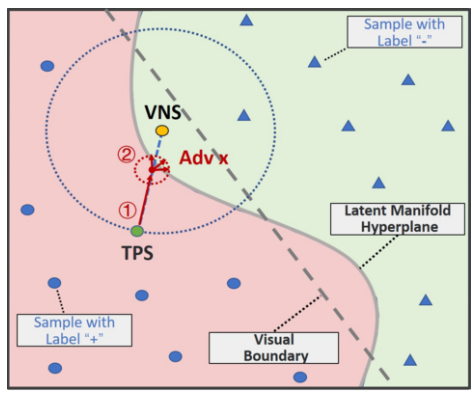

Figure 4: An intuitive illustration for the crafting of highly-imperceptible adversarial time series. The ① and ② correspond to the two parts of perturbation, respectively from the interpolation sampling and random masking noise.

5 Evaluation

In this section, with 10 univariate and one multivariate time series datasets respectively derived from the public UCR [13] and UEA [4] time series archives, we evaluate the effectiveness, efficiency and imperceptibility of TSFool. To illustrate its advantages, there are six white-box attacks and three black-box attacks from basic to state-of-the-art that serve as the benchmarks. The detailed experimental setup can be found in Section B.1. Besides, the Python code of TSFool,

Method	Attack Success Rate	Generation Number	Average Time Cost (s)	Perturbation Ratio (ρ^*)	Camouflage Coefficient
FGSM	72.12%	300.75	0.0018	37.13%	1.0804
JSMA	83.53%		1.0287	15.06%	0.9476
DeepFool	81.58%		0.0276	21.45%	1.0107
PGD (BIM)	76.84%		0.1327	22.71%	0.9938
C&W	69.90%		3.2016	5.16%	0.9372
Auto-Attack	80.11%		0.1824	22.55%	0.9745
Boundary Attack	79.01%		9.0399	3.04%	0.8788
HopSkipJump	83.17%		12.3068	3.86%	0.8872
Transfer Attack	19.54%	250	-	7.68%	1.2010
TSFool	**87.76%**	305	**0.0230**	**4.63%**	**0.8147**

Table 1: The average performance of experimental methods on the 10 UCR univariate time series datasets, including the ASR, the number of samples generated, the time cost and the two measures ρ^* and $\mathcal{C}$ for imperceptibility. TSFool realizes state-of-the-art performance in ASR and the proposed CC. For local perturbation, TSFool is slightly behind Boundary Attack and HopSkipJump, while they are two to three orders slower than TSFool. TSFool also outperforms all the benchmarks in efficiency except FGSM, which is not surprising because FGSM is a simple single-step method without satisfying results under other measures. The standard deviations of five runs are omitted as they are typically small (*i.e.* < 0.5% compared with the mean values). Table 2 in our Appendix[1] further illustrates the details for every single dataset, respectively.

the pre-trained RNN classifiers, and the raw data of our experiments including the crafted adversarial sets are publicly available in our GitHub repository.[2][3]

5.1 Adopted Measures

There are mainly four measures adopted for the evaluation. Firstly, we report the original accuracy of the target classifiers and the Attack Success Rate (ASR) to evaluate the effectiveness of attacks. Secondly, we record the average time for crafting a single adversarial sample as the measure of efficiency. Finally, the imperceptibility is considered from two perspectives namely the global Camouflage Coefficient Equation (5) and the local perturbation. The commonly used measure for the latter is the perturbation ratio:

$$\rho = \frac{\|\delta_{\vec{x}}\|}{\|\vec{x}\|}. \tag{9}$$

We propose a variant ρ^* named **Domain Perturbation Ratio** replacing the original denominator by the specific input domain. This is because typical time series usually have features with large values but narrow distribution ranges, so the input domain can be a better denominator than the absolute value. Given that $\mathcal{X}^{(i)}$ denotes the set of feature values at time step i of all the samples in $\mathcal{X}$, the ρ^* is defined as:

$$\rho^* = \frac{\|\delta_{\vec{x}}\|}{\sum_{i=1}^{|\vec{x}|} \|max(\mathcal{X}^{(i)}) - min(\mathcal{X}^{(i)})\|}, \tag{10}$$

which always provides similar information as ρ but reflects the relative size of perturbation better for time series data.

5.2 Overview of Experimental Results

The average performances (*i.e. modified mean* [2]) of TSFool and the benchmark methods on 10 UCR univariate time series datasets are shown in Table 1. For ***effectiveness***, the average ASR of TSFool is significantly higher than the benchmarks, with their gaps from 4.23% to 68.22%. The ***efficiency*** of TSFool and DeepFool are at the same level, behind FGSM but better than the other seven benchmarks by one to three orders of magnitude. As FGSM is a basic method not outstanding in other measures, we can state that TSFool is efficient enough. For ***imperceptibility***, TSFool not only performs the best under the proposed CC, but also beats seven in nine benchmarks under the conventional local perturbation, with the rest two to three orders slower. This confirms the imperceptibility of TSFool is impressive from the global perspective, and also considerable from local. An intuitive example of attack results by TSFool and the benchmarks is given in Figure 5.

Due to the limited space, this section is just a brief overview of our experimental results, and we leave the detailed analysis in Section B.2 of our Appendix.[1] There are also a number of additional experiments that support our points and strengthen the confidence of the above results, including I) implementing TSFool on a multivariate time series dataset from the UEA archive in Section B.3, where it achieves similar performance to the univariate cases; II) exploring the impact of hyper-parameters for all the experimental methods in Section B.4, to confirm the above results are of general significance; III) additionally evaluating the benchmarks on TPSs in Section B.5, to dispel a possible concern about the consistency of the final compared data and further confirm the fairness of our experiments; IV) conducting subjective human studies with 65 volunteers in Section B.6, to illustrate the benefit of using Camouflage Coefficient in representing real-world imperceptibility of adversarial samples; and V) evaluating TSFool by four common anomaly detection methods in Section B.7, to show its advantages beyond existing attacks in imperceptibility under the challenge of real-world defense.

6 Discussion

While the surprising results in Section 5 are revealed by RNN-based TSC specifically, they are enough to reflect the fact that general consideration beyond image data and feed-forward models is still lacking in the existing knowledge. As a consequence, we believe the current theory of adversarial attack is incomplete and needs to be further refined. Another problem to be noticed is that imperceptibility should have been one of the most important measures of an adversarial sample as this is part of its definition [38], without which it should not be believed really "adversarial". However, imperceptibility has not received sufficient attention to date, at least not at the same level as ASR. So constructing widely recognized measures for the imper-

[2] For univariate time series: https://github.com/FlaAI/TSFool.
[3] For multivariate time series: https://github.com/FlaAI/Multi-TSFool.

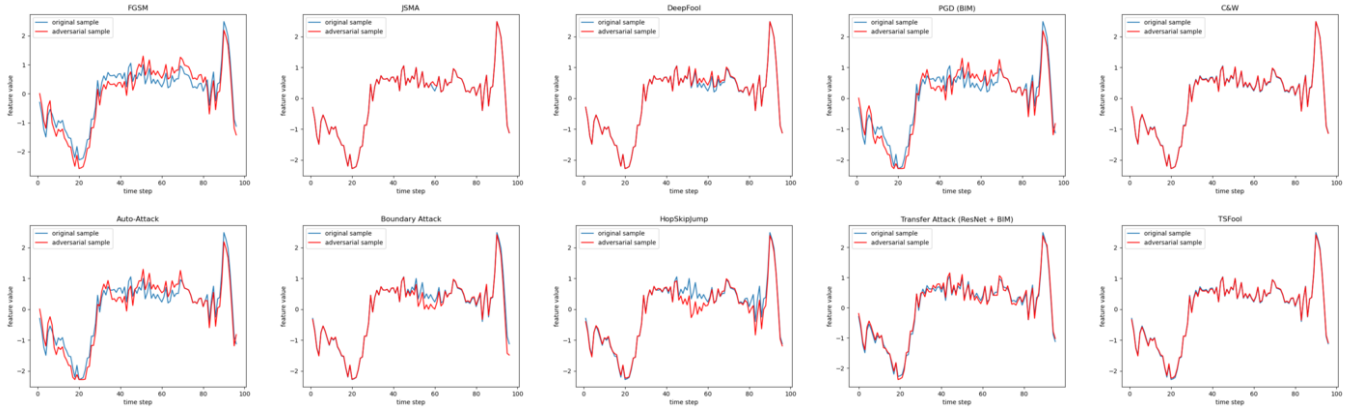

Figure 5: The figures show an instance of adversarial attacks through the nine benchmark methods and TSFool on the UCR-ECG200 dataset. The results of FGSM, PGD and Auto-Attack are unsatisfying with obvious distortion. While JSMA and DeepFool perform better in this case, their average performances in Table 1 are just mediocre. On the contrary, Boundary Attack and HopSkipJump are shown more successful in local perturbation control, but this instance reveals that sometimes they can be unstable, not to mention their worst efficiency. Only C&W, Transfer Attack and TSFool basically realize imperceptible perturbation here, while it should be remembered that on average, C&W is around two orders slower than TSFool and Transfer Attack achieves the worst ASR among all the experimental methods.

ceptibility of adversarial samples is still an important and promising direction. We hope some preliminary ideas regarding these two problems in this paper could raise the community's attention and be instructive for future research.

There are a few related works making preliminary explorations in these directions. For instance, Belkhouja et al. [5] provide theoretical and empirical evidence to demonstrate the effectiveness of *dynamic time warping* (DTW) over the standard *Euclidean* distance metric regarding the robustness of NNs for time series domain, which also emphasizes the significance of new measures in this field. We provide further discussion in Section C.1 and showcase that, by using DTW as a measure of distortion, TSFool still outperforms PGD. Zhang et al. [41] argue that considering the distance to class boundary, adversarial samples should have unequal importance and should be assigned with different weights, which also lays a solid foundation for our idea to pick TPS in Section 3.1. But still, we understand someone may view this as a weakness of TSFool as this violates the conventional setting in the evaluation of attack methods to exactly generate an adversarial sample for every single benign one. Fortunately, picking TPS is vital but not indispensable for TSFool. We further discuss the details in Section C.3, and provide an extended version of TSFool without the above weakness. Additional experiments there confirm that the extended TSFool is still the most competitive choice compared with the benchmarks.

Although as aforementioned, our contribution is not limited to proposing the TSFool approach, someone may still doubt whether it is indeed motivated enough to specifically design for RNN-based TSC, as more state-of-the-art solutions for TSC tasks are based on convolutional NNs or transformers [10]. Nevertheless, the fact is that to date RNN-based TSC applications are still popular in real-world practice [15, 16, 39], without an effective approach to measuring their robustness [16, 19], which leaves potential threats to the public. TSFool can be viewed as a "gray-box" method. In short, considering the impact of specific modes of RNN running to i-WFA extraction in different applications, TSFool may either be implemented in a black-box way, or rely on a part of white-box information. We leave a more detailed explanation in Section A.3. Please notice that this is just about the property of TSFool, instead of the background setting of this paper. Another point to be noticed is that as TSFool relies on existing vulnerable samples wrongly predicted by the target RNN classifier, a natural requirement is that such samples must exist. Generally, this should not be a serious concern, as they can be any real-world sample at inference time instead of just from the supervised dataset. In practice, TSFool provides several hyper-parameters for fine-tuning as shown in Section B.4 and supports both targeted and untargeted attacks, which makes it widely applicable. We also showcase its potential for adversarial training in Section C.2.

7 Conclusion

In this paper, given the lack of research and applicable approach in the field of adversarial samples for RNN-based TSC, we propose the TSFool attack, significantly outperforming existing methods in effectiveness, efficiency and imperceptibility. What's important, the novel global optimization objective "Camouflage Coefficient" proposed to refine the adversarial attack as a multi-objective optimization problem may be instructive for the improvement of the current theory, and the methodology proposed based on latent manifold to heuristically approximate the solution of such a new optimization problem can also be easily transferred to other types of models and data, providing a new feasible way to craft imperceptible adversarial samples. For future works, further exploring the newly defined multi-objective optimization problem to find better approximation solutions is an interesting topic, and the attempt to realize our methodology in other kinds of real-world tasks is in progress at present.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No: 92270123, 62072390), and the Research Grants Council, Hong Kong SAR, China (Grant No: PolyU 15203120, 15226221, 15209922, and 15210023).

References

[1] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 3

[2] G. Arulmozhi. *Statistics For Management, 2nd Edition*. Tata McGraw-Hill Education, 2009. ISBN 9780070153684. 6

[3] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018. 2

[4] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. The uea multivariate time series classification archive. *arXiv preprint arXiv:1811.00075*, 2018. 2, 5

[5] T. Belkhouja, Y. Yan, and J. R. Doppa. Dynamic time warping based adversarial framework for time-series domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022. 7

[6] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 3

[7] V. Campos, B. Jou, X. Giró-i Nieto, J. Torres, and S.-F. Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 3

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2017. 1, 2

[9] J. Chen, M. I. Jordan, and M. J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *IEEE Symposium on Security and Privacy (S&P)*, 2020. 1, 3

[10] L. Cheng, R. Khalitov, T. Yu, J. Zhang, and Z. Yang. Classification of long sequential data using circular dilated convolutional neural networks. *Neurocomputing*, 2023. 7

[11] F. Chollet. *Deep learning with Python*. Simon and Schuster, 2021. 4

[12] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, 2020. 2

[13] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 2019. 1, 2, 5

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 1

[15] 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 *International Conference on Data Engineering (ICDE)*. IEEE, 2022. 1, 2, 7

[16] D. Ding, M. Zhang, F. Feng, Y. Huang, E. Jiang, and M. Yang. Black-box adversarial attack on time series classification. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023. 1, 3, 7

[17] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Adversarial attacks on deep neural networks for time series classification. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019. 2, 3

[18] C. Fefferman, S. Mitter, and H. Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 2016. 4

[19] A. H. Galib and B. Bashyal. On the susceptibility and robustness of time series models through adversarial attack and defense. *arXiv preprint arXiv:2301.03703*, 2023. 1, 7

[20] P. Gao, L. Yu, Y. Wu, and J. Li. Low latency rnn inference with cellular batching. In *European Conference on Computer Systems (EuroSys)*, 2018. 1, 3

[21] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015. 1, 2

[22] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 2020. 1

[23] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery (DMKD)*, 2019. 2

[24] F. Karim, S. Majumdar, and H. Darabi. Adversarial attacks on time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 1

[25] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *Workshop Track @ International Conference on Learning Representations (ICLR)*, 2017. 1, 2

[26] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters (PRL)*, 2014. 2

[27] Y. Li, M. Cheng, C.-J. Hsieh, and T. C. Lee. A review of adversarial attack and defense for classification methods. *The American Statistician*, 2022. 2, 3

[28] L. Lin, B. Xu, W. Wu, T. W. Richardson, and E. A. Bernal. Medical time series classification with hierarchical attention-based temporal convolutional networks: A case study of myotonic dystrophy diagnosis. In *CVPR Workshops*, 2019. 2

[29] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2

[30] S. Marcel and Y. Rodriguez. Torchvision the machine-vision package of torch. In *ACM International Conference on Multimedia (ACM MM)*, 2010. 1

[31] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[32] M. C. Mozer. A focused backpropagation algorithm for temporal pattern recognition. In *Backpropagation*. Psychology Press, 2013. 1, 3

[33] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. 1, 3

[34] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016. 1, 2

[35] N. Papernot, P. McDaniel, A. Swami, and R. Harang. Crafting adversarial input sequences for recurrent neural networks. In *IEEE Military Communications Conference (MILCOM)*, 2016. 1, 2, 3

[36] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2017. 1, 3

[37] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation (TEC)*, 2019. 1, 3

[38] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. 1, 2, 6

[39] T. Wu, X. Wang, S. Qiao, X. Xian, Y. Liu, and L. Zhang. Small perturbations are enough: Adversarial attacks on time series prediction. *Information Sciences*, 2022. 1, 2, 3, 7

[40] X. Zhan, Y. Li, R. Li, X. Gu, O. Habimana, and H. Wang. Stock price prediction using time convolution long short-term memory network. In *International Conference on Knowledge Science, Engineering and Management (KSEM)*. Springer, 2018. 2

[41] J. Zhang, J. Zhu, G. Niu, B. Han, M. Sugiyama, and M. Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations (ICLR)*, 2020. 3, 7

[42] S. Zhang, Y. Wu, T. Che, Z. Lin, R. Memisevic, R. R. Salakhutdinov, and Y. Bengio. Architectural complexity measures of recurrent neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 1, 3

[43] X. Zhang, X. Du, X. Xie, L. Ma, Y. Liu, and M. Sun. Decision-guided weighted automata extraction from recurrent neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. 4