International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-023-00438-0

REGULAR PAPER

Time series adversarial attacks: an investigation of smooth perturbations and defense approaches

Gautier Pialla[1] · Hassan Ismail Fawaz[3] · Maxime Devanne[1] · Jonathan Weber[1] · Lhassane Idoumghar[1] · Pierre-Alain Muller[1] · Christoph Bergmeir[2,4] · Daniel F. Schmidt[2] · Geoffrey I. Webb[2] · Germain Forestier[1,2]

Received: 27 December 2022 / Accepted: 27 July 2023

Abstract
Adversarial attacks represent a threat to every deep neural network. They are particularly effective if they can perturb a given model while remaining undetectable. They have been initially introduced for image classifiers, and are well studied for this task. For time series, few attacks have yet been proposed. Most that have are adaptations of attacks previously proposed for image classifiers. Although these attacks are effective, they generate perturbations containing clearly discernible patterns such as sawtooth and spikes. Adversarial patterns are not perceptible on images, but the attacks proposed to date are readily perceptible in the case of time series. In order to generate stealthier adversarial attacks for time series, we propose a new attack that produces smoother perturbations. We introduced a function to measure the smoothness for time series. Using it, we find that smooth perturbations are harder to detect both visually, by the naked eye and by deep learning models. We also show two ways of protection against adversarial attacks: the first one by detecting the attacks using a deep model; the second one by using adversarial training to improve the robustness of a model against a specific attack, thus making it less vulnerable.

Keywords Time series · Adversarial attack · Smooth perturbations · InceptionTime · BIM

1 Introduction

A time series is a set of data points ordered in time. Time series have become a growing field of research in deep learning and more globally in artificial intelligence. Nowadays, thanks to the presence of sensors, they have become abundant and we can find use cases in almost all sectors of industry. For example, time series are used in healthcare [13], for weather forecasting [14] and for predictive maintenance [8].

Time series classification refers to the task of classifying time series according to the presence or not of phenomena. Szegedy et al. [7] have found that adding a small perturbation to an input sample can change a classifier's output. This is known as an *adversarial attack*. It is illustrated in Fig. 1.

As all neural networks are vulnerable to adversarial attacks, many attacks have been proposed. However, to date the majority of these have been developed for image classification tasks. It is necessary to study adversarial attacks in order to assess the robustness of the models, and to prevent them on critical systems. For example, Eykolt et al. [6]

✉ Germain Forestier
germain.forestier@uha.fr

Gautier Pialla
gautier.pialla@uha.fr

Maxime Devanne
Maxime.Devanne@uha.fr

Jonathan Weber
Jonathan.Weber@uha.fr

Lhassane Idoumghar
Lhassane.Idoumghar@uha.fr

Pierre-Alain Muller
Pierre-Alain.Muller@uha.fr

Christoph Bergmeir
Christoph.Bergmeir@monash.edu

Daniel F. Schmidt
Daniel.Schmidt@monash.edu

Geoffrey I. Webb
Geoffrey.I.Webb@monash.edu

1 Université de Haute-Alsace, Mulhouse, France

2 Monash University, Melbourne, Australia

3 Ericsson Research, Massy, France

4 University of Granada, Granada, Spain

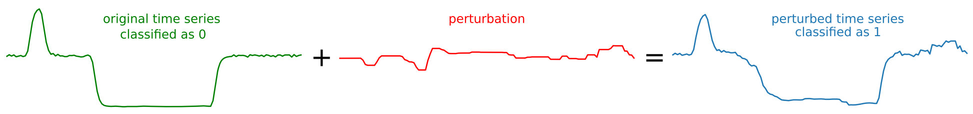


Fig. 1 Scheme of adversarial attack. Time series from the BME dataset, perturbation generated with SGM, not represented at scale

showed an application on real-world road sign classification, which is an obvious threat for autonomous vehicles.

Fawaz et al. [10] introduced and adapted some of them for time series classification. The main difference between adversarial attacks on images and time series lies in the visualization and the interpretation of the data. When slightly changing the value of one or few pixels, an image will always look the same and have the same appearance to the human visual system. These changes only affect how the neural network will process the data, but not how we, humans, perceive the image. For images, the human classifier is a competitive benchmark, often used as gold standard. For time series classification, it is not, because time series data are more complex to analyze.

The attacks introduced by Fawaz et al. [10] are effective to perturb time series of the UCR Archive in order to mislead a deep learning system [4]. But when we look at their visual appearance, it is sometimes easy to distinguish the disturbed series from the original ones. Indeed, these perturbed samples often contain patterns like spikes of a sawtooth. Because the presence of such elements can easily be spotted, they can warn about the presence of an attack.

In this paper, we will introduce a novel adversarial attack based on a gradient method. We will show that it outperforms BIM's performance over most of the UCR archive datasets in terms of misleading a deep learning classifier. But unfortunately this method generates perturbations that also contain spike and sawtooth patterns. We will then explain how we reduced these patterns, by enforcing a smoothness condition. First, we will show its impact visually, using a set of examples. Secondly, we introduced a function used to measure and compare the smoothness of time series. Finally, we will show how adversarial training is a good way to improve a time series classifier's robustness against smoothed perturbations.

Our main contributions are:

- A novel adversarial attack for time series classifiers that outperforms BIM
- An altered version of the first attack that produces smooth perturbations
- We introduced a function in order to measure and compare the smoothness of time series
- A quantitative benchmark of our two methods along with BIM over the UCR archive
- Qualitative evidence that smoothed perturbations are harder to detect by visual inspection
- Quantitative evidence that adversarial training is a good counter measure against smooth attacks

2 Related work

Given a neural network trained on an image classification task, such as ImageNet, Szegedy et al. [21] showed that it is possible to change the model output by adding low magnitude noise, small enough to be imperceptible to the human eye. It was also shown that this vulnerability is present regardless of the number of layers, activation functions or training data and thus affects all deep neural networks.

Goodfellow et al. [7] proposed a single step attack called Fast Gradient Sign Method (FGSM). Then, Kurakrin et al. [15] presented the Basic Iterative Method (BIM), an iterative version of FGSM. Inspired by them, many similar attacks were proposed, like M-FGSM [5] or vr-IGSM [22].

Other approaches where studied, like adding black and white strips on stop signs [6] or stickers on objects [16]. These real life attacks raised the issue of security threat for sensitive applications like autonomous vehicles. Along with new attacks, multiple defensive strategies have also emerged, including leveraging denoisers [17], randomization [24] and adversarial training [12, 23].

Adversarial training trains a model using both normal and perturbed samples. Rathore et al. [19] show how adversarial training can help a model to become more robust.

Most of the work on adversarial attacks was first done on image classification, as it is a trending topic in deep learning. It is only later that Fawaz et al. [10] introduced adversarial attacks for time series classification.

It is sometimes quite straightforward to adapt adversarial attacks from images to times series. However, some attacks that work well on images can't be used, or are ineffective on time series. For example Su et al. [20] describes attacks where only one pixel of an image is affected. An equivalent perturbation for time series would modify the value of only a single data point. But such modifications would be very noticeable as it takes extreme values to sufficiently perturb a sample based solely on a single data point.

Adversarial attacks can be categorized into black and white-box strategies. Black-box attacks, like those presented in [2, 18], don't use any knowledge of the architecture, the parameters, or the weights of the model. They also do not have access to the training data. Huan et al. [9] showed that even in these conditions, many current models are still at risk. In contrast, white-box attacks may use any of those elements to perform the attack. Some attacks have both black-box and white-box variants like the Carlini and Wagner method [3]. In this paper, we will focus exclusively on white-box attacks.

3 Background material

3.1 Mathematical description

In this paper, we only use univariate time series. We can describe each time series as a vector $\mathbf{x}$ such as $\mathbf{x} \in \mathbb{R}^T$, $\mathbf{x} = [x_1, ..., x_T]$ with T denoting its length.

Given a time series classifier f and a time series $\mathbf{x}$, the aim of an adversarial attack is to perturb the classifier by adding a small variation $\mathbf{r}$ to a time series $\mathbf{x}$. $\mathbf{r}$ will be referred as noise or perturbation. We call the perturbed time series $\mathbf{x}^{adv} = \mathbf{x}+\mathbf{r}$ an *adversarial sample*. The attack is successful if the class predicted for the original time series is different from the class predicted for the adversarial sample, $\arg\max f(\mathbf{x}) \neq \arg\max f(\mathbf{x}^{adv})$. The added noise $\mathbf{r}$ must be imperceptible by design, and thus we need that $\mathbf{x}$ and $\mathbf{x}^{adv}$ remain close to each other.

3.2 Basic iterative method

In order to improve the success rate of FGSM, Kurakin et al. [15] developed BIM. At each iteration N, the gradient is computed and then added to the input, in the same way as for FGSM. Instead of minimizing the loss function, the aim is to maximize it by taking a step in the direction of the gradient. At each iteration, the values are clipped using an ϵ parameter. This ensures that each value of $\mathbf{x}^{adv}$ will stay close to $\mathbf{x}$ within a ϵ-neighborhood.

$$\begin{aligned} \mathbf{x}_0^{adv} &= \mathbf{x} \\ \mathbf{x}_{N+1}^{adv} &= \text{Clip}_{\mathbf{x},\epsilon}\left\{\mathbf{x}^{adv}{}_N + \alpha\, \text{sign}(\Delta_{\mathbf{x}} J(\Theta, \mathbf{x}^{adv}{}_N, y_{true}))\right\} \end{aligned} \tag{1}$$

y_{true} denotes the label of the time series $\mathbf{x}$. If we don't know y_{true}, as in a real attack scenario, we replace it with $f(\mathbf{x})$. The noise clipping is done for $\mathbf{r} = \mathbf{x}^{adv} - \mathbf{x}$ as follows:

$$\forall r_i \in \mathbf{r}, r_i = \begin{cases} \epsilon, & \text{if } r_i > \epsilon \\ -\epsilon, & \text{if } r_i < -\epsilon \end{cases}$$

By adding iterations, BIM becomes more effective than FGSM to perturb time series. But BIM requires clipping in order to control the amount of noise. This method had two main disadvantages. First, clipping the noise in such a way often produce sawtooth shapes between $(-\epsilon)$ and $+\epsilon$ as we can see in Fig. 7. This particular pattern can easily be detected when added to a smooth time series and is, therefore, to be avoided.

With BIM, in order to obtain a stealthier noise, we need to reduce the value of ϵ. By doing this, the sawtooth shapes will be harder to be noticeable, but this will result in a lower attack success rate. This trade-off prevents the perturbations that are both hard to detect and have a high attack success rate.

In this paper, we decided to use BIM as our main competitor. FGSM and BIM are the most well-known attacks. Moreover, to our knowledge, they are the only adversarial attacks adapted for time series, so it was convenient to work with them. Other methods that we cited previously, M-FGSM and vr-IGSM, are, respectively, variations of FGSM and BIM. They do not tackle the issue of the occurrence of perceptible patterns. Thus, they should behave in the same way as FGSM and BIM, and we have not added them into our benchmark.

3.3 Carlini and Wagner

Carlini and Wagner's adversarial attack (C&W) [3] is a state-of-the-art adversarial attack for image classifiers. C&W is an optimization-based method that aims to minimize:

$$\min ||\mathbf{r}||_2 + c \times \phi(\mathbf{x}+\mathbf{r}) \text{ with } \phi(\mathbf{x}+\mathbf{r}) \leq 0 \tag{2}$$

where c is a hyper-parameter balancing the trade-off between the L_2 regularization and ϕ, a function that enforces the misclassification. Several functions ϕ are proposed in the original paper. We chose to use the recommended one:

$$\phi(\mathbf{x}) = \left(\left[\max_{i \neq t} Z(\mathbf{x})_i\right] - Z(\mathbf{x})_t\right)^+. \tag{3}$$

Here $Z(\mathbf{x})$ denotes the logits of the model before the softmax function. As the C&W method is a targeted attack, t depicts the class to be obtained after perturbation.

In this work, we adapted this attack to time series classifiers. The original work applied the constraint $\mathbf{x}^{adv} \in [0, 1]^n$. For time series, we removed this constraint as we are not working with pixels and time series are not bounded by specific values. When benchmarking C&W with other non-targeted attacks, we will select the targeted class as the class with the second biggest softmax score.

4 Proposed methods

4.1 Gradient method (GM)

In order to correct the flaws of BIM, we need to design a method that, given a model, can perturb a time series while optimizing the quantity of noise according to the L2 norm.

Ensuring $f(\mathbf{x}) \neq f(\mathbf{x}')$ can be written as a maximization problem of the KL-divergence between the two probability distributions, as follows:

$$\max D_{KL}(f(\mathbf{x}), f(\mathbf{x}')) \equiv \sum^{c} f(\mathbf{x}) \log \frac{f(\mathbf{x})}{f(\mathbf{x}')}, \tag{4}$$

with c denoting the classes in the dataset.

Generating an adversarial example can then be written as follows where the primary addition is the term $(-\|\mathbf{x}-\mathbf{x}'\|_2)$ to be maximized:

$$\max \left\{\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x}')) - \|\mathbf{x}-\mathbf{x}'\|_2\right\}, \tag{5}$$

with μ denoting a hyper-parameter to control the penalty of miss-classification.

Let us consider the generated time series $\mathbf{x}' = \mathbf{x} + \mathbf{r}$. Then the maximization problem is equivalent to the following minimization problem:

$$\min \{-\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x}+\mathbf{r})) + \|\mathbf{r}\|_2\} \tag{6}$$

We can add a hyper-parameter α in order to control the regularization of $\|\mathbf{r}\|$. Finally, we have:

$$\mathbf{x}^{adv} = \min \{-\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x}+\mathbf{r})) + \alpha\|\mathbf{r}\|_2\} \tag{7}$$

4.2 Smooth gradient method (SGM)

The previous method manages to generate adversarial samples while optimizing the L2 norm of $\mathbf{r}$. But it does not prevent the appearance of sawtooth. In order to obtain smoother perturbations, we need to ensure a smoothness condition on $\mathbf{r}$. This can be done by adding a fused lasso term to the minimization. The equation can now be written as:

$$\begin{aligned}\min\{&-\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x}+\mathbf{r})) + \alpha\|\mathbf{r}\|_2 \\ &+\lambda \sum_{i=1}^{T-1} \|r_i - r_{i+1}\|_1\}\end{aligned} \tag{8}$$

In equation 8, $\|.\|_1$ denotes the L1 norm. λ is a hyper-parameter that controls the penalty for the smoothness condition. To minimize the latter equation, we will use gradient descent by computing the gradient with respect to $\mathbf{r}$ (which will be initialized randomly).

4.3 Measuring the smoothness of a time series

In order to compare the smoothness between the original and perturbed time series, we need a suitable metric. As we did not find such a metric in the literature, we used penalties used for spline smoothing. Splines are functions defined by several pieces of polynomials. They are often used by computer graphics drawing software or computer-aided design software. Some of those software use as a smoothing penalty:

$$\int^{\prime\prime} f(x)^2 dx \tag{9}$$

In order to use this penalty for time series, we need to replace the integral by a sum and the second derivative by the finite difference, as time series are not continuous in time.

$$g(\mathbf{x}) = \sum_{t=1}^{T-1} (\mathbf{x}_{t-1} - 2\mathbf{x}_t + \mathbf{x}_{t+1})^2 \tag{10}$$

To avoid having the result impacted by the length of the series, we can divide it by T. Finally, we have:

$$s(\mathbf{x}) = \frac{1}{T-1} \sum_{t=1}^{T-1} (\mathbf{x}_{t-1} - 2\mathbf{x}_t + \mathbf{x}_{t+1})^2 \tag{11}$$

Theorem 1 *We can use the functions g and s to compare the smoothness of two time series. For two time series* $\boldsymbol{x}$ *and* $\boldsymbol{y}$, *if* $s(\boldsymbol{x}) < s(\boldsymbol{y})$, *we can say that* $\boldsymbol{x}$ *is smoother than* $\boldsymbol{y}$.

Proof A differentiable function f is said to have an L-Lipschitz continuous gradient if for some $L > 0$:

$$||\nabla f(a) - \nabla f(b)|| \leq L||a-b|| \tag{12}$$

Here L enforces a regularity on the function f. In the context of time series, this regularity can be translated by smoothness.

Previously, in order to compare the smoothness between time series, we proposed to use the function g:

$$\begin{aligned} g(\mathbf{x}) &= \sum_{t=1}^{T-1} (\mathbf{x}_{t-1} - 2\mathbf{x}_t + \mathbf{x}_{t+1})^2 \\ &= \sum_{t=1}^{T-1} ((\mathbf{x}_{t+1} - \mathbf{x}_t) - (\mathbf{x}_t - \mathbf{x}_{t-1}))^2 \\ &= \sum_{t=1}^{T-1} \left(d_{\mathbf{x}_{t+1}} - d_{\mathbf{x}_t}\right) \\ &= \sum_{t=1}^{T-1} ||d_{\mathbf{x}_{t+1}} - d_{\mathbf{x}_t}||^2 \end{aligned}$$

$$= ||d_{\mathbf{x}_1} - d_{\mathbf{x}_0}||^2 + ||d_{\mathbf{x}_2} - d_{\mathbf{x}_1}||^2 + \dots \\ + ||d_{\mathbf{x}_{T-1}} - d_{\mathbf{x}_{T-2}}||^2 \qquad (13)$$

By introducing Eq. 12, we get:

$$\begin{aligned} g(\mathbf{x}) &\le L^2||1-0||^2 + L^2||2-1||^2 \\ &\quad + \dots + L^2||(N-1)-(N-2)||^2 \\ &\le L * (T-1) \\ &\le L \end{aligned} \qquad (14)$$

We can therefore compare the smoothness of two time series, **x** and **y**, by using the function g. If $g(\mathbf{x}) < g(\mathbf{y})$, we have $L_{\mathbf{x}} < L_{\mathbf{y}}$ and we can conclude that the time series **x** is smoother than **y**.

In our GitHub repository, we provide some examples using several hand crafted time series. We shows, when the time series is a straight line, that the function returns zero as expected. For sinusoids, we showed that increasing either the amplitude or the number of periods also increases the value.

5 Experimental setup

In this section, we present the data, models, and the parameters we used during our experiments.

5.1 Classifier and datasets

As our work consists of fooling a deep learning model, we decided to use the InceptionTime [11] classifier in all our experiments. InceptionTime is a TS classifier, which was the state-of-the-art model on the UCR Archive [4], when published in 2019. During our preliminary work, we tested other deep models like ResNet or FCN. We decided not to add them to the paper because we did not notice any visual or quantitative difference from InceptionTime. As InceptionTime is still the state-of-the-art model for deep learning TSC, we decided to only focus on this model. All the weights used are the InceptionTime defaults, as used and presented in its paper.

In order to demonstrate our results over several datasets, we used the well know TSC benchmark UCR Archive. The 2018 version of this archive comprises 128 univariate time series datasets.

Each dataset of the UCR Archive is split between the training and the test sets. When generating adversarial samples, we used the samples of the test set, as the model has only been trained on the training set.

A single experiment is realized on the RTD dataset [1]. It was build upon a technique called air writing. It consists of writting characters in the air, here digits, which are then recognized by a computer or a mobile phone. This dataset is a collection of 20k trajectories made by various people. A class is assigned for each one of the ten digits.

5.2 Reproductibility

The code used and all our results are publicly available in our companion repository.[1]

All experiments were done by leveraging the computation power of a remote GPU cluster containing Nvidia GTX 1080 Ti graphic cards. Reproducing the results on a single graphic card takes roughly 7 days of computing time.

5.3 Hyper-parameters

For BIM we set the number of iterations at 1000. For the noise clipping, we use the value $\epsilon = 0.1$. We use the same value of ϵ, when applying the noise clipping to the Gradient Method.

In the case of GM and SGM, both μ and α parameters are always set to 1. In the case of SGM, when nothing is specified, λ is also equal to 1.

Regarding the C&W method, we chose $c = 1$. c controls the the L2 regularization, like α for SGM. Thus, we selected the same value.

5.4 Comparison metrics

5.4.1 Average success rate

For evaluating the relative success of adversarial attacks, we used the Average Success Rate (ASR). The ASR, corresponds to the rate of reclassified samples. In other words, it is equal to the percentage of cases where the attack was able to alter the output of the network ($f(\mathbf{x}) \neq f(\mathbf{x}^{adv})$).

5.4.2 L2 norm

The L_∞ norm is commonly used to quantify the noise for adversarial attacks. This is especially true in the case of attacks on images. The L_∞ norm of a time series is equal to $\|\mathbf{x}\|_\infty = \max_t |x_t|$. As explained earlier, our aim is to design smooth perturbations that are hard to detect by the naked eye. Moreover attacks designed for images are easily detectable when adapted to time series. Thus, we needed to evaluate the overall quantity of noise, not just its maximum value, and choose to use the L2 norm over the L_∞ norm.

[1] https://github.com/Gpialla/SmoothPerturbationsTSAA.

5.5 Adversarial training

We will present an example of adversarial training using adversarial samples generated by SGM. For each dataset, we doubled the size of the training set, by adding the corresponding adversarial samples of the original training set. The validation is done with the original test set, without additional adversarial samples. Finally, we will show how adversarial training is effective at reducing a classifier's susceptibility to adversarial attacks.

6 Results

In this section, we will first compare SGM with the other methods according to the two metrics we selected: the ASR and the L2 norm. The benchmark between the others methods is available in our companion repository. In a second study, we will vary the SGM's λ parameter and see its influence on the ASR. Finally, we will propose two counter measures against SGM. The first one consists of improving the robustness of the classifier by using adversarial training. The second one consists of identifying the adversarial samples using a deep classifier.

6.1 SGM benchmark

Figure 2 represents a Win/Draw/Loss diagram comparing BIM and SGM. Each blue dot represents a single dataset. If a dot lies above the median line in the upper left triangle, it means that this dataset has an average value bigger for SGM than for BIM for the given metric.

As we want to maximize ASR, in the corresponding plot, the most successful method is the one with the most dots on its side of the median line. For the L2 norm, however, the reasoning is reversed as we want to minimize the metric.

Given Fig. 2, as the dots are evenly distributed, we conclude that SGM as an overall ASR as good as BIM on the UCR archive. This also means that SGM manages to perturb

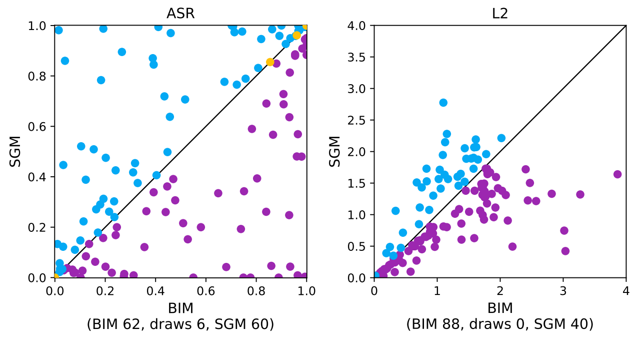


Fig. 2 Win/draw/loss diagram. BIM versus SGM. On the left: average success rate, on the right: L2 norm of the perturbation

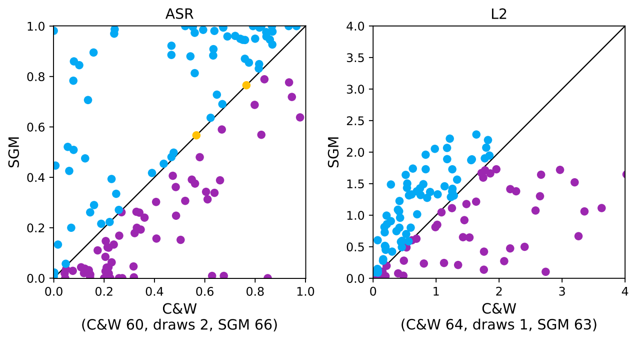


Fig. 3 Win/draw/loss diagram. C&W versus SGM. On the left: the average success rate, on the right: the L2 norm of the perturbation

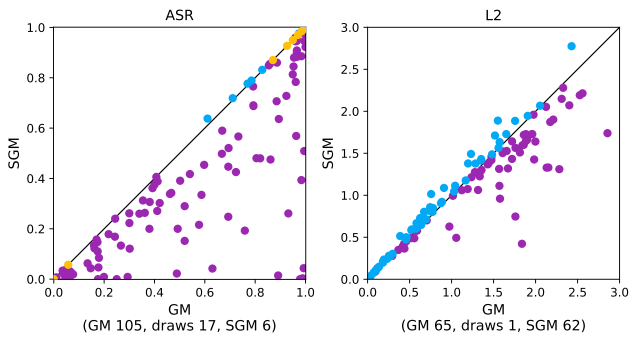


Fig. 4 Win/draw/loss diagram. GM versus SGM. On the left: the average success rate, on the right: the L2 norm of the perturbation

datasets that BIM cannot and vice-versa. But for an equivalent efficiency, BIM introduces an higher quantity of noise than SGM, in a majority of datasets.

Figure 3 compares SGM with C&W. Regarding the ASR we notice that SGM is slightly better than C&W. The same can be concluded for the L2 norm. But when looking at the distribution of the scores, we can all SGM's scores are below 2.5, where many C&W scores are bigger. This suggests that the C&W attack produces more noise than SGM.

Figure 4 compares GM with SGM. We can see that for almost all datasets, GM has a better ASR than SGM. Intuitively, we can assert that sawtooth and spikes are patterns that help to successfully perturb a TSC. SGM, by design, can not produce such patterns. Thus, it is harder for SGM to achieve a high ASR. By enforcing the smoothness condition, we are doomed obtain a decreased ASR. In addition, some datasets, such as those with truly distinct classes or with smooth time series, are particularly difficult to be perturbed by the SGM method. In these specific cases, smooth perturbations are not sufficient to fool the network.

6.2 Varying the λ parameter

According to our previous results, the best-case scenario would be an attack with GM's ASR and SGM's smoothness. As the only difference between the two methods is the

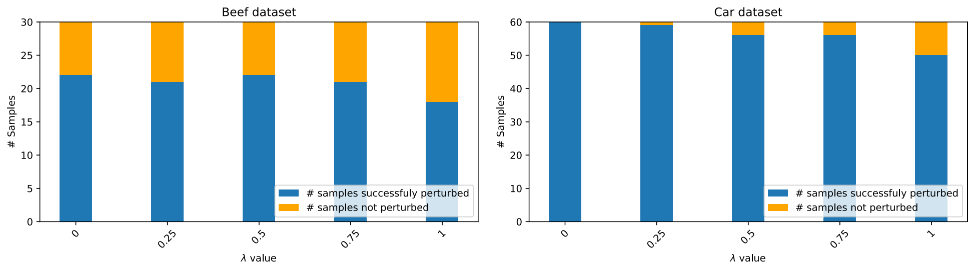


Fig. 5 Varying SGM's λ parameter. For each value of λ is displayed the number of samples successfully perturbed (blue) or not (orange) (color figure online)

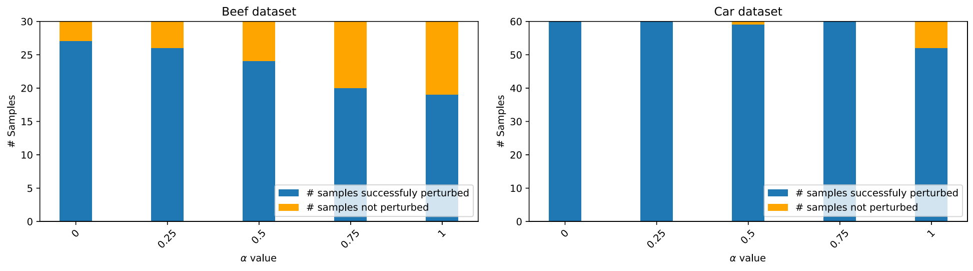


Fig. 6 Varying SGM's α parameter. For each value of α is displayed the number of samples successfully perturbed (blue) or not (orange) (color figure online)

addition of the smoothness condition, it is interesting to vary the λ parameter. If λ is equal to zero, the attack is GM, and if it's equal to 1, we have SGM as we tested it previously.

Figure 5 shows the impact of varying the λ parameter over two datasets, Beef and Car. As we could expect, the more we enforce the smoothness condition, the fewer samples the method manages to perturb successfully.

This parameter should be tuned for each dataset in order to get the optimal trade-off between smoothness and ASR.

6.3 Varying the α parameter

The α parameter controls the penalization of the L2 norm of $\mathbf{r}$. We can see the impact of this parameter in Fig. 6. In both cases, the smaller the α value, the better are the results. Indeed, if SGM is able to introduce more noise, it will perturb better the model. Once again, this parameter can be tuned in order to obtain the optimal trade-off between L2 norm and ASR.

6.4 Visual comparison

In order to remain undetectable by the naked eye, an attack performed on a time series must be as smooth as possible. We propose a visual comparison between the four methods presented, on the same test sample of the Beef dataset. To be fair, we picked a time series that is successfully perturbed by all the attacks.

In this example, shown in Fig. 7, we plotted in green the original time series, and for each method, in blue the perturbed time series and in red the perturbation.

We plotted a second version of GM with a clipped perturbation in the same way as BIM. As expected, for BIM and the GM methods, the perturbations are clearly visible, in particular the parts containing sawtooth patterns that are circled in purple. The example of GM with clipping shows that clipping the noise reduces indeed the amount of noise and the visual impact, but not sufficiently enough. SGM is the only attack that produces an adversarial sample with a perturbation that is not noticeable when judging with the naked eye.

The C&W attack is the one that produces the smallest noise. The magnitude of the noise is significantly smaller than the other methods. Thus, the sawtooth patterns are less noticeable but are still present.

Being closer to the eye doesn't mean being closer when using the L2 metric. Indeed, SGM's perturbation has the biggest L2 norm. This shows that, although a method is better on average for a given dataset, this is not necessarily true when we look at each sample independently.

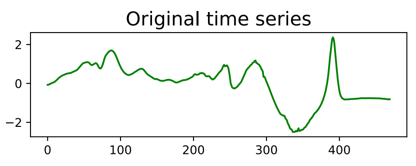
6.5 Comparison of the smoothness

In the previous subsection, we did a visual comparison. Now, using the function 11, we can compute the average for each dataset. We computed the average rank of smoothness for each attack on the UCR archive. These results are represented in Fig. 8.

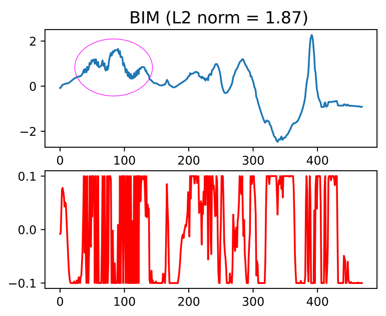
Like the visual comparison, SGM appears as the method that produces the smoother samples and BIM is the opposite. For GM, as expected, clipping the noise reduces its maximum amplitude, thus making it smoother.

The C&W method ranks third below SGM and GM with clipped noise. Nevertheless, C&W ranks first among the methods that do not enforce the smoothness of the perturbations.

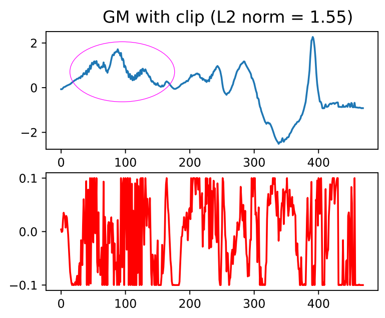
Fig. 7 Time series from the Beef dataset. All methods perturbed time series (blue) and generated noise (red). The purple circles show the presence of sawtooth on the perturbed time series (color figure online)



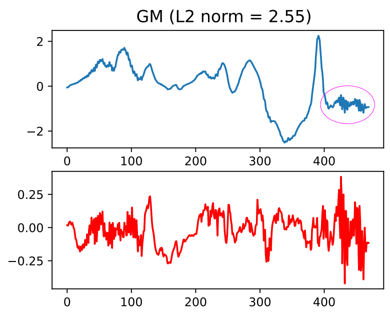
(a) Original time series from the Beef dataset



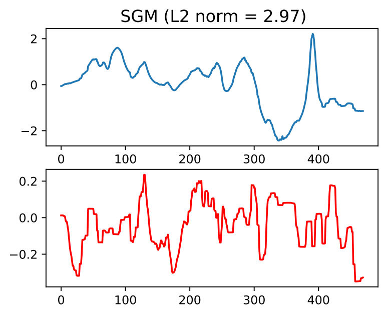
(b) BIM adversarial attack.



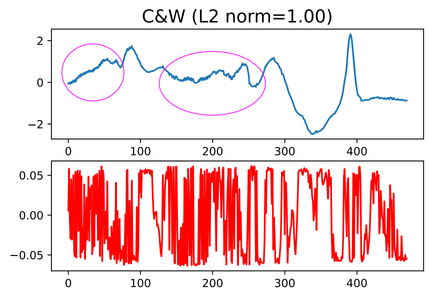
(c) GM adversarial attack.



(d) GM without clip adversarial attack.



(e) SGM adversarial attack.



(f) C&W adversarial attack.

6.6 Do adversarial attacks change the underlying class of the samples?

While in images human perception is a meaningful gold standard, in TSC most of the time it is not. If we classify traffic signs into stop signs and other signs, humans can easily do that. The goal of a machine is then to achieve human performance. The central argument of adversarial attacks is that while to us humans it clearly still is a stop sign, i.e., the underlying class has not changed, the algorithm now suddenly predicts it as something else.

Few time series datasets are easy to visualize and understand at a glance. RTD is one of them. When perturbing this dataset with SGM, we got an ASR equal to 0.829. As most of the samples were successfully perturbed, we can observe in Fig. 9 that the perturbed samples do not differ visually from the originals.

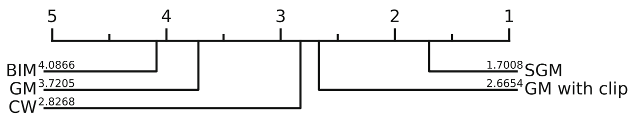


Fig. 8 Ranking of the methods according to the smoothness

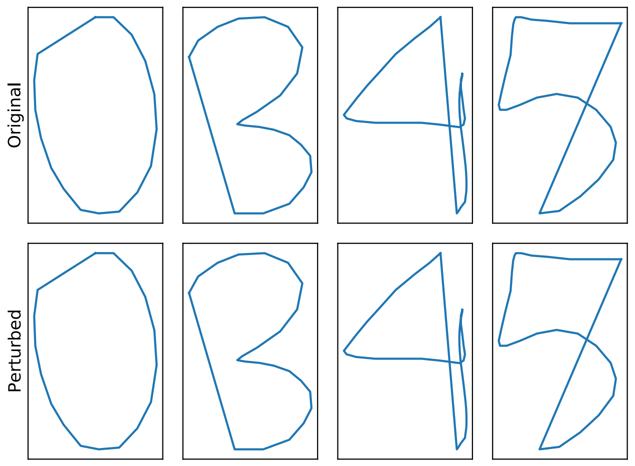


Fig. 9 Original and SGM perturbed samples of the RTD dataset

But most time series datasets are not as easy to visualize as the RTD dataset. For those datasets, we assume that by doing small perturbations the underlying class will not change. But in reality, we cannot be certain that this is the case. So, the question is now what could be a good gold standard to determine a time series' underlying class? In this section, we propose to use 1 Nearest Neighbor (1NN) to determine whether the underlying classes change or not. The 1NN classifier assigns to a sample the class of its closest neighbor. Here we use 1NN combined with the Dynamic Time Warping (DTW) distance.

We experimented on a subset of 22 datasets of the UCR Archive. All the results are available in our GitHub repository. By observing the confusion matrix produced, we can identify 2 groups. The first one is constituted by the datasets ArrowHead, BeetleFly, Car, Earthquake, FaceAll, FordB, FreezerSmall, Ham, InerSkate, InsectWingbeatSound, Lightning7, MoteStrain, OliveOil, ProximalPhalanxTW, TwoPatterns, Wine, WordSynonyms and Yoga. The second one by the datasets EOGVerticalSignal, GunPointOldVersusYoung, and MelbournePedestrian. In the first group, for a majority of samples, the 1NN classifier predicts the same class whether they are perturbed or not. This shows that the underlying class did not change and that the adversarial attack was conducted as expected.

For the second group of datasets, the results are different. For each one of these datasets, the perturbed samples were all classified as belonging to a single class. In those cases, the underlying classes did change. Two examples, one of each group, are shown in Fig. 10.

Through these examples, we have shown that, unlike computer vision, adversarial attacks on time series can change the underlying class. Nevertheless, for 19 out of the 23 datasets we tested, it is not the case. In addition, the RTD dataset is an example of a dataset that can be visualized and successfully perturbed.

6.7 Discriminating adversarial samples

A simple way to protect a model against adversarial attacks is to filter the inputs before feeding them to the network. One can do this by training a small classifier to discriminate perturbed samples. If an example is classified as an attack, then it will not be fed to the second classifier.

In order to know if a classifier is indeed able to do so, we trained a small FCN model. For each dataset, the samples have been split into a training and a test set: 80% of the samples in the training set, and the remaining 20% in the test set. Then, the samples of each set have been perturbed and added back to the set. Thus, we obtained, in the end, datasets balanced between original and perturbed samples.

Table 1 shows the classification results of the five methods presented previously. With almost 62% of accuracy, classifying the samples perturbed by BIM is the easiest task. The hardest one is when using the samples perturbed by SGM. These results confirm that the patterns visible by the naked eye are also easier to be discriminated using a deep model. We can note that the overall accuracies are not very high, proving that classifying perturbed samples is not trivial. In order to achieve better performances, we could have used a deeper model.

6.8 Adversarial training

Adversarial training aims to improve the robustness of a model but without the disadvantages of the previous method. Figure 11 presents the results of adversarial training using SGM adversarial samples. The scatter plot shows that adversarial training highly improves the robustness of the model. For 122 datasets, the ASR is highly lower, and for the remaining 6 datasets, the ASR is the same. In fact, for 72 datasets, after adversarial training, the attack did not manage to perturb any sample, leading to an ASR of zero. This shows the effectiveness of adversarial training against SGM attacks.

We can explain that adversarial training is very effective against SGM attacks because the strength of smoothed attacks is also their weakness. The adversarial samples generated by SGM are close to the original samples. We even showed that a FCN classifier can almost not tell the two apart. Once we improved the robustness of the classifier using adversarial training, it became robust enough not to be affected by these smoothed perturbations.

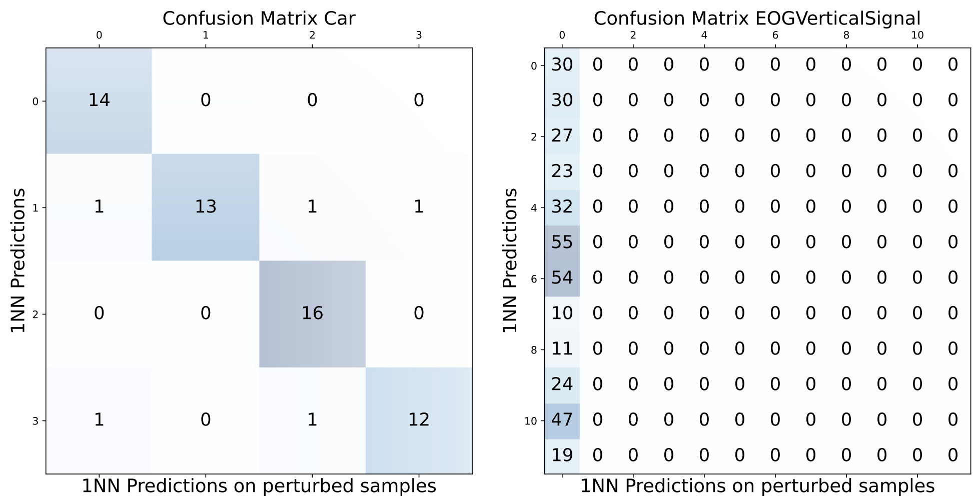


Fig. 10 Example of usage of a 1NN model to determine if the underlying class of a sample changed after being perturbed. Results are presented as confusion matrices. The left example shows that for most samples, the underlying class did not change after the attacks. On the right examples, regarding the EOGVerticalSignal dataset, all the underlying classes have changed to class 0

Table 1 FCN accuracy when classifying original and perturbed samples

Attack name	Accuracy (%)
BIM	61.99
GM with clip	56.97
GM	56.50
CW	55.70
SGM	**54.53**

Bold value indicates the lowest accuracy, highlighting the best performing attack

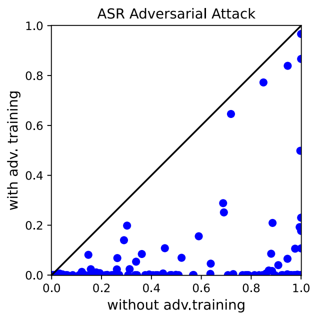


Fig. 11 Adversarial training results

7 Conclusion

In this paper, we explained that adapting adversarial attacks from image classifiers to time series classifiers is not trivial. The attacks are more likely to be detected on time series and thus need smoother perturbations.

We introduced two novel adversarial attacks for time series classification: The Gradient Method (GM) and a smooth version, called Smooth Gradient Method (SGM). We used the Basic Iterative Method (BIM) and the Carlini&Wagner method (C&W), well-known adversarial attacks, as baselines to have a benchmark over the entire UCR archive. We showed that GM has a higher success rate on perturbing an InceptionTime classifier, followed by BIM, SGM and C&W.

Through examples, we illustrated that GM, like BIM and C&W, produces perturbations that have recognizable patterns like spikes and sawtooth. On the one hand, these patterns can help the attack to fool the network, but on the other hand, they can be easily detected, even by the naked eye.

Our second method SGM is based on GM but has an added fuzed lasso regularization. It has the effect of smoothing the generated perturbations.

We introduced a new function that helps measure the regularity, i.e., the smoothness of time series and therefore of adversarial attacks. It is inspired by penalties used for spline smoothing in computer graphics. Using this function, we showed that SGM produces a perturbation that is smoother on average than GM and BIM.

Smoothing the noise makes it harder to differentiate between perturbed and original time series by the naked eye. But smoothed adversarial samples are less effective for attacking the neural network. This highlights the current trade-off between having a stealth attack and an effective one.

Finally, we proposed two methods to counter adversarial attacks. The first one leverages a FCN classifier by training it to discriminate perturbed time series. We showed that the FCN model is better at discriminating perturbed samples from BIM, GM, or C&W attacks than SGM. This proves that generating smooth perturbations is not only better at fooling humans but also deep classifiers.

Then, we showed that adversarial training is an effective way of countering SGM attacks. After adversarial training, for 72 datasets out of the 128 datasets of the UCR archives, it becomes impossible to successfully perturb InceptionTime.

The main limitation encountered while working on adversarial attacks is the possible change of the underlying class. Although we have shown that this happens in a minority of cases, it remains an unsolved issue. We think that tuning the hyper-parameters of the attacks or adding new constraints can be helpful to prevent them. We will explore those options in future works.

Acknowledgements The authors would like to thank the providers of the UCR archive as well as the Mésocentre of Strasbourg for providing access to the GPU cluster.

Author Contributions Conceptualization: GP, HIF, DS; Methodology: GP, MD, JW and GF; Experiments: GP; Validation: MD, JW and GF; Writing—original draft: GP; Writing—review and editing: GP, HIF, MD, JW, LI, P-AM, CB, DS, GW, and GF; Funding acquisition: GF, LI; all authors have read and agreed to the published version of the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. This work was funded by ArtIC project "Artificial Intelligence for Care" (Grant ANR-20-THIA-0006-01) and co-funded by Région Grand Est, Inria Nancy - Grand Est, IHU of Strasbourg, University of Strasbourg and the University of Haute-Alsace.

Declarations

Conflict of interest The authors certify that they have no conflict of interest in the subject matter or materials discussed in this manuscript.

References

1. Alam, M.S., Kwon, K.C., Alam, M.A., Abbass, M.Y., Imtiaz, S.M., Kim, N.: Trajectory-based air-writing recognition using deep neural network and depth sensor. Sensors (2020). https://doi.org/10.3390/s20020376
2. Bhambri, S., Muku, S., Tulasi, A., Buduru, A.B.: A survey of black-box adversarial attacks on computer vision models (2020)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (sp), pp. 39–57. IEEE (2017)
4. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The UCR time series archive (2019)
5. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9185–9193 (2018)
6. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1625–1634 (2018)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. (2014) arXiv preprint arXiv:1412.6572
8. Guillaume, A., Vrain, C., Wael, E.: Time series classification for predictive maintenance on event logs. (2020) arXiv preprint arXiv:2011.10996
9. Huan, Z., Wang, Y., Zhang, X., Shang, L., Fu, C., Zhou, J.: Data-free adversarial perturbations for practical black-box attack. In: Lauw, H.W., Wong, R.C.W., Ntoulas, A., Lim, E.P., Ng, S.K., Pan, S.J. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 127–138. Springer International Publishing, Cham (2020)
10. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Adversarial attacks on deep neural networks for time series classification. In: 2019 International Joint Conference on Neural Networks (IJCNN) (2019)
11. Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: Inceptiontime: finding alexnet for time series classification. Data Mining and Knowledge Discovery **34**(6), 1936–1962 (2020)
12. Jiang, Y., Ma, X., Erfani, S.M., Bailey, J.: Dual head adversarial training (2021)
13. Kaushik, S., Choudhury, A., Sheron, P.K., Dasgupta, N., Natarajan, S., Pickett, L.A., Dutt, V.: Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. Frontiers in Big Data **3**, 4 (2020)
14. Kumar, N., Jha, G.K.: A time series ann approach for weather forecasting. Int J Control Theory Comput Model (IJCTCM) **3**(1), 19–25 (2013)
15. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: Artificial Intelligence Safety and Security, pp. 99–112. Chapman and Hall/CRC, Boca Raton (2018)
16. Li, J., Schmidt, F., Kolter, Z.: Adversarial camera stickers: a physical camera-based attack on deep learning systems. In: International Conference on Machine Learning. pp. 3896–3904. PMLR (2019)
17. Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J.: Defense against adversarial attacks using high-level representation guided denoiser. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1778–1787 (2018)
18. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning (2017)
19. Rathore, P., Basak, A., Nistala, S.H., Runkana, V.: Untargeted, targeted and universal adversarial attacks and defenses on time series. In: 2020 International Joint Conference on Neural Networks (IJCNN) (2020)
20. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. **23**(5), 828–841 (2019)
21. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. (2013) arXiv preprint arXiv:1312.6199
22. Wu, L., Zhu, Z., Tai, C., et al.: Understanding and enhancing the transferability of adversarial examples (2018)
23. Xie, C., Tan, M., Gong, B., Yuille, A., Le, Q.V.: Smooth adversarial training (2021)
24. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.: Mitigating adversarial effects through randomization. (2017) arXiv preprint arXiv:1711.01991