Smooth Perturbations for Time Series Adversarial Attacks

Gautier Pialla[1](✉), Hassan Ismail Fawaz[1], Maxime Devanne[1], Jonathan Weber[1], Lhassane Idoumghar[1], Pierre-Alain Muller[1], Christoph Bergmeir[2], Daniel Schmidt[2], Geoffrey Webb[2], and Germain Forestier[1,2]

[1] Université de Haute-Alsace, Mulhouse, France
`firstname.lastname@uha.fr`
[2] Monash University, Melbourne, Australia
`firstname.lastname@monash.edu`

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 find that smooth perturbations are harder to detect by the naked eye. We also show how adversarial training can improve model robustness against this attack, thus making models 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 [12], for weather forecasting [13] and for predictive maintenance [7].

Time series classification (TSC), refers to the task of classifying time series according to the presence or not of phenomena. Szegedy et al. [6] 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 on Figure 1.

As adversarial attacks are a vulnerability present in every neural network, many attacks were proposed but first for image classification tasks. It is necessary

This is the author's version of an article published at PAKDD 2022. The final authenticated version is available online at: https://doi.org/10.1007/978-3-031-05933-9_38

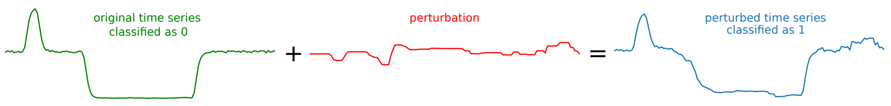


Fig. 1: Scheme of adversarial attack. Time series from the BME dataset, perturbation generated with SGM, not represented at scale.

to study them in order to assess the robustness of the models, and to prevent them on critical systems. For example, Eykolt et al. [5] showed an application on real-world road sign classification, which is an obvious threat for autonomous vehicles.

Fawaz et al. [9] 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 sightly changing the value of one or few pixels, an image will always look the same and have the same appearance. Theses 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 TSC it is not, because time series data are more complex to analyze.

The attacks introduced by Fawaz et al. [9] are effective to perturb time series of the UCR Archive [3]. But when we look at their visual appearance, it is sometimes easy to distinguish the disturbed series from the original ones. Indeed, theses 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 introduced 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. 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. 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
- A benchmark of our two methods along with BIM over the UCR archive
- We showed how smoothed perturbations are harder to be detected
- We showed 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. [20] 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. [6] proposed a single step attack called Fast Gradient Sign Method (FGSM). Then, Kurakrin et al. [14] presented the Basic Iterative Method (BIM), an iterative version of FGSM. Inspired by them, many similar attacks were proposed, like M-IGSM [4] or vr-IGSM [21].

Other approaches where studied, like adding black and white strips on stop signs [5] or stickers on objects [15]. 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 [16], randomization [23] and adversarial training [22,11].

Adversarial training trains a model using both normal and perturbed samples. Rathore et al. [18] shows 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. [9] 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. [19] 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 presented in [17,1], don't use any knowledge of the architecture, the parameters or the weights of the model. They have also no access of the datasets used for the training. Huan et al. [8] 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 & Wagner method [2]. 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, 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. [14] 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-neighbourhood.

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

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 by $f(\mathbf{x})$. The noise clipping is done for $\mathbf{r} = \mathbf{x}^{adv} - \mathbf{x}$ as follow:

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

By adding iterations, BIM becomes more effective than FGSM to perturb time series. But BIM requires clipping in order to control the amount of the noise. This method had two main disadvantages. First, clipping the noise in such way often produce sawtooth shapes between $-\epsilon$ and $+\epsilon$ as we can see on Figure 5. 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 to value of ϵ. By doing this, the saw-tooth shapes will be harder to be noticeable, but this will result in a lower attack success rate. This trade-off prevents the perturbation that are both hard to detect and have a high attack success rate.

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:

$$\max D_{KL}(f(\mathbf{x}), f(\mathbf{x}')) \equiv \sum^{c} f(\mathbf{x}) \log \frac{f(\mathbf{x})}{f(\mathbf{x}')}, \tag{2}$$

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:

$$\max\left\{\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x}')) - \|\mathbf{x} - \mathbf{x}'\|_2\right\}, \tag{3}$$

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:

$$\min\left\{-\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x} + \mathbf{r})) + \|\mathbf{r}\|_2\right\} \tag{4}$$

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

$$\mathbf{x}^{adv} = \min\left\{-\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x} + \mathbf{r})) + \alpha\|\mathbf{r}\|_2\right\} \tag{5}$$

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:

$$\min\{-\mu D_{KL}(f(\mathbf{x}), f(\mathbf{x} + \mathbf{r})) + \alpha\|\mathbf{r}\|_2 + \lambda \sum_{i=1}^{T-1} \|r_i - r_{i+1}\|_1\} \tag{6}$$

In equation 6, $\|.\|_1$ 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 the gradient descent by computing the gradient with respect to $\mathbf{r}$ (which will be initialized randomly).

5 Experimental Setup

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

5.1 Classifier and Datasets

We used InceptionTime [10] for all our experiments. InceptionTime is a TS classifier, that was the state-of-the-art model on the UCR archive, when published in 2019. 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[3]. 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 set. When generating adversarial samples, we used the samples of the test set, as the model has only been trained on the training set.

5.2 Reproductibility

The code used and all our results are publicly available in our companion repository [3].

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.

5.4 Comparison Metrics

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 ($f(\mathbf{x}) \neq f(\mathbf{x}^{adv})$).

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 explain 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 times 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.

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 attack.

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

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 will perform an adversarial training, in order to propose a counter measure against SGM attacks.

6.1 SGM benchmark

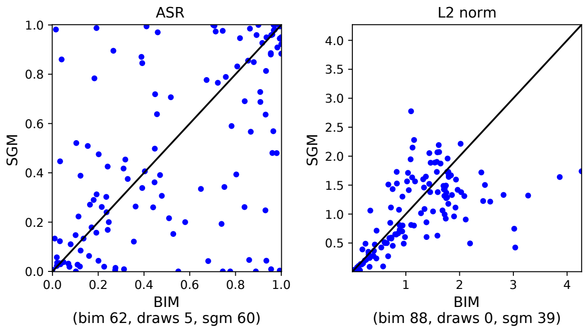


Fig. 2: Win/Draw/Loss diagram. BIM vs SGM. On the left: average success rate, on the right: L2 norm of the perturbation

Figure 2 represents a Win/Draw/Loss diagram comparing BIM and SGM. Each blue dot represent 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 Figure 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 datasets that BIM can not 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 GM with SGM. We can see that for almost all datasets, GM has a better ASR than SGM. This shows that the sawtooth and spikes which can only be produced by GM are decisive elements in order to perturb a TSC.

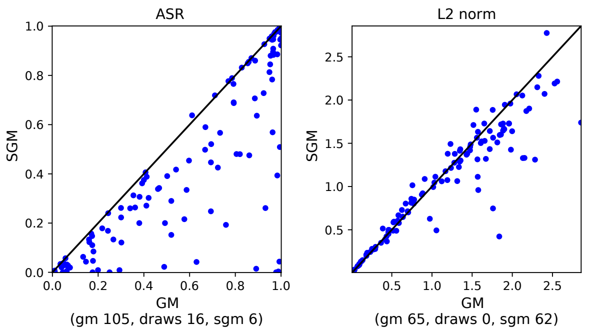


Fig. 3: Win/Draw/Loss diagram. GM vs SGM. On the left: average success rate, on the right: L2 norm of the perturbation

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 adding 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 4 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 the 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.

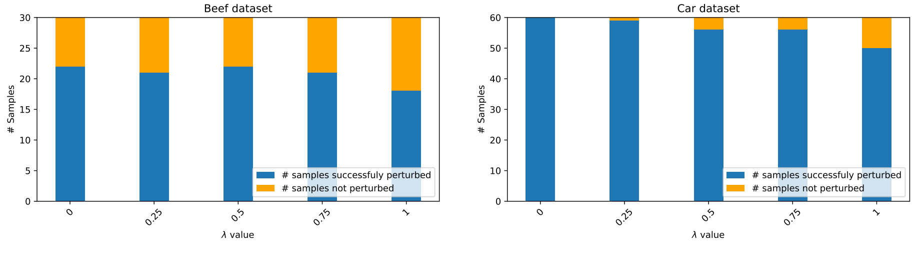
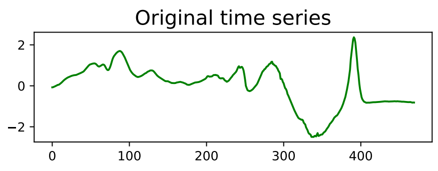


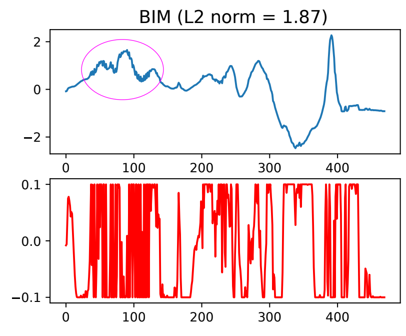
Fig. 4: Varying SGM's λ parameter. For each value of λ is displayed the number of samples successfully perturbed (blue) or not (orange).

6.3 Visual Comparison

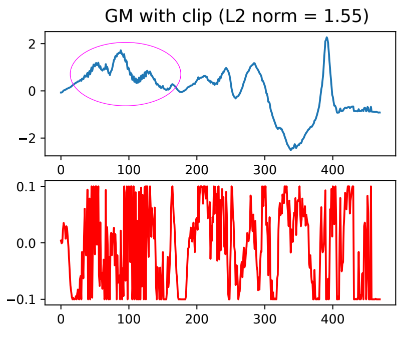
In order to remain undetectable by the naked eye, an attack performed on a time series must be as smooth as possible. As we did not find any suitable metric to



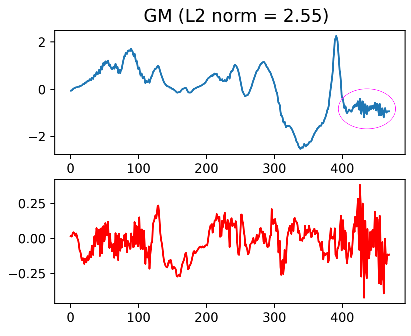
(a) Original time series from the Beef dataset



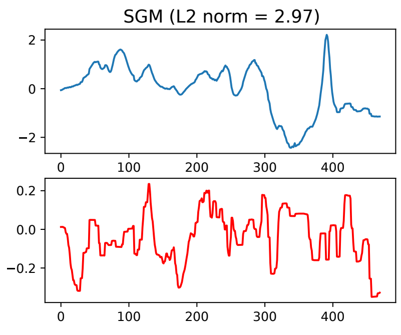
(b) BIM adversarial attack.



(c) GM adversarial attack.



(d) GM without clip adversarial attack.



(e) SGM adversarial attack

Fig. 5: Time series from the Beef dataset. All methods perturbed time series (blue) and generated noise (red). The purple circles show the presence of saw-tooth on the perturbed time series.

assess the smoothness of a time series, 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 which is successfully perturbed by all the attacks.

In this example, shown on Figure 5, 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 reduce indeed the amount of noise and the visual impact, but not sufficiently enough. SGM is the only attack that produced an adversarial sample with a perturbation that is not noticeable when judging with the naked the eye.

But 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 in average for a given dataset, this is not necessary true when we look at each sample independently.

6.4 Adversarial Training

Figure 6 presents the results of adversarial training using SGM adversarial samples. On the left scatter plot, we compare the classification accuracy of the basic InceptionTime compared to the accuracy of InceptionTime with adversarial training. In most cases, adversarial training led to a decrease of accuracy.

The right scatter plot, shows that the model trained with adversarial training led to zero ASR for most of the datasets. This huge drop, shows the effectiveness of adversarial training against SGM attacks.

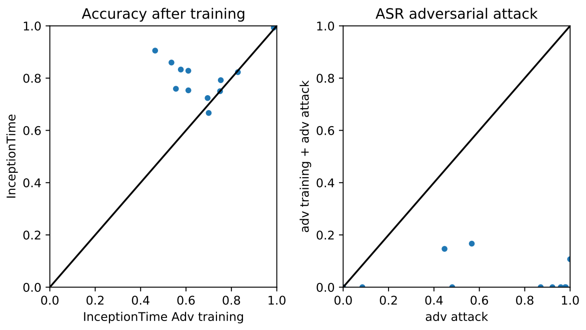


Fig. 6: Adversarial training results of 13 randomly chosen datasets.

7 Conclusion

In this paper, we explained that adapting adversarial attacks from image classifiers to time series classifier 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), a well known adversarial attack, as a baseline to have a benchmark over the entire UCR archive. We showed that GM, has the higher success rate on perturbing an InceptionTime classifier, followed by BIM and SGM.

Through examples, we illustrated that GM, like BIM produces perturbations which have recognizable patterns like spikes and sawtooth. On 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. Smoothing the noise makes it harder to differentiate 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 showed that adversarial training is an effective way of countering SGM attacks.

For the future works, we would like to find a metric that can measure the smoothness of a time series. This would help in order to find a new smooth adversarial attack that is better than SGM at fooling time series classifiers.

Acknowledgment

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 University of Haute-Alsace. 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.

References

1. Bhambri, S., Muku, S., Tulasi, A., Buduru, A.B.: A survey of black-box adversarial attacks on computer vision models (2020)
2. 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)
3. 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)
4. 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)

5. 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)
6. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
7. Guillaume, A., Vrain, C., Wael, E.: Time series classification for predictive maintenance on event logs. arXiv preprint arXiv:2011.10996 (2020)
8. 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)
9. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Adversarial attacks on deep neural networks for time series classification. 2019 International Joint Conference on Neural Networks (IJCNN) (Jul 2019)
10. 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 (Sep 2020)
11. Jiang, Y., Ma, X., Erfani, S.M., Bailey, J.: Dual head adversarial training (2021)
12. 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)
13. 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)
14. 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 (2018)
15. 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)
16. 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)
17. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning (2017)
18. Rathore, P., Basak, A., Nistala, S.H., Runkana, V.: Untargeted, targeted and universal adversarial attacks and defenses on time series. 2020 International Joint Conference on Neural Networks (IJCNN) (Jul 2020)
19. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (Oct 2019)
20. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
21. Wu, L., Zhu, Z., Tai, C., et al.: Understanding and enhancing the transferability of adversarial examples (2018)
22. Xie, C., Tan, M., Gong, B., Yuille, A., Le, Q.V.: Smooth adversarial training (2021)
23. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.: Mitigating adversarial effects through randomization. arXiv preprint arXiv:1711.01991 (2017)