ITC 2/54
Information Technology and Control
Vol. 54 / No. 2/ 2025
pp. 613-628
DOI 10.5755/j01.itc.54.2.37758

Learn from Adversarial Examples: Learning-Based Attack on Time Series Forecasting

Received 2025/01/20 | Accepted after revision 2025/04/07

HOW TO CITE: Xiao, Y., Yang, Z., Zou, Q, Zhang, P. (2025). Learn from Adversarial Examples: Learning-Based Attack on Time Series Forecasting. *Information Technology and Control,* 54(2), 613-628. https://doi.org/10.5755/j01.itc.54.2.37758

Learn from Adversarial Examples: Learning-Based Attack on Time Series Forecasting

Youbang Xiao
School of Information Science and Technology, North China University of Technology, Beijing, China

Zhongguo Yang*
Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data

Qi Zou
Brunel London School, North China University of Technology, Beijing, China

Peng Zhang
School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing, China

Corresponding author: yangzhongguo@ncut.edu.cn

Adversarial attack in Time Series Forecasting(TSF) has been a topic of growing interest in recent years. While some black box attack methods have been proposed for TSF, they require continuous query to the target model. And the computational costs increase as model and data complexity grows. In fact, The perturbations generated by these methods have certain patterns, especially constrained in $L0$ norm. Those patterns can be captured and learned by a model. In this study, we proposed Learning-Based Attack(LBA), a novel black box adversarial attack method for TSF tasks, focusing on adversarial example, the perturbed data. By utilizing a model to learn adversarial examples and generate a similar one, we can achieve a comparable performance with the original attack methods while significantly reducing the number of queries to the target model, ensuring high efficient and stealthiness. We evaluate our method through several public datasets. In this paper, we learn the adversarial samples attacked by n-Values Time Series Attack(nVITA), a sparse black box attack for TSF. The results show that we can effectively learn the attack information and generate similar adversarial samples with lower computational overhead, thus achieving the stealthiness and efficiency of the attack. Furthermore, we also verify the transferability of our method and found its applicability to attack other models. Our code is available on Github[1].

KEYWORDS: Time Series Forecasting, Adversarial Attack, Deep Learning

[1]https://github.com/Six6stRINGs/LearningBased_Atk

1. Introduction

Time series Forecasting(TSF) serves as a cornerstone in numerous fields, offering a dynamic perspective on evolving phenomena. Its intrinsic ability to capture temporal dependencies has fueled its applications across diverse domains, ranging from finance [1], economics, healthcare [2] and environmental sciences. Furthermore, the advent of several sophisticated deep learning models has marked a significant leap in addressing TSF challenges, enhancing efficiency and precision [3], [4].

However neural networks have been proved to be vulnerable to some special noise called *adversarial attack*. Szegedy et al. [5] discovered the phenomenon that neural networks could make wrong decision at a high probability by adding some kinds of human-imperceptible perturbations to the original datasets. Subsequently, many methods were proposed to generate adversarial examples such as FGSM [6], BIM [7], DeepFool [8]. Fig. 1 shows the adversarial attack on TSF tasks. The adversarial examples generated by these methods can make the target model generate wrong trend of predictions. The target model is unreliable under such adversarial attacks.

Some of the existing studies on adversarial attack in TSF tasks rely on gradient-based methods [9]–[11], which are not practical in real conditions. Additionally, global perturbations on time series data are more visible than those on images [11]. Thus, local perturbations are more suitable for TSF tasks. Fig. 2 shows the comparison of the adversarial attack methods on TSF tasks. (a) and (b) use the same sample with different attack methods. (a) illustrates the FGSM attack, where the entire dataset is subject to perturbations, leading to a sawtooth-like pattern that significantly deviates from the original data. This global perturbation is more conspicuous and can potentially be detected more easily due to its broader impact across the dataset. In contrast, (b) depicts the *n*VITA attack, which introduces perturbations at only a few critical points within the data window. This approach results in a less dramatic but targeted deviation in the adversarial example.

Sparse attack methods based on sensitive points are a topic of growing interest in recent years. Su et al. [12] proposed a method called One Pixel Attack, which can generate adversarial examples by perturbing a single pixel in the image classification tasks. Croce et al. [13] proposed l_1-APGD, an adaptive form of PGD [14] which is highly effective even with a small budget of iterations. Several sparse black box methods [12], [15], [16] turn the adversarial attack into a search problem — to find the local optimal perturbations. Moreover,

Search problem can be solved by optimization algorithms, such as Differential Evolution(DE) [17]. By utilizing such algorithms, the attacker can get a sequence of *sensitive points* to perturb and generate adversarial examples. In order to achieve a high attack performance, they often require a large number of queries to the target model. The computational cost of these methods increases rapidly as the complexity of the model and data grows.

Figure 1

Adversarial attack on Time Series Forecasting. The perturbations are small and hard to be detected by human, but can fool the target model to generate wrong predictions.

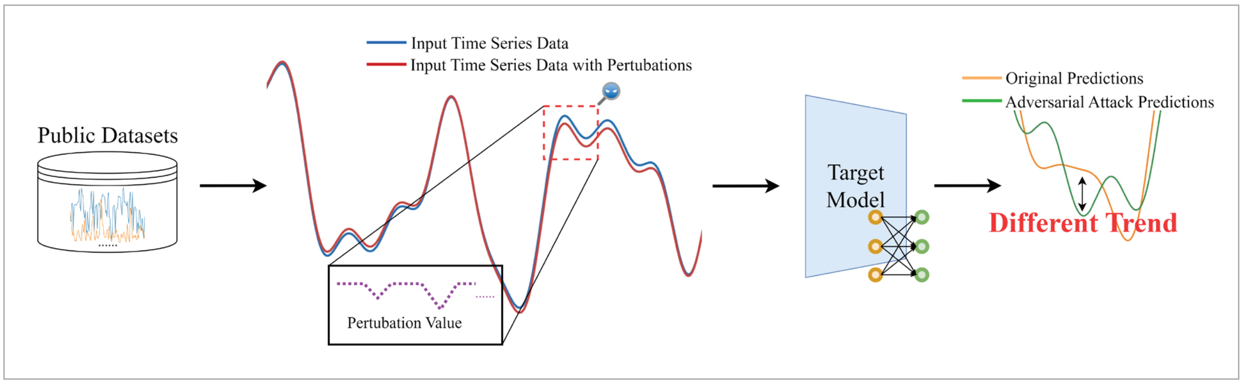
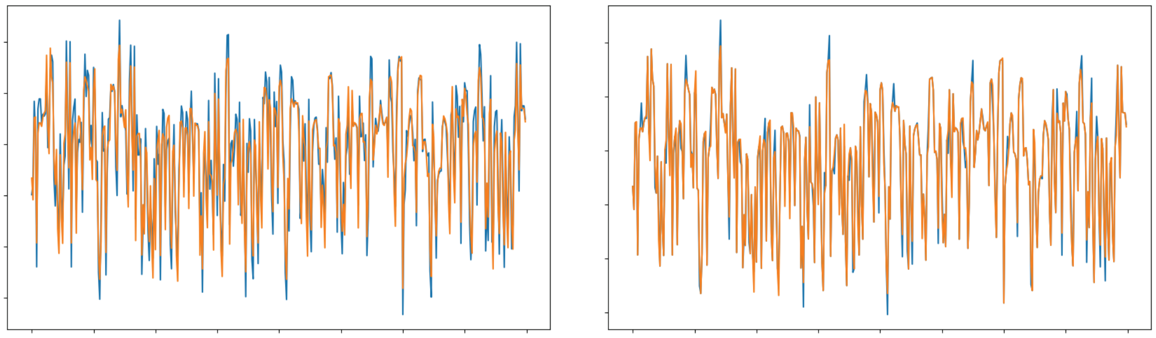


Figure 2

A comparison of the adversarial attack methods for TSF tasks. Time series data with orange line refers to the original data, blue line refers to the adversarial examples. Note that the figures above are data combined with multiple window data in a single feature. In TSF tasks, data are usually split into windows to predict the future values. Thus sparse attack means perturb few sensitive points in each window data, while global attack means perturb all the points in every window data.



(a) Comparison of global attack method(FGSM) (b) Comparison of sparse attack method(nVITA)

So, the question is: *Since sparse attack methods involve fewer perturbations, might it be possible to learn the adversarial examples and generate similar ones?* In the context of image data, the volume of a single instance is typically quite substantial, which can complicate the process of identifying and learning adversarial examples. By contrast, in TSF tasks, time series data are split into multiple windows. This technique significantly reduces the scale of each individual window, making it feasible to learn the adversarial examples. In this study, we proposed a novel black box adversarial attack method for TSF tasks, focusing on adversarial example, the perturbed data. The underlying patterns and characteristics of these adversarial examples can be effectively captured and generalized by a learning model. By training a small model on known adversarial attack patterns from a given target black box attack method, we are able to generate some similar adversarial examples without the need for direct queries to the target model.

To the best of our knowledge, none of study has been conducted on such way to generate adversarial examples. This Learning-Based Attack(LBA) method allows us to rapidly generate comparable adversarial examples with the target attack method. We evaluate our method through some public datasets. The obtained results demonstrate the effectiveness of our method. We also verified the transferability of our method and found its applicability to attack other models. Our contributions can be summarized as follows:

1 We demonstrate that adversarial examples can be learned and generated by a model on TSF tasks.

2 A novel perspective on adversarial attack. We focus on the adversarial examples, the perturbed data.

3 Query-Less and Effectiveness. By learning the existing adversarial examples, we can achieve a comparable attack performance with the target attack method while significantly reducing the number of queries to the target model, ensuring high efficient and stealthiness.

2. Related Works and Background

In this section, we will introduce an overview of TSF tasks and adversarial attack.

a *Deep Learning on Time Series Forecasting*

TSF tasks holds a pivotal position in data analysis and data mining, leveraging advanced models to extract valuable insights from sequential data. Its significance lies in providing accurate predictions, enabling informed decision-making across various domains. TSF tasks can be described as follows [18]:

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, x_{i,t-k:t}, s_i) \quad (1)$$

where $\hat{y}_{i,t+1}$ is the predicted value of the i-th time series at time $t+1$, $y_{i,t-k:t} = \{y_{i,t-k,}, \cdots, y_{i,t}\}$, $x_{i,t-k:t} = \{x_{i,t-k,}, \cdots, x_{i,t}\}$ represent the true values of the i-th time series and the corresponding time series data, which composed a k window data, s_i is the static metadata of the i-th time series, $f(X)$ is the prediction function of the model.

The equation adove focuses on the univariate TSF tasks. For Multivariate Time Series (MTS) forecasting tasks, some adjustments are required to accommodate the multiple variables. The essence, however, remains the same — predicting the future values based on historical information.

Deep Learning has made significant progress over the past few years. It has been widely used in TSF tasks and achieved obvious success. Convolutional Neural Networks (CNN), due to their ability to extract local features from data, have demonstrated effectiveness in certain TSF tasks. Recurrent Neural Network(RNN) [19] can capture the temporal dependencies in sequential data, making it suitable for TSF tasks. But long term dependencies are hard to learn for RNNs due to the limited information stored in the hidden state. Hence, Long Short- Term Memory(LSTM) [20] was developed, which is effective in learning and retaining information over extended sequences with its memory cell structure. Cui et al. [4] proposed a novel model called Traffic Graph Convolutional LSTM(TGC-LSTM) which can learn the interactions between different roadways and then forecast the traffic state. Gated Recurrent Unit(GRU) [21], featuring gate mechanisms, provides a balance between model complexity and efficiency, finding valuable applications in TSF tasks. It can achieve comparable performance with LSTM while being computationally more efficient. Li et al. [3] evaluated the water quality through dissolved oxygen prediction using GRU with high performance and lower computational cost than LSTM. In this study, we will extend our method based on CNN, LSTM and GRU.

b *Adversarial Attack on Time Series Forecasting*

Adversarial attack has been a topic of growing interest in recent years. Adversarial attack can be classified into three categories: white box attack, black box attack, and grey box attack. White box attack assumes that the attacker has full access to the target model, including its architecture and all the parameters. Black box attack assumes that the attacker has no information of the target model [22]. Gray box attack is a combination of white box attack and black box attack which means the attacker has partial information of the target model.

After the discovery of adversarial attack [5], many methods have been proposed to generate adversarial examples. Good- fellow et al. [6] proposed Fast Gradient Sign Method(FGSM) to effectively generate adversarial examples. FGSM is a one- step method which can be described as follows:

$$X_{adv} = X + \beta \cdot sign(\nabla_X J(\theta, X, y)) \quad (2)$$

where X represents the original data, X_{adv} represents the adversarial example, β means the factor of the perturbation, $J(\theta, X, y)$ means the loss function of the model, θ is the parameters of the model, y is the label of the original data.

Later, Kurakin et al. [7] proposed a Basic Iterative Method(BIM) to generate adversarial examples by iteratively applying FGSM.

Despite the existing gradient-based methods have been proposed to generate adversarial examples for TSF tasks [10]

[9] which can also achieve a high attack success rate, the noise they generated is not stealthy enough, making it easier to be detected [23], [24].

Wu et al. [11] proposed a method called Adversarial Time Series Generator(ATSG) to generate adversarial examples based on the gradient of the target model. By using the Adversarial Attack with Importance Measuring(AAIM) strategy, they can generate adversarial examples with smaller perturbation.

***n*-Values Time Series Attack(*n*VITA)**, a black box sparse attack method which can both make targeted and non-targeted attack proposed by Chen et al. [15]. It's based on DE [17], a population-based meta-heuristic search algorithm for global optimization. DE algorithm emulates the processes of biological evolution, including mutation, crossover, and selection, in its search for the optimal solution. It maintains a population of candidate solutions, where each individual corresponds to a potential solution to the problem. DE proceeds by mutating individuals to generate new candidate solutions, then combining these new solutions with the existing population through crossover operations. Finally, selection is used to determine which individuals will continue to the next generation based on their fitness. Given a time series data X, nVITA describes the n perturbations as a triple (t, fe, p) [15] as follows:

$$\eta_{nVITA} = [t_1, fe_1, p_1, t_2, fe_2, p_2, \cdots, t_n, fe_n, p_n] \quad (3)$$

where t refers to the timestamp of the data X, fe refers to the feature, p refers to the perturbation, The length of η_{nVITA} is $3n$.

By randomly generating parent samples $Gen_1 = (\eta_1, \cdots, \eta_s)$, DE will generate the offspring samples iteratively to improve the quality through crossover and mutation. Whether the offspring samples can be added to the next generation is determined by the fitness function. In nVITA, Mean Square Error(MSE) is used as the fitness function for *non-targeted attack* and the Absolute Error(AE) is used as the fitness function for *targeted attack*.

By specifying the n, we can control the number of altering timesteps and features. But as n increases, the computational cost of nVITA increases rapidly, which also means it queries the target model continuously to find a better solution.

Table 1 shows the comparison of several adversarial attack methods for TSF tasks. We will introduce our method(LBA) in the next section.

Table 1

Several adversarial attack methods for TSF tasks. The number of iterations for the BIM method is generally set within the range of tens to hundreds. nVITA method based on DE algorithm. Thousands of iterations may required to achieve a high attack success rate.

Method	Type	Perturbation	Theory	Query
FGSM/ ATSG	White Box	Global	Gradient	Once
BIM	White Box	Global	Gradient	Low
nVITA	Black Box	Local	DE	High
LBA	Black Box	Local	Deep Learning	Low

3. Methodology

In this section, We first formulate the adversarial attack on TSF tasks. Subsequently, we will introduce our method and attack process in detail.

a *Problem Definition*

LBA relies on a learning model, which we denote as f_{learn}, to learn and generate adversarial examples. We designate D_{adv} to represent the dataset of adversarial examples and f_{tar} to indicate the target model that we intend to attack.

A time series data can be describe as $X = (x_1, \cdots, x_n)$, each x_i refers to the value of the i-th timestamp. The magnitude of perturbation ε is controlled via a factor β. ε is defined as

$$\varepsilon = \beta \cdot \left(\frac{1}{T}\sum_{i=1}^{T}(x_i^{max} - x_i^{min})\right) \quad (4)$$

where x_i^{max} and x_i^{min} separately represent the maximum and minimum value of a time series of the total T items.

The adversarial attack on TSF tasks can be formulated as follows:

$$\begin{aligned} &\text{maximize} && \left\|Y - \hat{Y}\right\| \\ &\text{subject to} && f_{tar}(X) = Y, f_{tar}(\hat{X}) = \hat{Y}\ , \\ &&& \left\|X - \hat{X}\right\| \leq \varepsilon \end{aligned} \quad (5)$$

where X represents the time series data, Y representsthe corresponding prediction, $\hat{X}$ represents the adversarial example, $\hat{Y}$ represents the corresponding prediction of the adversarial example.

From the equation above, it is evident that the object of adversarial attack is to maximize the difference in predictions between the original data and the adversarial example while keeping the perturbation as small as possible.

b *Learning-Based Attack Framework*

LBA employs a model to learn the adversarial attack patterns and generate the adversarial examples. Fig. 4 shows the whole adversarial attack framework.

1 **A Target Black Box Attack Method:** To learn adversarial attack patterns, we first adopt a *target black box attack method* to generate adversarial examples, producing sparse attacks. In this study, we utilize the nVITA as the target black box attack

method to generate adversarial examples. The generated adversarial examples, crafted by the target black box attack, are then systematically collected and utilized as training data for the learning model.

2 **Phase 1: Collect Adversarial Examples Datasets:** As our method is based on learning the adversarial attack patterns, we need to collect the adversarial examples as the training datasets. The components of datasets is crucial in training phase. For our adversarial example datasets, we contain 3 sequences: the original time series data X, the location of sensitive points Loc, and the perturbation values P. Each sequence is individually stored in the datasets. For Loc and P, they can be formulated as follows:

$$Loc_i = [j \mid \hat{X}_i^j - X_i^j \neq 0], Loc_i \in Loc$$
$$P_i = [\hat{X}_i^j - X_i^j \mid \hat{X}_i^j - X_i^j \neq 0], P_i \in P \quad (6)$$

where j refers to the timestamp index in the i-th sample where the perturbation occurs.

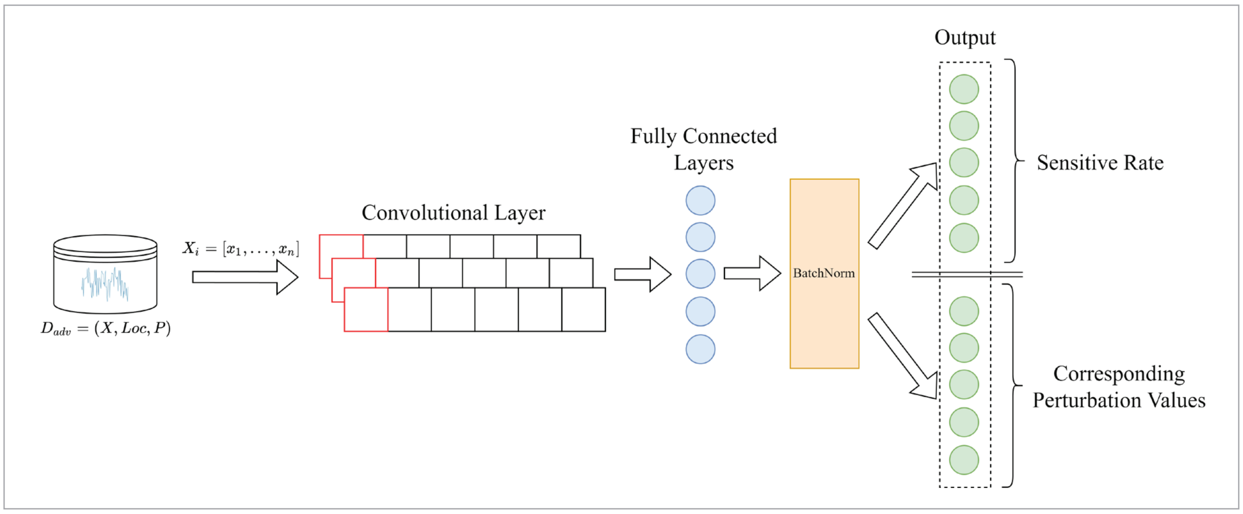
By subtracting adversarial samples from the original data, we can easily and quickly obtain important information about adversarial attacks — *sensitive* points and *perturbation values*. We collect them and the original data all individually to form our adversarial examples datasets as the training datasets for our learning model. This datasets also can be described as a triple $D_{adv} = (X, Loc, P)$.

3 **Phase 2: Train Learning Model:** Our learning model is designed with a dual-purpose architecture, playing a critical role in the LBA framework. It is first trained to identify sensitive points within the input data that are susceptible to attacks. At the same time, it generates perturbations targeted at these sensitive points. This two-pronged approach enables the model to effectively generate adversarial examples for time series forecasting tasks, enhancing its ability to simulate realistic and impactful adversarial scenarios. We utilized CNN as our learning model. CNN is widely used in TSF tasks, as it can capture the local patterns and dependencies within the data. Fig. 3 shows its architecture for generating the adversarial examples.

To enhance the model's performance and prevent over- fitting, we incorporate Bayesian convolutional layers. [25] Bayesian convolutional layers introduce a probabilistic framework, which not only enables the model to capture the underlying patterns but also allows it to estimate and account for the uncertainty associated with each parameter.

Unlike normal CNN designed to tackle only single task, our learning model is uniquely designed to simultaneously train both on sensitive points and perturbation values. The output of the learning model f_{learn} is a pair of sequence: $f_{learn}(X) = (Rate, P)$, $Rate$ refers to the sensitive rate of each timestamp in the time series data. Table 2 shows all the layers of our CNN model.

Figure 3
Our f_{learn} is a CNN model. We designed the model to simultaneously train on sensitive points and perturbation values.



As can be seen, we used a very small CNN network, which enables our training and inference processes to be very fast. Meanwhile, the results indicate that our model can effectively learn the patterns between sensitive points and perturbation values.

Sensitive Points The timestamp of the attacked point is the sensitive point. We need to predict sensitive rate of each timestamp in the time series data. By the Loc sequence in the D_{adv}, we can train the model to predict the sensitive points of the original data. Then we select the top k points as the sensitive points $\hat{Loc}$ we need to perturb. k will be automatically adjusted to match the number of sensitive points with the target black box method. We use Cross Entropy Loss as the loss function to train the model to predict the sensitive points.

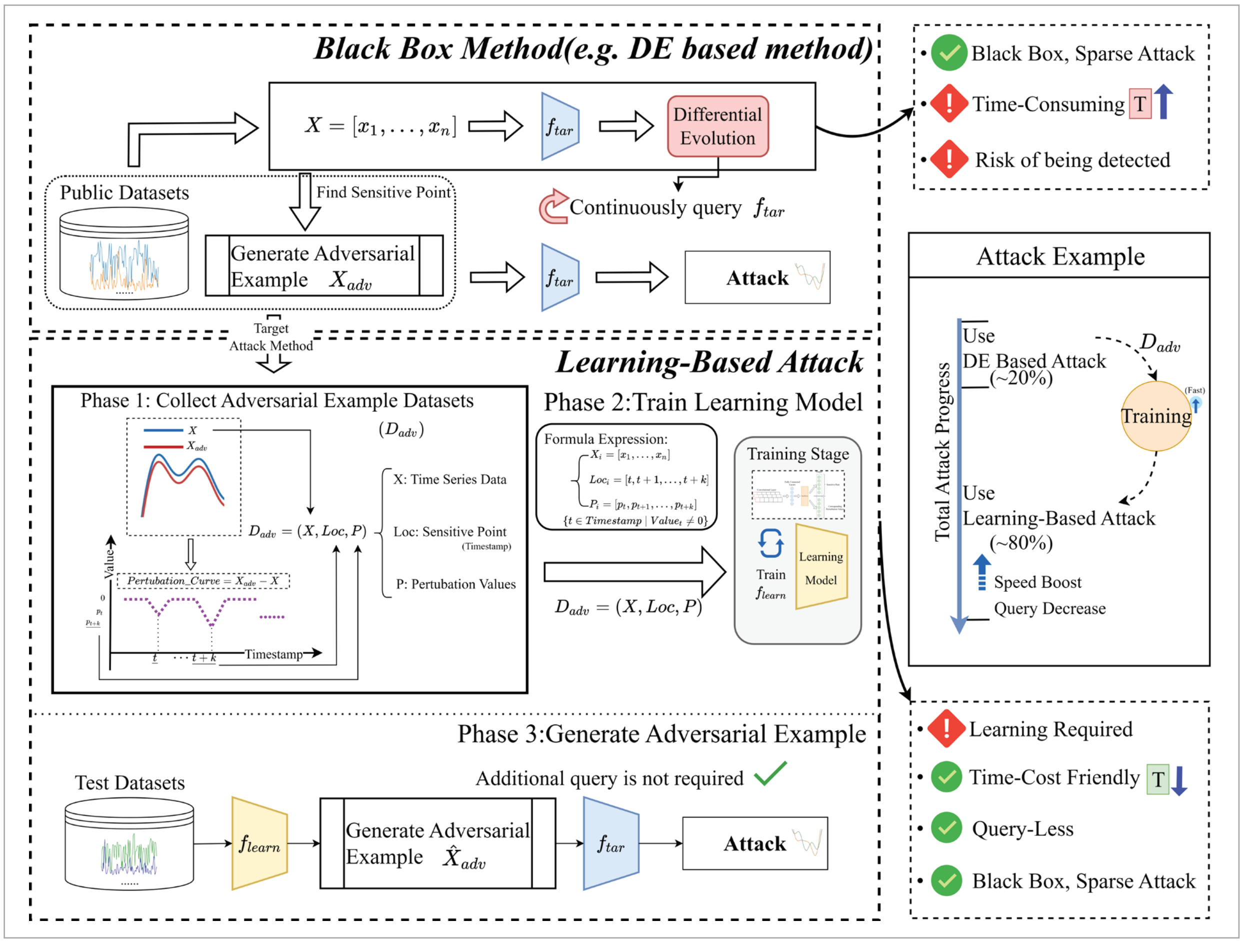
Table 2

CNN Model Architecture of our LBA attack. We believe that a relatively simple convolutional network can learn the information of adversarial attacks — the sensitive points and disturbance values. This is a multitasking model with two independent output layers.(Layer 5 and 6) Specific parameters can be found in our code.

Layer	Type
1	Bayesian Conv Layer
2	Bayesian Conv Layer
3	Fully Connected Layer
4	Batch Normalization Layer
5	Layer of Sensitive Rate
6	Layer of Perturbation Values

Figure 4

Adversarial attack framework of Learning-Based Attack. LBA needs a target attack method to learn to adversarial attack patterns and then generate adversarial examples without any query to the target model.



Perturbation Values The perturbation values are the difference between the original data and the adversarial examples. To learn the perturbation values of the target black box method, We use MSE as the loss function to train the model to predict the perturbation values.

4 **Phase 3: Generate Adversarial Examples:** Once the learning model is trained, it can be utilized to generate adversarial examples without making any further queries to the target model.

Fig. 5 describes the output mechanisms of our learning model and the process by which adversarial examples are crafted. A critical design choice in our approach was to not task the learning model with directly generating adversarial examples. If our model were to generate adversarial examples outright, it would have to learn not only the characteristics of adversarial samples but also the features of the original data. This dual learning objective would complicate the learning our learning model is designed to output the sensitivity values and corresponding perturbation values for time series data. By focusing solely on these aspects, the learning task becomes more straightforward, allowing the model to more reliably produce the adversarial effects we aim to achieve.

Once the sensitive rate is identified, we apply the *softmax* function followed by a *top-k selection* method to determine the final sensitive points. By adding the perturbation values generated by our model to the corresponding points in the original data, we can effectively create the adversarial examples that are central to our attack strategy.

The adversarial examples generated by the learning model can be formulated as follows:

$$\begin{aligned} &\hat{X}_{adv} = [x_1, \cdots, \hat{x}_t, \cdots, x_n] \\ &\text{subject to } \hat{x}_t = x_t + \delta * \hat{p}_t, t \in \hat{Loc} \end{aligned} \tag{7}$$

where $\hat{Loc}$ refers to the timestamps of top k sensitive points. $\hat{p}_t$ refers to the predict perturbations of the corresponding sensitive points. δ is the factor of the perturbation. As the perturbations are small enough, δ is usually set around 1. We can also adjust the parameter δ to a higher value to enhance the effectiveness of the adversarial examples.

4. Experiments

We conduct a series of experiments to evaluate the performance of our method. All experiments with local model are conducted on the same GPU device. Repeated experiments are conducted to ensure accuracy, as well as to avoid random errors. Our code is available on Github.

a *Datasets*

We evaluate our method through several public datasets to validate the effectiveness and efficiency of our method. Table 3 shows the datasets we used in our experiments.

Figure 5

The output of the learning model and the process of generating adversarial examples.

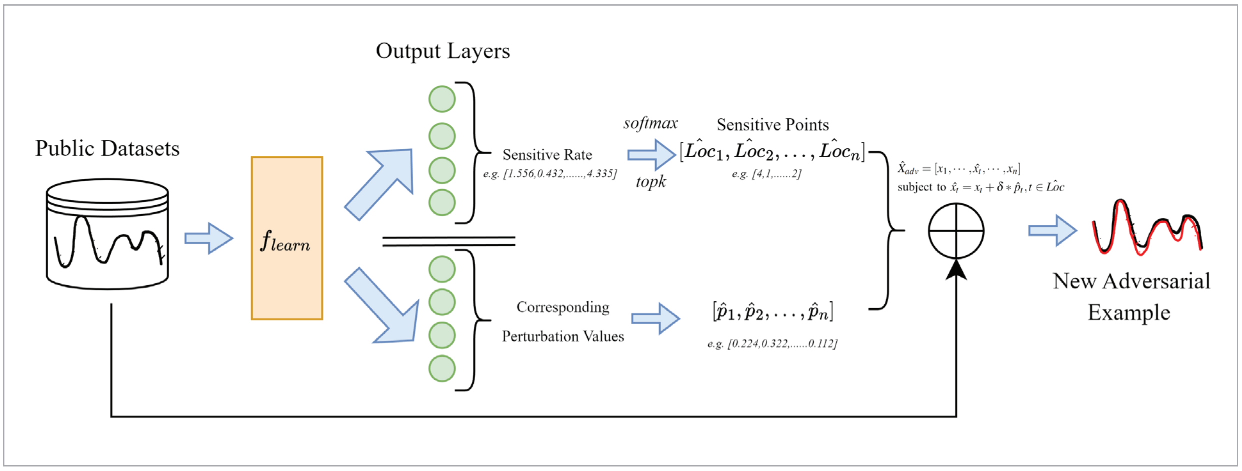

Output Layers
Public Datasets
f_{learn}
Sensitive Rate
e.g. [1.556,0.432,......,4.335]
softmax
topk
Sensitive Points
$[\hat{Loc}_1, \hat{Loc}_2, \ldots, \hat{Loc}_n]$
e.g. [4,1,......2]
Corresponding Perturbation Values
$[\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n]$
e.g. [0.224,0.322,......0.112]
$\hat{X}_{adv} = [x_1, \cdots, \hat{x}_t, \cdots, x_n]$
subject to $\hat{x}_t = x_t + \delta * \hat{p}_t, t \in \hat{Loc}$
New Adversarial Example

Table 3
Datasets for time series forecasting

Dataset	Features	Length	Description
Electricity	3	2192	German Electricity Consumption [2]
CNYExch	5	2865	USD/CNY Exchange Rate [3]
NZTemp	9	1928	New Zealand Temperatures [4]
Oil	6	2692	iPath Pure Beta Crude Oil ETN [3]

During the experimental phases, we preprocessed the datasets for better model training. All datasets are normalized to the range of [0,1]. For attack evaluation, we prepared 250 samples randomly selected from the test set of each dataset. We split these samples into two parts: x samples collected as the adversarial example datasets D_{adv} for training the f_{learn}, and the remaining (250–x) samples used to evaluate each attack performance. x may vary in different experiment phases.

b *Parameter Setting*

During the training phase of the LBA, we systematically determined the appropriate hyperparameters through a series of experiments. This direct training approach allowed us to effectively leverage the performance of the LBA. We set learning rate as 0.005, batch size as 8, and the number of epochs as 50. For the balance between performance and computational cost of nVITA, we adjusted the max iteration of DE algorithm as 60, the population size as 15. The n of the nVITA is set as 1, which means the 1VITA will generate 1 perturbation for each sample. Similarly, Learning model also select 1 sensitive point for each sample. LBA learns from 1VITA in different factors, to be precise, factor β is set as {0.05,0.1,0.15,0.2}. For the factor δ of LBA, we set {1.00} as default. Different δ will be noted in the corresponding experiment phase.

c *Evaluation Metrics*

Because LBA depends on learning target adversarial attack, the fitness of the LBA is crucial. In this study, we will not only evaluate attack performance of the LBA, but also evaluate the fitting ability of the LBA.

To be specific, we utilize the root relative square error(RSE) to evaluate the attack performance. At the same time, root mean square error(RMSE) will be adopted to evaluate the fitting ability of the LBA. The *RSE* between the original predictions and the perturbed predictions will be used to evaluate the attack performance. A larger *RSE* indicates that the model has a lower accuracy. *RMSE* refers the error between the perturbations generated by the 1VITA and the perturbations generated by LBA. This metric is used to evaluate whether the LBA could learn the adversarial examples effectively. A lower *RMSE* indicates that the LBA has a better fit to the 1VITA in terms of the perturbation values.

For the RSE:

$$RSE = \frac{\sqrt{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}}{\sqrt{\sum_{i=1}^{n}(Y_i - mean(Y))^2}} \quad (8)$$

For the RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i^{1VITA} - \hat{Y}_i^{LBA})^2} \quad (9)$$

where $Y \epsilon \Omega_{test}$, Y_i refers to the true value of the i-th sample, $\hat{Y}_i$ refers to the prediction. $\hat{Y}_i^{1VITA}$ refers to the perturbed prediction only generated by the 1VITA, $\hat{Y}_i^{LBA}$ refers to perturbed prediction only generated by the LBA, n refers to the number of test length.

Accuracy Rate(AR) is used to evaluate whether LBA have a right decision on the sensitive points. Higher AR indicates that the LBA has a better fit to the 1VITA in terms of the sensitive points prediction. For the AR:

$$AR = \frac{1}{n}\sum_{i=1}^{n}(Loc_i^{1VITA} == \hat{Loc}_i^{LBA}), AR \in [0,1] \quad (10)$$

where Loc_i^{1VITA} refers to the sensitive point selected by the 1VITA, $\hat{Loc}_i^{LBA}$ refers to the sensitive point selected by the LBA.

[2] Source: https://open-power-system-data.org/
[3] Source:https://finance.yahoo.com/
[4] Source:https://climatedataguide.ucar.edu/climate-data/

Additionally, we use Mean Latency(ML) to evaluate the computational cost of several adversarial attack methods. ML can be formulated as follows:

$$ML = \frac{1}{n}\sum_{i=1}^{n} T_i^{cost} \tag{11}$$

where T_i^{cost} refers to the time cost of generating the i-th adversarial example.

d *Results*

Effectiveness: *RSE* is a crucial metric for evaluating the effectiveness of the adversarial attack.

In our experiments, we utilized δ values of 1.00 and 1.75 to craft adversarial examples for the LBA. Notably, when δ is set to 1.75, the perturbations introduced by our learning model, flearn, are amplified to 1.75 times their original magnitude. For LBA attack, we trained the learning model with the corresponding adversarial examples generated by 1VITA on each dataset and model. The adversarial example datasets contain 100 samples, and the remaining 150 samples are used to evaluate each attack performance. Table 4 and 5 separately presents the attack results under low perturbation and relatively high perturbation. We can observe that when attack happens, the *RSE* of predictions will increase. As FGSM and BIM is a global attack method, they have a higher *RSE* than LBA and 1VITA. Additionally, FGSM and BIM achieve nearly identical effects as their *RSE* values are nearly the same. LBA has a similar performance with 1VITA, as its *RSE* is close to the 1VITA. But we can also observe that in Oil dataset, LBA and 1VITA attacks make a small impact.

Time Cost: The computational expense of adversarial at- tacks is inherently linked to the capabilities of the hardware utilized. Given the variability in computational costs across different hardware devices, assessing mean latency offers a standardized measure of efficiency. Fig. 6 illustrates the mean latency across various datasets for the GRU model, with CNN and LSTM models exhibiting similar latencies.

It's noteworthy that FGSM, being a one-step attack, has a negligible mean latency (approximately 0.004s), which we also confirmed through our experiments. In contrast, the 1VITA method demonstrates significantly higher mean latencies, with the Electricity dataset averaging at 3.5s and the Oil dataset reaching up to 16s. This disparity underscores the increased time 1VITA requires to find optimal solutions as dataset complexity grows, particularly due to its reliance on the DE process. Interestingly, the mean latency of 1VITA varies with different values of β, showing an increase in the CNYExch dataset but a decrease in the

Table 4

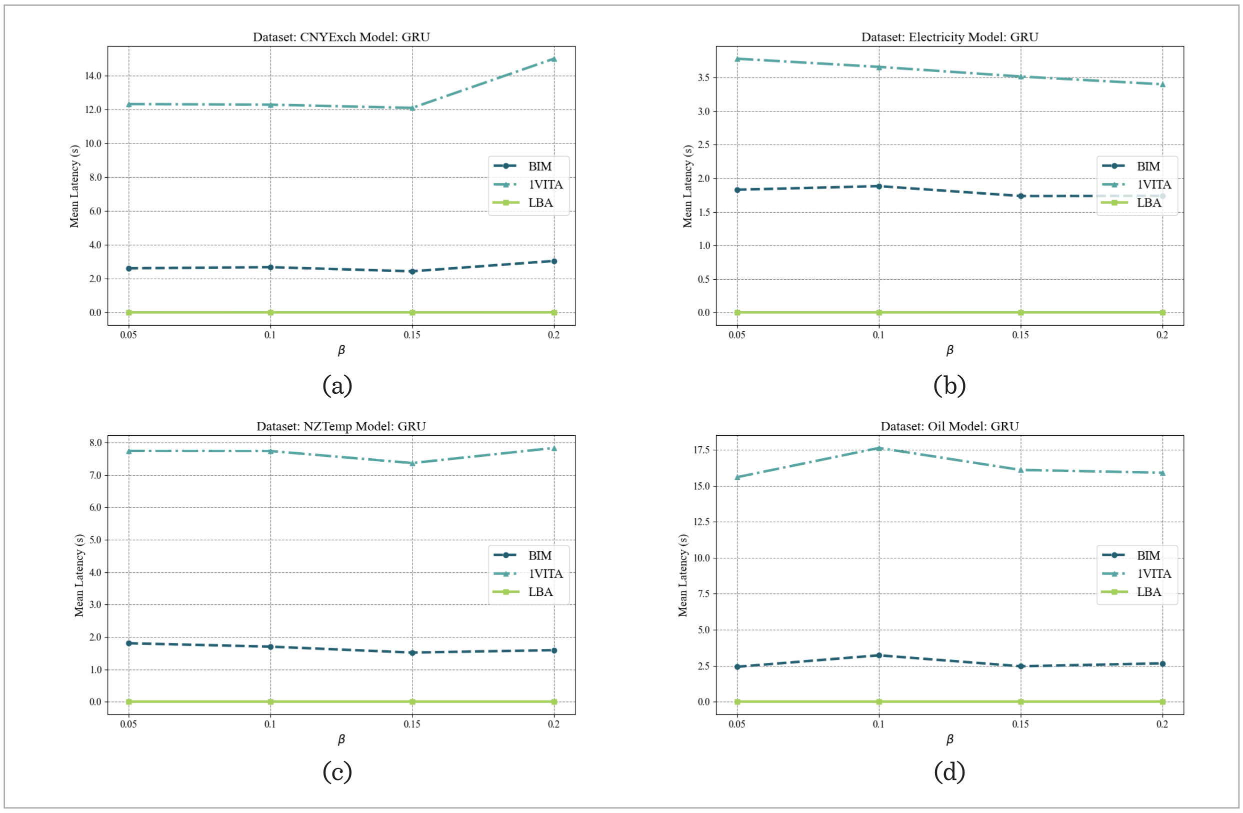
RSE when β = 0.05, count of D_{adv} = 100. A Larger *RSE* indicates that the attack is more effective.

Dataset	Model	*RSE*					
		Normal	LBA δ = 1.00	LBA δ = 1.75	1VITA	FGSM	BIM
CNYExch	CNN	0.081	**0.095**	**0.103**	0.082	0.092	0.092
	GRU	0.054	**0.057**	**0.060**	0.053	0.061	0.061
	LSTM	0.061	**0.064**	**0.068**	0.061	0.067	0.067
Electricity	CNN	0.591	**0.605**	**0.629**	0.648	0.700	0.700
	GRU	0.541	**0.549**	**0.573**	0.603	0.698	0.710
	LSTM	0.557	**0.566**	**0.574**	0.594	0.629	0.635
NZTemp	CNN	0.336	**0.333**	**0.334**	0.368	0.504	0.504
	GRU	0.318	**0.356**	**0.430**	0.338	0.476	0.476
	LSTM	0.322	**0.386**	**0.464**	0.354	0.540	0.540
Oil	CNN	0.064	**0.064**	**0.064**	0.064	0.074	0.074
	GRU	0.047	**0.048**	**0.047**	0.047	0.055	0.055
	LSTM	0.051	**0.050**	**0.050**	0.052	0.061	0.061

Table 5
RSE when $\beta = 0.2$, count of $D_{adv} = 100$

Dataset	Model	*RSE*					
		Normal	LBA $\delta = 1.00$	LBA $\delta = 1.75$	1VITA	FGSM	BIM
CNYExch	CNN	0.081	**0.083**	**0.093**	0.085	0.132	0.132
	GRU	0.054	**0.055**	**0.054**	0.056	0.081	0.080
	LSTM	0.061	**0.064**	**0.068**	0.062	0.087	0.087
Electricity	CNN	0.591	**0.705**	**0.848**	0.917	1.298	1.301
	GRU	0.541	**0.620**	**0.769**	0.883	1.134	1.396
	LSTM	0.557	**0.659**	**0.861**	0.790	0.986	1.066
NZTemp	CNN	0.335	**0.382**	**0.446**	0.488	1.071	1.071
	GRU	0.318	**0.337**	**0.362**	0.419	1.015	1.019
	LSTM	0.322	**0.390**	**0.492**	0.469	1.254	1.260
Oil	CNN	0.064	**0.064**	**0.065**	0.066	0.096	0.095
	GRU	0.047	**0.048**	**0.048**	0.047	0.068	0.068
	LSTM	0.051	**0.051**	**0.051**	0.052	0.074	0.074

Figure 6
Mean latency of adversarial attack in different datasets

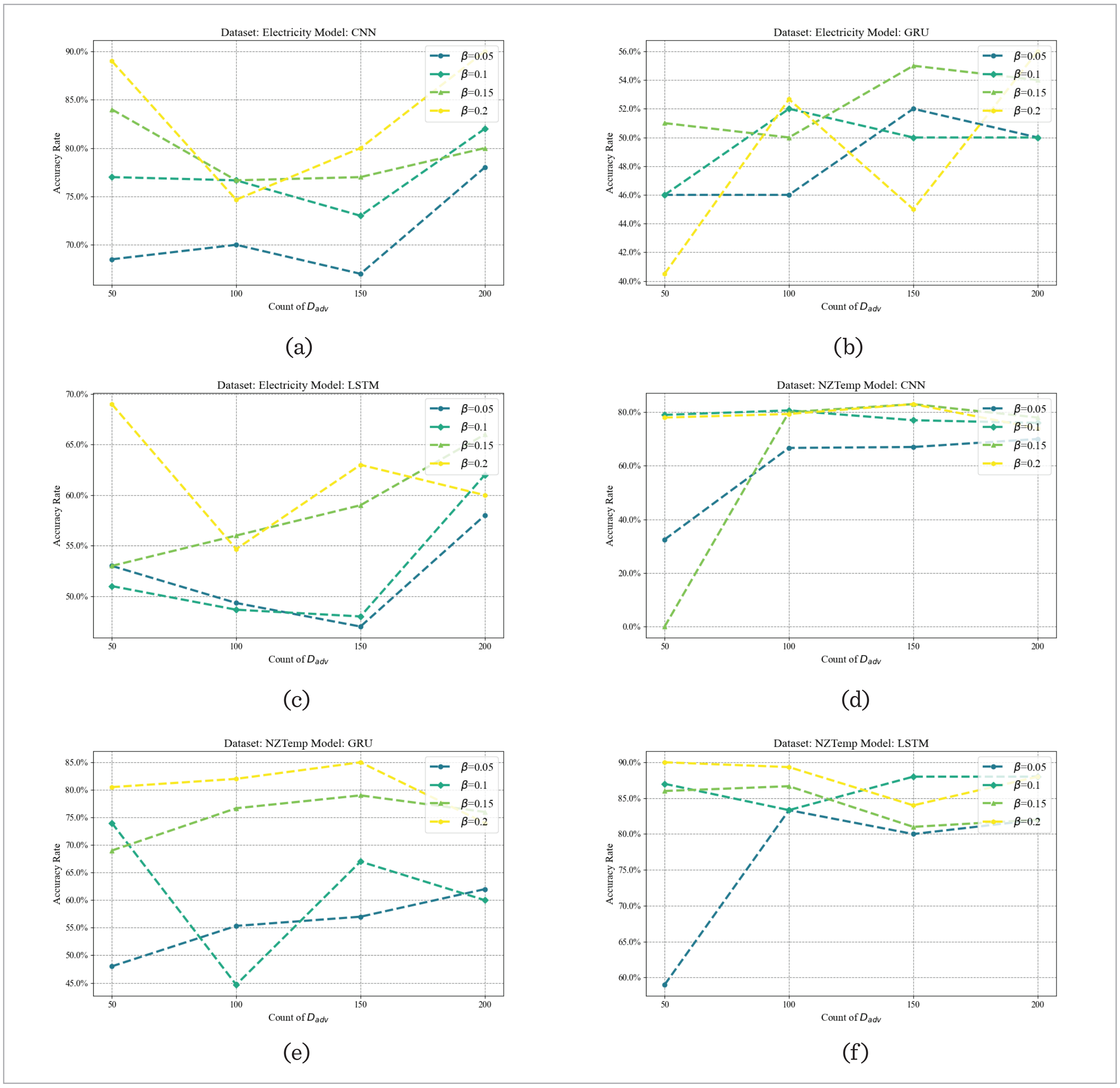


Electricity dataset as β increases. The BIM method, with a fixed perturbation iteration of 200, maintains a relatively lower and stable mean latency of around 2.0s across datasets compared to 1VITA. Our LBA approach, which queries the trained f_{learn} model only once to generate each adversarial example, achieves a lower mean latency than both 1VITA and BIM. Similar to FGSM, LBA is essentially a one-step attack, with the trained f_{learn} capable of generating adversarial examples within 1.0s. The relatively small size of our f_{learn} model also contributes to its rapid training phase, enhancing overall efficiency.

Fitting Ability: The essence of our Learning-Based At- tack (LBA) hinges on the accuracy of our learning model, f_{learn}. This model is tasked with predicting both the sensitive points and the perturbation values to craft effective adversarial examples. The precision of these dual outputs is crucial for the success of the

Figure 7

AR(Accuracy Rate) of sensitive points prediction in Electricity and NZTemp datasets



LBA. To determine the adequate size of the adversarial dataset, D_{adv}, for training f_{learn}, we analyzed the model's performance as the dataset size varied. Figure 8 presents RMSE between our LBA and the 1VITA method across different D_{adv} counts on a CNN model. Additionally, the factor β plays a role in the RMSE, with lower values of β generally leading to a better fit, as indicated by a lower RMSE. In Oil dataset, curves present a consistent trend, but the attack on this dataset gets small impact on the target model (Table 4 and 5). Despite the variability in the RMSE curves, the values remain relatively low in most cases, indicating that LBA maintains a robust fitting capability.

The predictions of sensitive points is also important to the LBA. Fig. 7 shows the accuracy of sensitive points predictions in Electricity and NZTemp datasets. AR curves are fluctuating in some cases. As count of D_{adv} reaches 200, the accuracy of sensitive points predictions usually gets a higher value. But in CNYExch and Oil datasets, we find our LBA has a poor fitting ability($AR \leq 20\%$) on sensitive points predictions. The sensitive points predictions is more difficult than perturbations prediction.

Transferability: A transferable attack is an attack generated for one model that can also be used to fool another model. As the adversarial examples generated by LBA don't require any query to the target model, they will remain same on other models. Hence, the transferability become a task: Whether the sensitive points and perturbations learned from 1VITA on one model can also be feasible to attack other models, revealing a inner sensitive relationship on datasets itself.

Fig. 9 shows the RSE of LBA in different datasets and models. The results are displayed through heat maps. As the color change on the heat maps, we can have a clear view of the transferability of LBA in different datasets and models.

A deeper color indicates a higher *RSE*, which means a better attack performance. Through a glance, CNN models are more vulnerable than GRU and LSTM by observing the 1st column of heat map. The transferability on GRU and LSTM have a similar result. The numerical value of the 2x2 area located in the lower right corner of the heat map, where indicates transferability of GRU and LSTM models, shows a similar color ((a),(b),(d)). The structure of GRU and LSTM are both designed to capture long-term dependencies, which may lead to a similar transferability. In con-

Figure 8

RMSE between LBA and 1VITA in different datasets as the count of D_{adv} increases

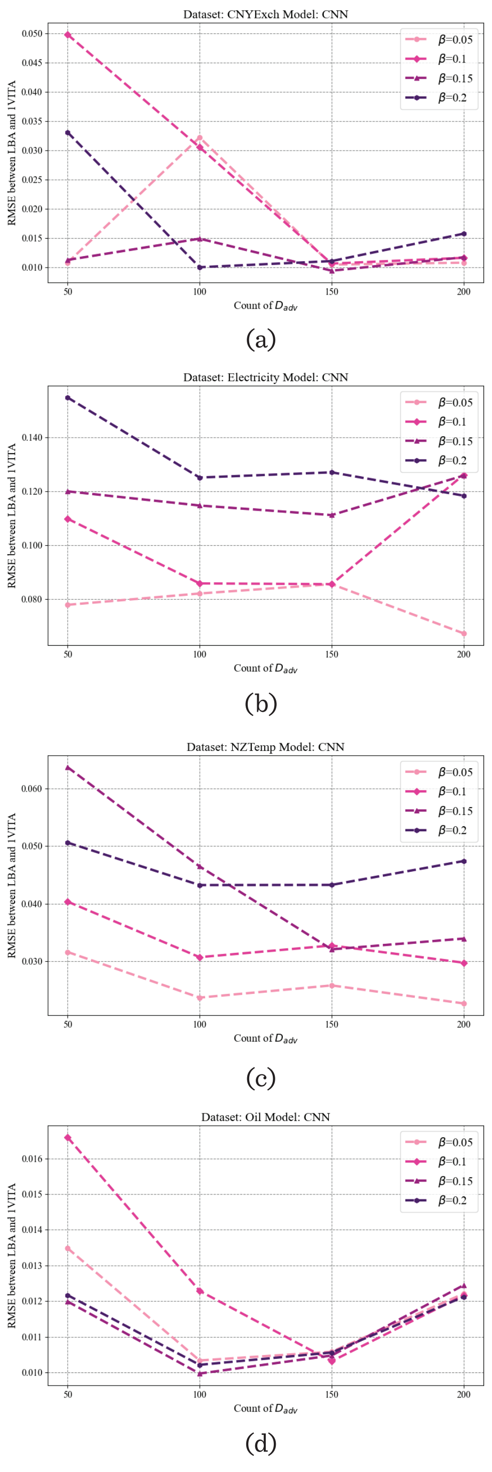
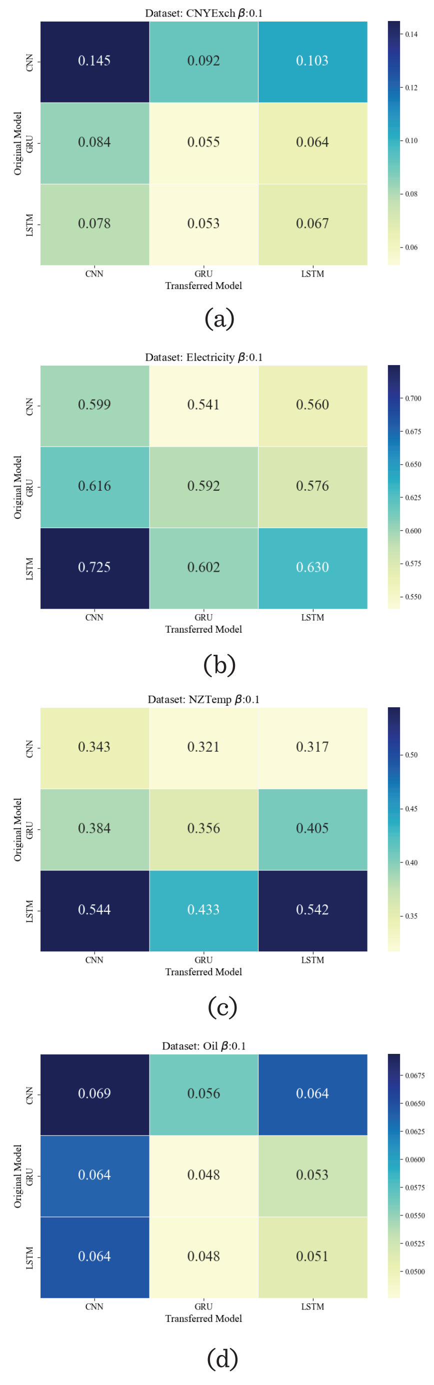


Figure 9

Transferability of LBA in different datasets and models, count of $D_{adv} = 100, \beta = 0.1$. Models on the left side indicate original models, while models on the bottom side indicate transferred target models. Results on diagonal line indicate the attack performance of LBA on original models.



trast, when CNN is employed as the original model and transferred to LSTM and GRU models, it exhibits lower performance. By observing the 1st row of heat map, we can clearly see that 1st element is deeper than 2nd and 3rd element.

5. Conclusion and Future Work

The key of sparse attack is to find the sensitive points to perturb. This finding process is time-consuming and computationally expensive. Many black box attack methods require continuous queries to the target model to find the sensitive points.

Image data, with its multitude of pixel points, presents a complex landscape that makes it challenging for a model to learn the sensitive points within. However, in TSF tasks, temporal data are segmented into smaller, manageable windows. This reduction in the scale of individual samples not only streamlines the data but also renders it feasible for a model to learn the sensitive points of the samples. By focusing on these compact segments, model can more effectively identify and understand the sensitive points within the data.

Our experiments demonstrate that window-split time series adversarial examples can be learn by a learning model. The adversarial examples generated by LBA can make a compa- rable performance with the target black box attack methods while continuous queries to the target model are not required. But through the experiments, we find that the LBA has a poor ability on some datasets. We also find that the fitting ability of the LBA is not stable and hard to control in some cases. Plus, multiple sensitive points prediction is still a challenge for the LBA. Hence, the learning process and structure of f_{learn} of LBA needs to be improved to make it more accurate and stable.

While adversarial attacks can fool models, in TSF tasks, modifications to samples that affect prediction are inevitable due to the close relationship between prediction and samples. The delineation of adversarial attacks on TSF tasks is not well- defined, leading to a challenge in discerning whether data have been deliberately compromised by an attack or are merely experiencing the effects of random noise. Thus, our future work will also focus on developing more sophisticated methods to differentiate between adversarial attacks and random noise in data.

References

1. J. Cao, Z. Li, and J. Li, ″Financial time series forecasting model based on ceemdan and lstm,″ Physica A: Statistical Mechanics and its Applications, vol. 519, pp. 127-139, 2019. https://doi.org/10.1016/j.physa.2018.11.061
2. V. K. R. Chimmula and L. Zhang, ″Time series forecasting of covid-19 transmission in canada using lstm networks,″ Chaos, solitons & fractals, vol. 135, p. 109864, 2020. https://doi.org/10.1016/j.chaos.2020.109864
3. W. Li, H. Wu, N. Zhu, Y. Jiang, J. Tan, and Y. Guo, ″Prediction of dissolved oxygen in a fishery pond based on gated recurrent unit (gru),″ Information Processing in Agriculture, vol. 8, no. 1, pp. 185-193, 2021. https://doi.org/10.1016/j.inpa.2020.02.002
4. Z. Cui, K. Henrickson, R. Ke, and Y. Wang, ″Traffic graph convolutional recurrent neural network: A deep learning framework for network- scale traffic learning and forecasting,″ IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 11, pp. 4883-4894, 2020. https://doi.org/10.1109/TITS.2019.2950416
5. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfel- low, and R. Fergus, ″Intriguing properties of neural networks,″ arXiv: Computer Vision and Pattern Recognition,arXiv: Computer Vision and Pattern Recognition, 2013.
6. I. Goodfellow, J. Shlens, and C. Szegedy, ″Explaining and harnessing adversarial examples,″ Cornell University - arXiv,Cornell University - arXiv, Dec 2014.
7. A. Kurakin, I. J. Goodfellow, and S. Bengio, Adversarial examples in the physical world. Chapman and Hall/CRC, 2018. https://doi.org/10.1201/9781351251389-8
8. S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, ″Deepfool: a simple and accurate method to fool deep neural networks,″ in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2574-2582. https://doi.org/10.1109/CVPR.2016.282
9. A. Xu, X. Wang, Y. Zhang, T. Wu, and X. Xian, ″Adversarial attacks on deep neural networks for time series prediction,″ in 2021 10th Interna- tional Conference on Internet Computing for Science and Engineering, 2021, pp. 8-14. https://doi.org/10.1145/3485314.3485316
10. G. R. Mode and K. A. Hoque, ″Adversarial examples in deep learning for multivariate time series regression,″ in 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), 2020, pp. 1-10. https://doi.org/10.1109/AIPR50011.2020.9425190
11. T. Wu, X. Wang, S. Qiao, X. Xian, Y. Liu, and L. Zhang, ″Small perturbations are enough: Adversarial attacks on time series prediction,″ Inf. Sci., vol. 587, no. C, pp. 794-812, 2022. https://doi.org/10.1016/j.ins.2021.11.007
12. J. Su, D. V. Vargas, and K. Sakurai, ″One pixel attack for fooling deep neural networks,″ IEEE Transactions on Evolutionary Computation, pp. 828-841, Oct 2019. https://doi.org/10.1109/TEVC.2019.2890858
13. F. Croce and M. Hein, ″Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers,″ in International Conference on Machine Learning, 2021, pp. 2201-2211.
14. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, ″Towards deep learning models resistant to adversarial attacks.″ International Conference on Learning Representations,International Conference on Learning Representations, Feb 2018.
15. Z. Chen, K. Dost, X. Zhu, X. Chang, G. Dobbie, and J. Wicker, ″Targeted attacks on time series forecasting,″ in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2023, pp. 314-327. https://doi.org/10.1007/978-3-031-33383-5_25
16. W. Yang, J. Yuan, X. Wang, and P. Zhao, ″Tsadv: Black-box adversarial attack on time series with local perturbations,″ Engineering Applications of Artificial Intelligence, vol. 114, p. 105218, 2022. https://doi.org/10.1016/j.engappai.2022.105218
17. R. Storn and K. Price, ″Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces,″ Journal of Global Optimization, vol. 11, pp. 341-359, 1997. https://doi.org/10.1023/A:1008202821328
18. B. Lim and S. Zohren, ″Time-series forecasting with deep learning: a survey,″ Philosophical Transactions of the Royal Society A: Mathemati- cal, Physical and Engineering Sciences, vol. 379, no. 2194, p. 20200209, 2021. https://doi.org/10.1098/rsta.2020.0209
19. Z. Lipton, J. Berkowitz, and C. Elkan, ″A critical review of recurrent neural networks for sequence learning,″ arXiv: Learning,arXiv: Learn- ing, 2015.
20. K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmid- huber, ″Lstm: A search space odyssey,″ IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 10, pp. 2222-2232, 2017. https://doi.org/10.1109/TNNLS.2016.2582924
21. K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, ″Learning phrase

representations using rnn encoder-decoder for statistical machine translation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724-1734.

22. N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and Swami, "Practical black-box attacks against machine learning," in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 506-519. https://doi.org/10.1145/3052973.3053009
23. M. G. Abdu-Aguye, W. Gomaa, Y. Makihara, and Y. Yagi, "Detecting adversarial attacks in time-series data," in ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 3092-3096. https://doi.org/10.1109/ICASSP40776.2020.9053311
24. J. Teraoka and K. Tamura, Detecting Adversarial Examples for Time Series Classification and Its Performance Evaluation. Springer, 2021. https://doi.org/10.1007/978-981-16-2765-1_47
25. Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," Cornell University - arXiv,Cornell University - arXiv, 2015.