Adversarial Attacks on Time Series

Fazle Karim[1], *Graduate Student Member, IEEE* Somshubra Majumdar[2], and Houshang Darabi[1], *Senior Member, IEEE*

Abstract—Time series classification models have been garnering significant importance in the research community. However, not much research has been done on generating adversarial samples for these models. These adversarial samples can become a security concern. In this paper, we propose utilizing an adversarial transformation network (ATN) on a distilled model to attack various time series classification models. The proposed attack on the classification model utilizes a distilled model as a surrogate that mimics the behavior of the attacked classical time series classification models. Our proposed methodology is applied onto 1-Nearest Neighbor Dynamic Time Warping (1-NN DTW) and a Fully Convolutional Network (FCN), all of which are trained on 42 University of California Riverside (UCR) datasets. In this paper, we show both models were susceptible to attacks on all 42 datasets. When compared to Fast Gradient Sign Method, the proposed attack generates a larger faction of successful adversarial black-box attacks. A simple defense mechanism is successfully devised to reduce the fraction of successful adversarial samples. Finally, we recommend future researchers that develop time series classification models to incorporating adversarial data samples into their training data sets to improve resilience on adversarial samples.

Index Terms—Time series classification, Adversarial machine learning, Perturbation methods, Deep learning

◆

1 INTRODUCTION

Over the past decade, machine learning and deep learning have been powering several aspects of society [1]. Machine learning and deep learning are being used in some areas such as web searches [2], recommendation systems [3], and wearables [4]. With the advent of smart sensors, advancements in data collection and storage at vast scales, ease of data analytics and predictive modeling, time series data being collected from various sensors can be analyzed to determine regular patterns that are interpretable and exploitable. Classifying these time series data has been an area of interest by several researchers [5]–[8]. Time series classification models are being used in health care, where ECG data are used to detect patients with severe cognitive defects, in audio, where words are categorized into different phenomes, and in gesture recognition, where motion data is used to categorize actions being made. Sensor data for resource and safety-critical applications such as manufacturing plants, industrial engineering, and chemical compound synthesis, when augmented by on-device analytics would allow automated response to avert significant issues in normal operation [9]. A successful time series classification model is able to capture and generalize the pattern of time series signals such that it is able to classify unseen data. Similarly, computer vision classification models exploit the spatial structure intrinsic to images obtained in the real world. However, computer vision models have been shown to make incorrect predictions on the seemingly correct input, which is termed as an adversarial attack. Utilizing a variety of adversarial attacks, complex models are tricked to incorrectly predict the wrong class label. This is a serious security issue in neural networks widely used for vision-based tasks where adding slight perturbations or carefully crafted noise on an input image can mislead the image classification algorithm to make highly confident, yet wildly inaccurate predictions [10, 11]. This has been a growing concern in the Computer Vision field, where Deep Neural Networks (DNN) have been shown to be particularly susceptible to attacks [12, 13]. While DNNs are state-of-the-art models for a variety of classification tasks in several fields, including time series classification [14]–[16], these vulnerabilities harmfully impact real-world applicability in domains where secure and dependable predictions are of paramount importance [10, 17]. Compounding the severity of this issue, recent work by Papernot et al. has shown that adversarial attacks on a particular computer vision classifier can easily be transferred into other similar classifiers [18]. Only recently has the focus of attacks been shifted to Time Series classification models based on deep neural networks and classical models [12].

Several adversarial sample crafting techniques have been proposed to trick various image classification models that rely on DNN (state-of-the-art models for computer vision). Most of these techniques target the gradient information from the DNNs, which make them susceptible to these attacks [19]–[21]. Currently, new research is being done in generating natural language adversarial samples [22, 23]. This is extremely difficult due to the semantic-preserving perturbations. Even though time series classification models and natural language processing (NLP) models use similar deep learning modules (1D CNN, LSTM) the domains are completely different. Further, NLP models work better than time series classification models on discrete input data because they model different patterns and structures. While adversarial attacks on NLP models which accept discrete tokens as input has been well studied, there has been relatively little study on adversarial attacks on time series models, which accept real values time series sequence as input. Generating adversarial samples for time series classification model has not been studied as much, in spite of the potentially large security risk they may possess. One

---

[1] *Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago,IL*
[2] *Computer Science, University of Illinois at Chicago, Chicago, IL*

major security concern exists in voice recognition tasks that convert speech-to-text. Carlini and Wagner [24] show how speech-to-text classifiers can be attacked. In addition, they provide various audio clips where a speech-to-text classifier, DeepSpeech, is not able to correctly detect the speech.Other security concerns can exist in healthcare devices that use time series classification algorithms, where it can be tricked into misdiagnosing patients that can affect the diagnosis of their ailment. Time series classification algorithms used to detect and monitor seismic activity can be manipulated to create fear and hysteria in our society. Wearables that use time series data to classify activity of the wearer can be fooled into convincing the users they are doing other actions. Most of the current state-of-the-art time series classification algorithms are classical approaches, such as 1 Nearest Neighbor - Dynamic Time Warping (1-NN DTW) [25], Kernel SVMs [26], or sophisticated methods such as Weasel [27], COTE [28], and Fast-Shapelet [29]. However, DNNs are fast becoming excellent time series classifiers due to their simplicity and effectiveness. The traditional time series classification models are harder to attack as it can be considered a black-box model with a non-differentiable internal computation. As such, no gradient information can be exploited. However, DNN models are more susceptible to white-box attacks as their gradient information can easily be exploited. A white-box attack is where the adversary is "given access to all elements of the training procedure" [21] - which includes the training dataset, training algorithm, the parameters and weights of the model, and the model architecture itself [21]. In other words, during a white-box attack, the attacker has full knowledge about the time series classification model (parameters, hyper-parameters, architecture, etc). Conversely, a black-box attack only has access to the target model's training procedure and model architecture [21]. During a black-box attack, the attacker has almost no knowledge of the time series classification model. In some cases, the only knowledge known when performing black-box attacks is the length of the input time series data. In this paper, we propose a black-box and a white-box attack that can attack both classical and deep learning time series classification state-of-the-art models.

In this work, we propose a proxy attack strategy on a target classifier via a student model, trained using standard model distillation techniques to mimic the behavior of the target classical time series classification models. The "student" network is the neural network distilled from another time series classification model, called the "teacher" model, that learns to approximate the output of the teacher model. Once the student model has been trained, our proposed adversarial transformation network (ATN) can then be trained to attack this student model. We apply our methodologies onto 1-NN DTW, Fully Connected Network and Fully Convolutional Network (FCN) that are trained on 42 University of California Riverside (UCR) datasets [9]. When compared to a Baseline adversarial attack (Fast Gradient Sign Method), our proposed attack is able to generate a larger fraction of successful adversarial black-box attacks. Further, a simple defense mechanism is devised to reduce the fraction of successful adversarial samples on various time series classification attacks. Finally, we recommend future researchers that develop time series classification models to consider model robustness as an evaluative metric and incorporate adversarial data samples into their training data sets in order to further improve resilience to adversarial attacks.

The remainder of this paper is structured as follows: Section 2 provides a brief background on a couple time series classification models and background information on a few adversarial crafting techniques used on computer vision problems. Section 3 details our proposed methodologies and Section 4 presents and explains the results of our proposed methodologies on a couple of time series classification models. Section 5 concludes the paper and proposes future work.

2 BACKGROUND & RELATED WORKS

2.1 Time Series Classification Models

2.1.1 1-NN Dynamic Time Warping

The equations and definitions below are obtained from Kate et al. [30] and Xi et al. [8]. Dynamic Time Warping is a measure of similarity between 2 time series, Q and C, which is detected by finding their best alignment. Time series Q and C are defined as:

$$Q = q_1, q_2, q_3, ..., q_i, ..., q_n \tag{1}$$

$$C = c_1, c_2, c_3, ..., c_i, ..., c_m. \tag{2}$$

To align both the time series data, the distance between each timestep of Q and C is calculated, $(q_i - c_j)^2$, to generate a n-by-m matrix. In other words, the i^{th} and j^{th} of the matrix is the q_i and c_j. The optimal alignment between Q and C is considered the warping path, W, such that $W = w_1, w_2, w_3, ..., w_k, ..., w_K$. The warping path is computed such that,

1) $w_1 = (1, 1)$,
2) $w_k = (n, m)_k$,
3) given $w_k = (a, b)$ then $w_{k-1} = (a', b')$ where $0 \leq a - a' \leq 1$ and $0 \leq b - b' \leq 1$.

The optimal alignment is the warping path that minimizes the total distance between the aligning points,

$$DTW(Q, C) = \underset{W=w_1,w_2,...,w_K}{\operatorname{argmin}} \sqrt{\sum_{k=1, w_k=(i,j)}^{k} (q_i - c_j)^2}. \tag{3}$$

2.1.2 Fully Convolutional Network

The Fully Convolutional Network (FCN) is one of the earliest deep learning time series classifier [31]. It contains 3 convolutional layers, with convolution kernels of size 8, 5 and 3 respectively, and emitting 128, 256 and 128 filters respectively. Each convolution layer is followed by a batch normalization [32] layer that is applied with a ReLU activation layer. A global average pooling layer is employed after the last ReLU activation layer. Finally, softmax is applied to determine the class probability vector.

2.2 Adversarial Transformation Network

Several methods have been proposed to generate adversarial samples that attack deep neural networks that are trained for computer vision tasks. Most of these methods use the gradient with respect to the image pixels of these neural networks. Baluja and Fischer [33] propose Adversarial Transformation Networks (ATNs) to efficiently generate an adversarial sample that attacks various networks by training a feed-forward neural network in a self-supervised method. Given the original input sample, ATNs modify the classifier outputs slightly to match the adversarial target. ATN works similarly to the generator model in the Generative Adversarial Training framework.

According to Baluja and Fischer et al. [33], an ATN can be parametrized as a neural network $g_f(x) : x \rightarrow \hat{x}$, where f is the target model (either a classical model or another neural network) which outputs either a probability distribution across class labels or a sparse class label, and $\hat{x} \sim x$, but argmax $f(x) \neq$ argmax $f(\hat{x})$. To find g_f, we minimize the following loss function :

$$L = \beta * L_x(g_f(\mathbf{x}_i), \mathbf{x}_i) + L_y(f(g_f(\mathbf{x}_i)), f(\mathbf{x}_i)) \tag{4}$$

where L_x is a loss function on the input space (e.g. L_2 loss function), L_y is the specially constructed loss function on the output space of f to avoid learning the identity function, $\mathbf{x}_i$ is the i-th sample in the dataset and β is the weighing term between the two loss functions.

It is necessary to carefully select the loss function L_y on the output space to successfully avoid learning the identity function. Baluja and Fischer et al. [33] define the loss function L_y as $L_y(\mathbf{y}', \mathbf{y}) = L_2(\mathbf{y}', r(\mathbf{y}, t))$, where $\mathbf{y} = f(x)$, $\mathbf{y}' = f(g_f(x))$, t is the index of the target class such that $t \in [1 \ldots C]$, where C is the number of classes and $r(\cdot)$ is a reranking function that modifies $\mathbf{y}$ such that $y_k < y_t, \forall k \neq t$. This reranking function $r(\mathbf{y}, t)$ can either be the simple one hot encoding function $onehot(t)$ or be formulated to take advantage of the already present $\mathbf{y}$ to encourage better reconstruction. We therefore utilize the reranking function proposed by Baluja and Fischer et al. [33], which can be formulated as:

$$r_\alpha(\mathbf{y}, t) = norm\left(\left\{\begin{matrix} \alpha * max\ y, & \text{if } k = t \\ y_k, & \text{otherwise} \end{matrix}\right\}_{k \in y}\right) \tag{5}$$

where $\alpha > 1$ is an additional hyperparameter which defines how much larger y_t should be than the current max classification and $norm$ is a normalizing function that rescales its input to be a valid probability distribution

2.3 Transferability Property

Papernot et al. [18] propose a black-box attack by training a local substitute network, s, to replicate or approximate the target DNN model, f. The local substitute model is trained using synthetically generated samples and the output of these samples are labels from f. Subsequently, s is used to generate adversarial samples that it misclassifies. Generating adversarial samples for s is much easier, as its full knowledge/parameters are available, making it susceptible to various attacks. The key criteria to successfully generate adversarial samples of f is the transferability property, where adversarial samples that misclassify s will also misclassify f.

2.4 Knowledge Distilation

Knowledge distillation, first proposed by Bucila et al. [34], is a model compression technique where a small model, s, is trained to mimic a pre-trained model, f. This process is also known as the model distillation training where the teacher is f and the student is s. The knowledge that is distilled from the teacher model to the student model is done by minimizing a loss function, where the objective of the student model is to imitate the distribution of the class probabilities of the teacher model. Hinton et al. [35] note that there are several instances where the probability distribution is skewed such that the correct class probability would have a probability close to 1 and the remaining classes would have a probability closer to 0. Hence, Hinton et al. [35] recommend computing the probabilities q_i from the pre-normalized logits z_i, such that:

$$q_i = \sigma(z; T) = \frac{exp\ (z_i/T)}{\sum_j exp\ (z_j/T)} \tag{6}$$

where T is a temperature factor normally set to 1. Higher values of T produce softer probability distributions over classes. The loss that is minimized is the model distillation loss, further explained in Section 3.3.

3 METHODOLOGY

3.1 Gradient Adversarial Transformation Network

In this work, we employ distinct methodologies for white-box and black-box attacks, in order to adhere to a strictly realistic set of limitations in black-box attacks. For both methodologies, we incorporate Adversarial Transformation Networks (ATN) [33] as a generative neural network that accepts an input time series sample x and transforms it to an adversarial sample $\hat{x}$.

An Adversarial Transformation Network can be formulated as a neural network $g_f(x) : x \rightarrow \hat{x}$, where f is the model that will be attacked. We further augment the information available to the ATN with the gradient of the input sample x with respect to the softmax scaled logits of the target class predicted by the attacked classifier. We can therefore formally define a Gradient Adversarial Transformation Network (GATN) as a neural network parametrized as $g_f(x, \tilde{x}) : [x; \tilde{x}] \rightarrow \hat{x}$, where:

$$\tilde{x} = \frac{\partial f_t}{\partial x} \tag{7}$$

such that $x \in \mathbb{R}^T$ is an input time series of maximum length T, f_t represents the probability of the input time series being classified as the target class, t, and $[\,;\,]$ represents the concatenation operation of two vectors. GATN computes the gradient of the input with respect to the output as the objective of an adversarial network is to learn the perturbations necessary to alter the input in order to affect the classification outcome. Hence, with the availability of the input gradient $\tilde{x}$, the Gradient Adversarial Transformation Network can better construct adversarial samples that can affect the targeted model while reducing the overall perturbation added to the sample. Therefore we utilize the GATN model for all of our attacks.

A significant issue with the above formulation of the GATN is the non-differentiability of classical models. Distance-based models such as 1-NN Dynamic Time Warping do not have the notion of gradients during either training or evaluation. Therefore, we cannot directly compute the gradient of the input ($\tilde{x}$) with respect to the 1-NN DTW model f. We discuss the solution to this issue in Section 3.3, by building a student neural network s which approximates the predictions of the non-differentiable classical classifier f.

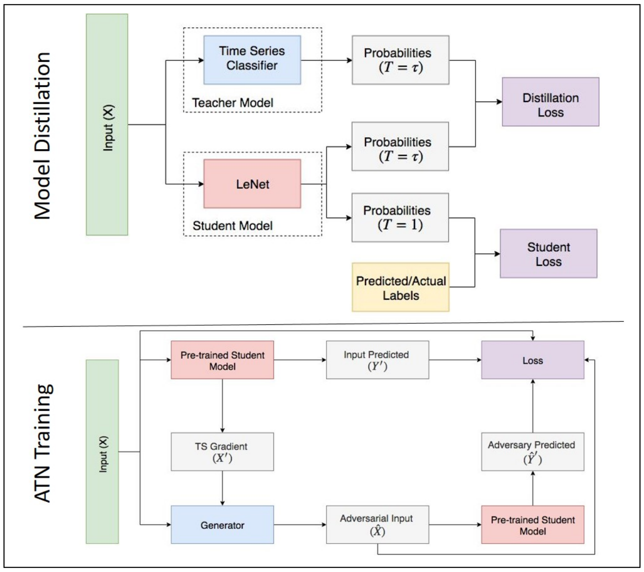


Fig. 1: The top diagram shows the methodology of training the model distillation used in the white-box and black-box attacks. The bottom diagram is the methodology utilized to attack a time series classifier.

3.2 Black-box & White-box Restrictions

While this formulation of the GATN is sufficient for white-box attacks where we have access to the attacked model f or the student model s, this assumption is unrealistic in the case of black-box attacks. For a black-box, we are not permitted access to either the internal model (a neural network or a classical model) or to the dataset that the model was trained on. Furthermore, for black-box attacks, we impose a restriction on the predicted labels, such that we utilize only the class label predicted, and not the probability distribution produced after softmax scaling (for neural networks), or the scaled probabilistic approximations of classical model predictions.

To further restrict ourselves to realistic attack vectors, we stratify the available dataset D, which will be used to train the GATN, into two halves, such that we train the GATN on one subset, D_{eval}, and are able to perform evaluations on both this train set and the wholly unseen test set, D_{test}. Note that this available dataset D is not the dataset on which the attacked model f was trained on. As such, we never utilize the train set available to the attacked classifier to either train or evaluate the GATN model. In order to satisfy these constraints on available data, we define our available dataset D as the test set of the UCR Archive [9]. As the test set was never used to train any attacked model f, it is sufficient to utilize it as an unseen dataset. We then split the test dataset into two class-balanced halves, D_{eval} and D_{test}. Another convenience is the availability of test set labels, which can be harnessed as a strict check when evaluating adversarial generators.

When we evaluate under the constraints of black-boxes, we further limit ourselves to "unlabeled" train sets, where we assume the available dataset is unlabeled, and thereby utilize only the predicted label from the attacked classifier f to label the dataset prior to attacks. We state this as an important restriction, considering that it is far more difficult to freely obtain or create datasets for time series than for images which are easily understood and interpreted. For time series, significant expertise may be required to distinguish one sample amongst multiple classes, whereas natural images can be coarsely labeled with relative ease without sophisticated equipment or expertise.

3.3 Training Methodology

A chief consideration during training of ATN or GATN is the loss formulation on the prediction space (L_y) is heavily influenced by the reranking function $r(\cdot)$ chosen. If we opt for the one hot encoding of the target class, we lose the ability to maintain class ordering and the ability to adjust the ranking weight (α) to obtain adversaries with less distortion. However, to utilize the appropriate reranking function, we must have access to the class probability distribution, which is unavailable to black-box attacks. It may not even be possible to compute for certain classical models such as 1-NN DTW which uses distance-based computations to determine the nearest neighbor.

To overcome this limitation, we employ knowledge distillation as a mechanism to train a student neural network s, which is trained to replicate the predictions of the attacked model f. As such, we are required to compute the predictions of the attacked model on the dataset we possess just one time, which can be either class labels or probability distribution over all classes. We then utilize these labels as the ground truth labels that the student s is trained to imitate. In case the predictions are class labels, we utilize one hot encoding scheme to compute the cross entropy loss, otherwise, we try to imitate the probability distribution directly. It is to be noted that the student model shares the training dataset D_{eval} with the GATN model.

As suggested by Hinton et al. [35], we describe the training scheme of the student as shown in Figure 1. We scale the logits of the student s and teacher f (iff the teacher provides probabilities and it is a white-box attack) by a temperature scaling parameter τ, which is kept constant at 10 for all experiments. When training the student model, we minimize the loss function defined as:

$$L_{transfer} = \gamma * L_{distillation} + (1 - \gamma) * L_{student} \tag{8}$$

$$L_{distillation} = \mathcal{H}(\sigma(z_f; T = \tau), \sigma(z_s; T = \tau)) \tag{9}$$

$$L_{student} = \mathcal{H}(y, \sigma(z_s; T = 1)) \tag{10}$$

where $\mathcal{H}$ is the standard cross entropy loss function, z_s and z_f are the un-normalized logits of the student (s) and teacher (f) models respectively, $\sigma(\cdot)$ is the scaled-softmax operation described in Equation (6), y is the ground truth labels, and γ is a gating parameter between the two losses and is used to maintain a balance between how much the student s imitates the teacher f versus how much it learns from the hard label loss. When training a student as a white-box attack, we set γ to be 0.5, allowing the equal weight to both losses, whereas for a black-box attack, we set γ to be 1. Therefore for black-box attacks, we force the student s to only mimic the teacher f to the limit of its capacity. In setting this restriction, we limit the amount of information that may be made available to the GATN.

Once we have a student model s which is capable of simulating the predictions of the attacked model f, we then train the GATN using this student model. Figure 1 shows the methodology of training such a model. Since the GATN requires not just the original sample x but also the gradient of that sample $\tilde{x}$ with respect to the predictions for the targetted class, we require two forward passes from the student model. The first forward pass is simply to obtain the gradient of the input $\tilde{x}$, as well as the predicted probability distribution of the student y'. The adversarial sample crafted ($\hat{X}$) is then used in a second forward pass to compute the predicted probability distribution of the student with respect to the adversarial sample, $\hat{y}'$. We minimize the weighted loss measure L defined in Section 2.2 in order to train the GATN model.

3.4 Evaluation Methodology

Due to the different restrictions imposed between available information depending on whether the attack is a white-box or black-box attack, we train the GATN on one of two models. We assert that we train the GATN by attacking the target neural network f directly only when we perform a white-box attack on a neural network. In all other cases, whether the attack is a white-box or black-box attack, and whether the attacked model is a neural network or a classical model, we select the student model s as the model which is attacked to train the GATN, and then use the GATN's predictions ($\hat{x}$) to check if the teacher model f is also attacked when provided the predicted adversarial input ($\hat{x}$) as a sample.

During evaluation of the trained GATN, we compute the number of adversaries of the attacked model f that have been obtained on the training set D_{eval}. During the evaluation, we can measure any metric under two circumstances. Provided a labeled dataset which was split, we can perform a two-fold verification of whether an adversary was found or not. First, we check that the ground truth label matches the predicted label of the classifier when provided with an unmodified input ($y = y'$ when input x is provided to f), and then check whether this predicted label is different from the predicted label when provided with the adversarial input ($y' \neq \hat{y}'$ when input $\hat{x}$ is provided to f). This ensures that we do not count an incorrect prediction from a random classifier as an attack.

Another circumstance is that we do not have any labeled samples prior to splitting the dataset. This training set is an unseen set for the attacked model f, therefore we consider that the dataset is "unlabeled", and assume that the label predicted by the base classifier is the ground truth ($y = y'$ by default, when sample x is provided to f). This is done prior to any attack by the GATN and is computed just once. We then define an adversarial sample as a sample $\hat{x}$ whose predicted class label is different than the predicted ground truth label ($y' \neq \hat{y}'$, when sample $\hat{x}$ is provided to f). A drawback of this approach is that it is overly optimistic and rewards sensitive classifiers that misclassify due to very minor alterations.

In order to adhere to an unbiased evaluation, we chose the first option, and utilize the provided labels that we know from the test set to properly evaluate the adversarial inputs. In doing so, we acknowledge the necessity of a labeled test set, but as shown above, it is not strictly necessary to follow this approach.

4 EXPERIMENTS AND RESULTS

All methodologies were tested on 42 benchmark datasets for time series classification found in the UCR repository. The 42 datasets selected were all from the types "Sensor", "ECG", "EOG", and "Hemodynamics", where an adversarial attack is a potential security concern. The 42 datasets contain data with application varying from classifying chlorine concentration to earthquakes. The ECG datasets contain a few time series classification problems that require classification of humans with heart conditions or Myocardial Infarctions. A couple of the EOG and Hemodynamics datasets require the classification of Japanese Katakana strokes and heart air pressure of pigs respectively. Further detailed information on these datasets can be found online [9]. These 42 datasets are all the datasets in all the domains in the repository that possess a security concern. With the exception of images and motion, we believe the remaining domains do not possess a serious security concern realistically. Adversarial attacks in the domain of images and motion is well studied and is the reason why the proposed time series classification adversarial methodologies are not used upon it.

We evaluate based on two criterion, the mean squared error between the training dataset and the generated samples (lower is better) and the fraction of successful adversaries (higher is better). For all experiments, we keep α, the reranking weight, set to 1.5, the target class set to 1, and perform a grid search over 5 possible values of β, the reconstruction weight term, such that $\beta = 10^{-b};\ \ b \in \{1, 2, 3, 4, 5\}$. In addition, all student models are trained only using D_{eval}. The codes and weights of all models are available at https://github.com/houshd/TS_Adv

4.1 Experiments

We select both neural networks as well as traditional models as the attacked model f. For the attacked neural network, we utilize a Fully Convolutional Network, whereas for the base traditional model, 1NN-Dynamic Time Warping Classifier is utilized.

To maintain the strictest definition of the black and white-box attacks, we utilize only the discrete class label of the attacked model for black-box attacks and utilize the probability distribution predicted by the classifier for white-box attacks. The only exception where a student-teacher network is not used is when performing a white-box attack on a FCN time series model, as the gradient information of a neural network can be directly exploited by an Adversarial Transformation Network (ATN). The performance of the adversarial model is evaluated on the original time series classification "teacher" model.

For every student model we train, we utilize the LeNet-5 architecture [36]. The student models are trained only using D_{eval}. We define a LeNet-5 time series classifier as a classical Convolutional Neural network following the structure : Conv (6 filters, 5x5, valid padding) - Max Pooling - Conv (16 filters, 5x5, valid padding) - Max Pooling - Fully Connected (120 units, relu) - Fully Connected (84 units, relu) - Fully Connected (number of classes, softmax).

The fully convolutional network is based on the FCN model proposed by *Wang et al.* [31]. It is comprised of 3 blocks, each comprised of a sequence of Convolution layer - Batch Normalization - ReLU activations. All convolutional kernels are initialized using the uniform he initialization proposed by He et al. [37]. We utilize [128, 256, 128] filters and kernel sizes of [8, 5, 3] to be consistent.

A strong determinisitic baseline model to classifiy time series is 1-NN DTW with 100% warping window. Due to its reliance on a distance matrix as a means of its classification, it cannot easily be used to compute an equivalent soft probabilistic representation. Since white-box attacks have access to the probability distribution predicted for each sample, we utilize this distance matrix in the computation of an equivalent soft probabilistic representation. The equivalent representation is such that if we compute the top class (class with highest probability score) on this representation, we get the exact same result as selecting the 1-nearest neighbor on the actual distance matrix.

To compute this soft probabilistic representation, consider a distance matrix V computed using a distance measure such as DTW between all possible pairs of samples between the two datasets being compared.

Algorithm 1 is an intermediate normalization algorithm which accepts a distance matrix V and the class labels of the training set y as inputs and computes an equivalent probabilistic representation that can directly be utilized to compute the 1-nearest neighbor. The $Soft{-}1NN$ algorithm selects all samples that belong to a class c_i, where $i \in \{1, \ldots, C\}$ as v_c, computes the maximum over all train samples for that class and appends the vector v_c_max to the list V_c. The concatenation of all of these lists of vectors in V_c then represents the matrix V', on which we then apply the $softmax$ function, as shown in Equation 6 with T set to 1, to represent this matrix V' as a probabilistic equivalent of the original distance matrix V.

An implicit restriction placed on Algorithm 1 is that the representation is equivalent only when computing the 1-nearest neighbor. It cannot be used to to represent the K-nearest neighbors and therefore cannot be used for K-nearest neighbor classification. However, in time series classification, the general consensus is on the use of 1-nearest neighbor classifiers and its variants to classify time series [6, 7, 25, 28, 38]. While the above algorithm has currently been applied to convert the 1NN-DTW distance matrix, it can also be applied to normalize any distance matrix utilized for 1-NN classification algorithms.

Algorithm 1: Equivalent probabilistic representation of the distance matrix for 1-nearest neighbor classification

```
Algorithm: Soft-1NN (V, y)
  Data: V is a distance matrix of shape $[N_{test}, N_{train}]$
        and y is the train set label vector of length
        $N_{train}$
  Result: Softmax normalized predictions p of shape
          $[N_{test}, C]$ and the discrete label vector q of
          length $N_{test}$
begin
    $V \longleftarrow (-V)$
    $uniqueClasses = Unique(y)$ //class labels
    $V_c$ = []
    for $c_i$ in $uniqueClasses$ do
        $v_c = V_{(y=c_i)}$ //$[N_{test}, N_{train}(y = c_i)]$
        $v_c\_max = max(v_c)$ //$[N_{test}]$
        $V_c$.append($v_c\_max$)
    end
    V' = concatenate($V_c$) // $[N_{test}$, number of classes$]$
    p = softmax(V') // $[N_{test}$, number of classes$]$
    q = argmax(p) // $[N_{test}]$
    return (p, q)
end
```

4.2 Results

Figures 2 and 3 depict the results from white-box attacks on 1-NN DTW and FCN that is applied on 42 UCR datasets. Further, Figures 4 and 5 represent the results from black-box attacks on 1-NN DTW and FCN classifiers that are trained on the same 42 UCR datasets. The detailed results can be found in Appendix A. The proposed methodology is successfully in capturing adversaries on all datasets. An example of an adversarial attack on the dataset "FordB" is shown in Figure 6.

4.2.1 Fraction of Successful Adversaries

The fraction of successful adversaries (the number of successful adversaries divided by the total number of correctly classified samples by the original classifier) and amount of perturbation per sample in each dataset can increase or decrease depending on the hyper-parameters that are tested on. For example, the dataset "Trace" has 0 adversaries for most of the attacks (black-box attack on 1-NN DTW, white-box attack on 1-NN DTW, black-box attack on FCN) when the Target Class is set to 1. However, if the target class is changed to 2, the number of adversaries generated increases to 9,3,1,37 for a black-box attack on 1-NN DTW, white-box attack on 1-NN DTW, black-box attack on FCN and white-box attack on FCN, respectively. These numbers can be higher if the hyper-parameters are changed. In addition, due to the loss function of the ATN, the target class has a significant impact on the adversary being generated. It is easier to generate adversaries for time series classes that are similar to each other.

A Wilcoxon signed-rank test is utilized to compare the fraction of successful adversaries generated by white-box and black-box attacks on FCN and 1-NN classifiers that

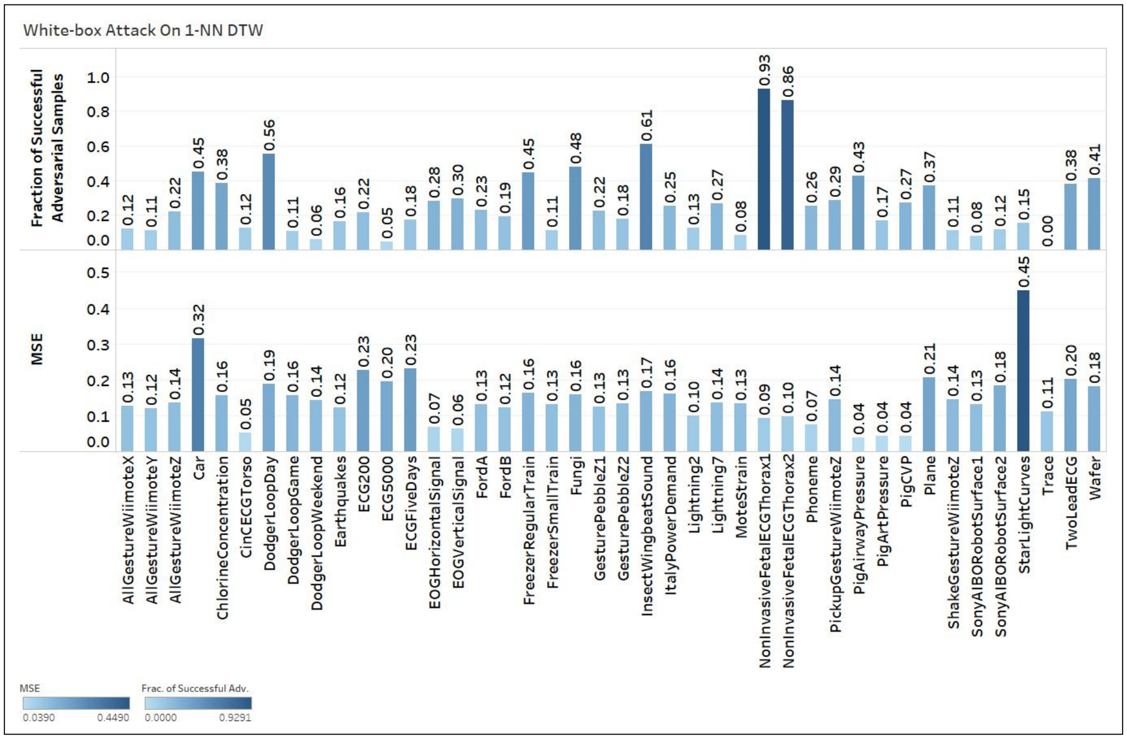


Fig. 2: White-box attack on 1-NN DTW that is trained on all 42 datasets

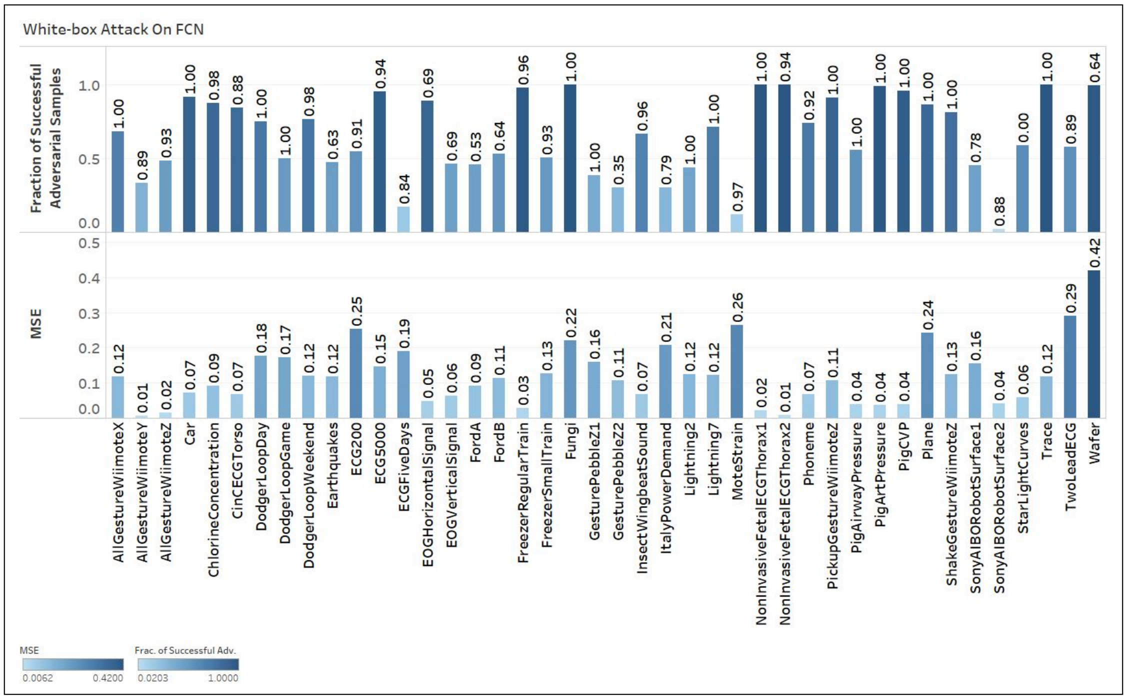


Fig. 3: White-box attack on FCN that is trained on all 42 datasets

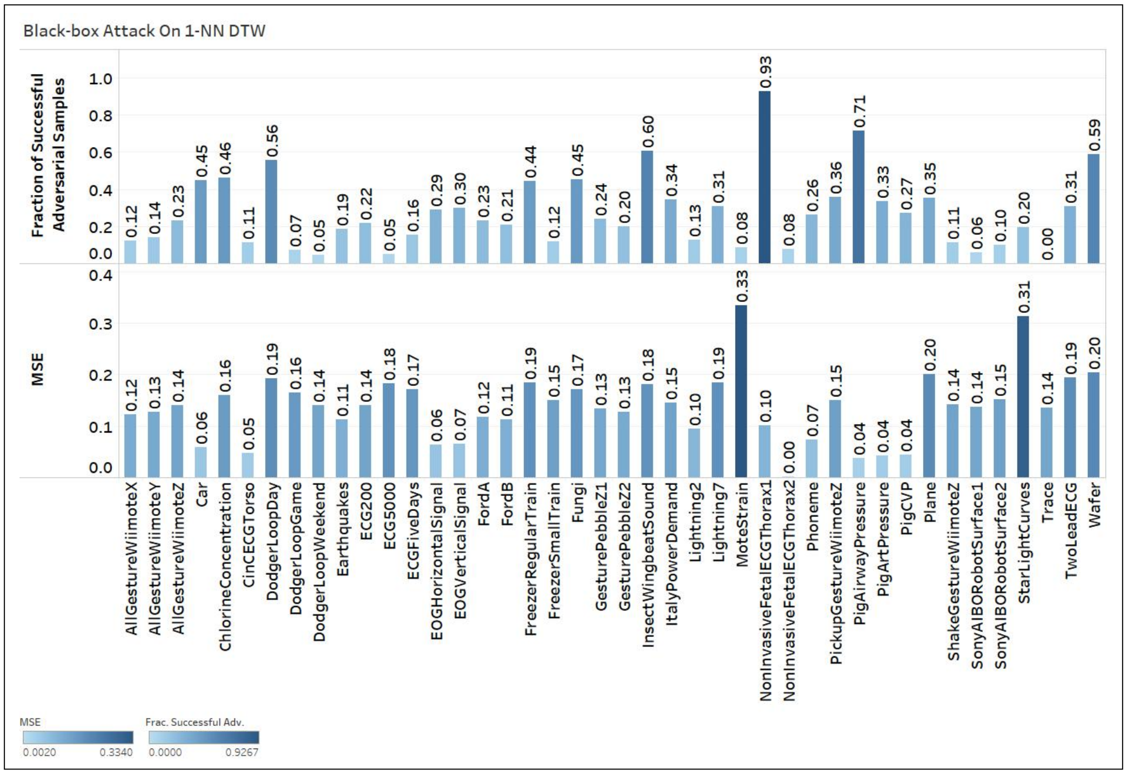


Fig. 4: Black-box attack on 1-NN DTW that is trained on all 42 datasets

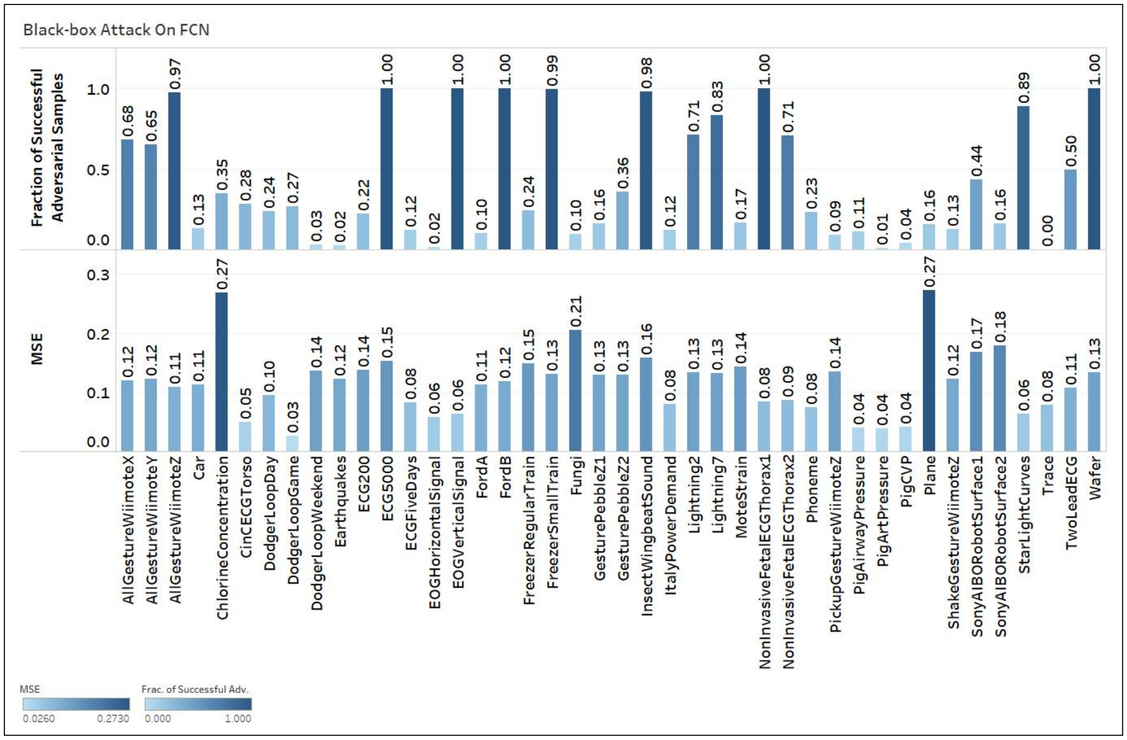


Fig. 5: Black-box attack on FCN that is trained on all 42 datasets

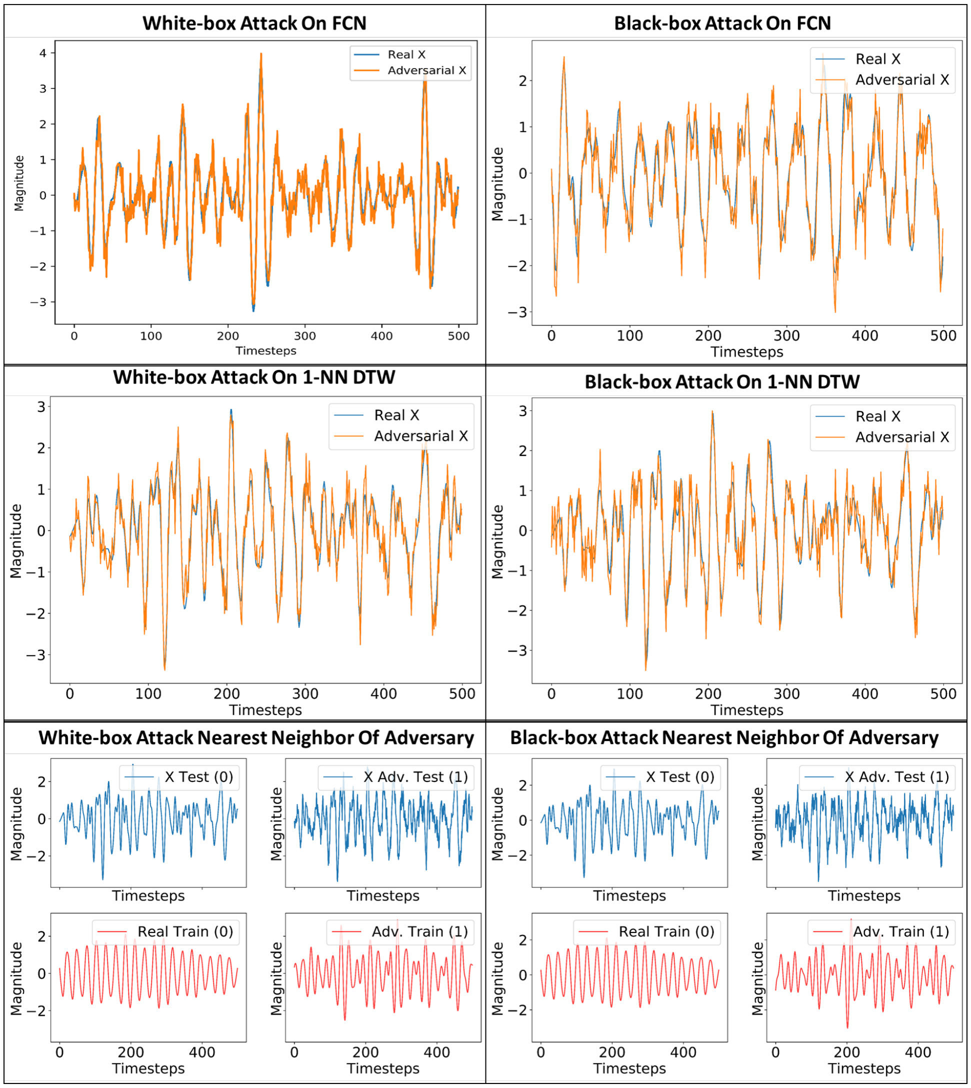


Fig. 6: A sample black-box and white-box attack on an FCN and 1-NN DTW classifier that is trained on the dataset "FordB". The last row of the figure depicts the nearest neighbor of the original and adversarial time series.

are trained on the 42 datasets, summarized in Table 1. Our results indicate that the FCN classifier is more susceptible to a white-box attack compared to a white-box attack on 1-NN DTW. It is to be noted that the white-box attack on the FCN classifier generates significantly more fraction of successful adversaries than its counterparts. This is because the white-box attack is directly on the FCN model and not on a student model that approximates the classifier behavior. We observe that the fraction of successful adversarial samples obtained from black-box attacks on FCN classifiers are approximately the same as the fraction of successful adversarial samples from either white-box or black-box attacks on DTW classifiers. A Wilcoxon signed-rank test confirms this observation by showing no statistical difference in the fraction of successful adversarial samples detected due to the black-box or white-box attacks on 1-NN DTW classifiers versus the fraction of successful adversarial samples obtained via black-box attacks on the FCN classifiers. In summary, we observe the largest fraction of successful adversarial samples for the FCN model when under a white-box attack. We also detect that 1-NN DTW classifiers under either attack and FCN classifiers under a black-box attack have approximately the same number of adversarial samples. These observations are important for future researchers who develop time series classifiers, as the fraction of successful adversarial samples generated under each methodology can be used as an evaluative metric to measure the robustness of a model.

TABLE 1: Wilcoxon signed-rank test comparing the fraction of successful adversarial between the different attacks

	White-box 1-NN DTW	Black-box FCN	White-box FCN
Black-box 1-NN DTW	2.278E-01	9.850E-02	5.949E-08
White-box 1-NN DTW		1.345E-01	2.731E-07
Black-box FCN			3..198E-03

4.2.2 *Comparison to a Baseline Model*

The proposed architectures are compared to a baseline adversarial attack, Fast Gradient Sign Method (FGSM) [39]. FGSM requires the gradient of each model. The black-box attacks and attacks on classical time series classification models is difficult when using it as it does not have access

TABLE 2: Wilcoxon signed-rank test comparing GATN with FGSM

	FGSM Instead of GATN
GATN Black-box 1-NN DTW	2.973E-03
GATN White-box 1-NN DTW	1.039E-01
GATN Black-box FCN	1.961E-07
GATN White-box FCN	2.664E-05

to its gradient. However, this is mitigated when applying it on the student network. We apply a Wilcoxon signed-rank test to compare the fraction of successful adversaries generated by FGSM to the fraction of successful adversaries generated by the respective black-box or white-box attack on FCN or 1-NN DTW using GATN. This is summarized in Table 2. The black-box attack on FCN and 1-NN DTW using GATN generates significantly more adversaries than when applying FGSM. However, FGSM generates significantly more adversaries than GATN when performing a white-box attack on FCN. A white-box attack on FCN using GATN performs statistically the same as a white-box attack on 1-NN DTW classifiers using FGSM. This indicates FGSM works better when the actual gradients of the classifier are accessible. GATN works better on black-box attacks.

4.2.3 Mean Squared Error (MSE)

We use MSE as a metric to depict how much perturbation is required for each dataset. Since, each dataset is normalized with a mean of 0 and a variance of 1, a lower MSE is better as each sample requires a lesser amount of perturbation. Hence, the optimal scenario would be when the MSE is closer to 0. The average MSE of adversarial samples after black-box attacks on FCN classifiers is significantly lower than the average MSE of the adversarial samples obtained via black-box and white-box attacks on 1-NN DTW classifiers, as observed in Table 5. A lower MSE indicates the black-box attack on FCN classifiers requires minimal perturbations per time series sample in comparison to the attacks on 1-NN DTW classifiers.

4.2.4 Defense

TABLE 3: Wilcoxon signed-rank test comparing testing accuracy of various adversarial attacks on initial time series classifiers and time series classifiers trained with adversarial samples

	After Defense
Black-box 1-NN DTW	7.656E-01
White-box 1-NN DTW	1
Black-box FCN	7.498E-01
White-box FCN	2.187E-03

TABLE 4: Wilcoxon signed-rank test comparing various adversarial attacks on initial time series classifiers and time series classifiers trained with adversarial samples

	After Defense
Black-box 1-NN DTW	3.394E-02
White-box 1-NN DTW	5.197E-01
Black-box FCN	1.546E-03
White-box FCN	4.016E-06

Defending against these adversarial attacks is important. We apply a simple defense to provide a starting point to prevent these attacks for future researchers.

Initially, a time series classification model is trained on the original training data. Subsequently, a black-box and white-box attack is performed on this model is using the original training data. We append the successful adversaries onto the training data and retrain the original time series classification model. Finally, a white-box and black-box attack is done on the new time series model.

Let us consider the testing accuracy (D_{eval} and D_{test}) of a model that has not undergone adversarial training (an undefended model that has not been trained on adversarial samples from a white box attack but has adversaries) as the baseline of comparison. After adversarial training of the above model, we find that the accuracy on the testing sets is better. However, as the defense strategy is not particularly strong, we also find that the increase in accuracy is not substantial. A summary of the Wilcoxon signed-rank test comparing the testing accuracy of all the attacks on both the classifiers using the defense mechanism are shown in Table 3. In addition, the fraction of successful adversarial samples generated by the white-box attack on 1-NN DTW classifiers does not significantly decrease after the defense mechanism (summarized in Table 4). The defense mechanism was able to make FCN classifiers more robust towards black-box and white-box attacks.

TABLE 5: Wilcoxon signed-rank test comparing the MSE between the different attacks

	White-box 1-NN DTW	Black-box FCN	White-box FCN
Black-box 1-NN DTW	7.362E-01	3.567E-02	1.240E-01
White-box 1-NN DTW		2.085E-03	6.856E-02
Black-box FCN			3.713E-01

4.2.5 Generalization

Finally, we test how well GATN generalizes onto an unseen dataset, D_{test}, such that GATN does not require any additional training. This is beneficial in situations where the time series adversarial samples are generated in constant time of a single forward pass of the GATN model without requiring further training. Such a generalization is uncommon to adversarial methodologies (Fast Gradient Sign Method or Jacobian-based Saliency Map Attack [39]) because they require retraining to generate adversarial samples. Our proposed methodology is robust, successfully generating adversarial samples on data that is unseen to both the GATN and the student models, for the respective targeted time series classification models. Figure 7 depicts the fraction of successful adversarial samples detected, on an unseen dataset, with a white-box and black-box attack on the 1-NN DTW classifiers and FCN classifiers. The white-box attack on the FCN classifier obtains the most adversarial samples per dataset. This is followed by a white-box and black-box attack on the 1-NN DTW, which show similar number of adversarial samples constructed. Finally, we find that the FCN classifier is the least susceptible to black-box attacks.

The unique consequence of this generalization is the application of trained GATN models for attacks that are feasible on real world devices, even for black box attacks. The deployment of a trained GATN with the paired student model affords a near constant-time cost of generating a reasonable number of adversarial samples. As the forward pass of the GATN requires few resources, and the student

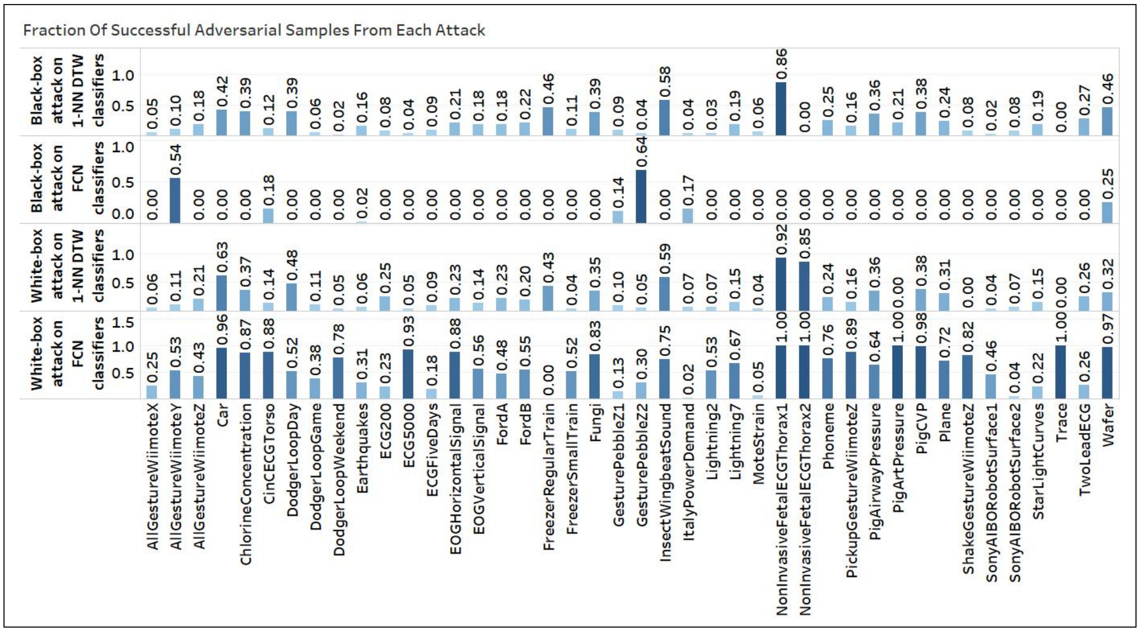


Fig. 7: Black-box and white-box attacks on FCN and 1-NN DTW classifiers that are tested on D_{test} without any retraining.

model is small enough to compute the input gradient ($\tilde{x}$) in reasonable time, these attacks can be constructed without significant computation on small, portable devices. Therefore, the fact that certain classifiers that are trained on certain datasets can be attacked without requiring any additional on-device training is concerning.

5 CONCLUSION & FUTURE WORK

In this paper, we extend [18] by proposing a model distillation technique to mimic the behavior of the various classical time series classification models and an adversarial transformation network to attack various time series datasets. We also tackle the issue of non differentiable target models (widely used in the time series domain) and propose the student-teacher framework as a general solution to perform a proxy attack on the target model, differentiable or not. In addition, we study the generalization capability of adversarial models on samples that have never before been seen before by the adversarial model. The proposed methodology is applied onto 1-NN DTW and Fully Connected Network (FCN) that are trained on 42 University of California Riverside (UCR) datasets. All 42 datasets were susceptible to not-targeted attacks. The FCN model is more prone to GATN white-box attacks than its counterparts. GATN is able to generate a larger fraction of successful adversarial attacks on black-box attacks than FGSM. We also show how a simple defense mechanism is able to reduce the fraction of successful adversarial samples on various GATN attacks. We recommend future researchers that develop time series classification models to consider model robustness as an evaluative metric and to incorporate adversarial data samples into their training data sets in order to further improve resilience to adversarial attacks. Further research should be done in developing models that can generate targeted adversaries on time series classification models. The goal of these targeted adversarial models is to generate adversaries where time series classification models will miss-classify the adversaries into a particular label. Finally, In all our experiments the scale of the time series data was known to the attacker. Additional research can be done in generating time series adversaries where the scale of the time series data is unknown to the attacker.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] J. Boyan, D. Freitag, and T. Joachims, “A machine learning architecture for optimizing web search engines,” in *AAAI Workshop on Internet Based Information Systems*, 1996, pp. 1–8.

[3] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The adaptive web*. Springer, 2007, pp. 325–341.

[4] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, “A deep learning approach to on-node sensor data analytics for mobile or wearable devices,” *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 56–64, 2017.

[5] K. Sirisambhand and C. A. Ratanamahatana, “A dimensionality reduction technique for time series classification using additive representation,” in *Third International Congress on Information and Communication Technology*. Springer, 2019, pp. 717–724.

[6] A. Sharabiani, H. Darabi, A. Rezaei, S. Harford, H. Johnson, and F. Karim, “Efficient classification of long time series by 3-d dynamic time warping,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2688–2703, 2017.

[7] A. Sharabiani, H. Darabi, S. Harford, E. Douzali, F. Karim, H. Johnson, and S. Chen, “Asymptotic dynamic time warping calculation with utilizing value repetition,” *Knowledge and Information Systems*, pp. 1–30, 2018.

[8] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, “Fast time series classification using numerosity reduction,” in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 1033–1040.

[9] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, “The ucr time series classification archive,” October 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[10] K. R. Mopuri, A. Ganeshan, and V. B. Radhakrishnan, “Generalizable data-free objective for crafting universal adversarial perturbations,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[11] M. Zhang, K. T. Ma, J. Lim, Q. Zhao, and J. Feng, "Anticipating where people will look using adversarial networks," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[12] I. Oregi, J. Del Ser, A. Perez, and J. A. Lozano, "Adversarial sample crafting for time series classification with elastic similarity measures," in *International Symposium on Intelligent and Distributed Computing*. Springer, 2018, pp. 26–39.

[13] C. Song, H.-P. Cheng, H. Yang, S. Li, C. Wu, Q. Wu, Y. Chen, and H. Li, "Mat: A multi-strength adversarial training method to mitigate adversarial attacks," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 476–481.

[14] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *arXiv preprint arXiv:1809.04356*, 2018.

[15] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[16] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *arXiv preprint arXiv:1801.04503*, 2018.

[17] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

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

[19] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *arXiv preprint arXiv:1801.00553*, 2018.

[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[21] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[22] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv:1804.07998*, 2018.

[23] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," *arXiv preprint arXiv:1707.07328*, 2017.

[24] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," *arXiv preprint arXiv:1801.01944*, 2018.

[25] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[26] A. Kampouraki, G. Manis, and C. Nikou, "Heartbeat time series classification with support vector machines," *IEEE Trans. Information Technology in Biomedicine*, vol. 13, no. 4, pp. 512–518, 2009.

[27] P. Schäfer and U. Leser, "Fast and accurate time series classification with weasel," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 637–646.

[28] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: the collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[29] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 668–676.

[30] R. J. Kate, "Using dynamic time warping distances as features for improved time series classification," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 283–312, 2016.

[31] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1578–1585.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[33] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017.

[34] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.

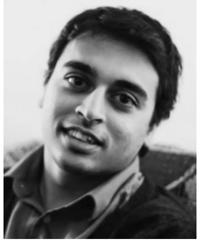
[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[36] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, p. 20, 2015.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[38] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[39] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, 2019.



Fazle Karim received the B.Sc. degree in industrial engineering from the University of Illinois at Urbana-Champaign in 2012, the M.Sc. degree in industrial engineering from the University of Illinois at Chicago in 2016. He is currently pursuing the Ph.D. degree with the Mechanical and Industrial Engineering Department, University of Illinois at Chicago. He is also the Lead Data Scientist with Prominent Laboratory, the university's foremost research facility in process mining. His research interests include education data mining, health care data mining, and time series analysis.

Somshubra Majumdar received the B.S. degree in computer engineering from the University of Mumbai in 2016 and received his M.S. degree in computer science with the University of Illinois at Chicago in 2018. He is currently an aspiring Artificial Intelligence Researcher. His research interests lie in the domain of image classification and segmentation using convolutional neural networks, time series classification using recurrent neural networks, speech recognition and machine learning.

Houshang Darabi (S'98-A'00-M'10-SM'14) received the Ph.D. degree in Industrial and Systems Engineering from Rutgers University, New Brunswick, NJ, USA, in 2000.

He is currently an Associate Professor with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago (UIC), and also an Associate Professor with the Department of Computer Science, UIC. He has been a contributing author of two books in the areas of scalable enterprise systems and reconfigurable discrete event systems. His research has been supported by several federal and private agencies, such as the National Science Foundation, the National Institute of Standard and Technology, the Department of Energy, and Motorola. He has extensively published on various automation and project management subjects, including wireless sensory networks for location sensing, planning and management of projects with tasks requiring multi-mode resources, and workflow modeling and management. He has published in different prestigious journals and conference proceedings, such as the IEEE Transaction on Robotics and Automation, the IEEE Transactions on Automation Science and Engineering, and the IEEE Transactions on Systems, Man, and Cybernetics, and Information Sciences. His current research interests include the application of data mining, process mining, and optimization in design and analysis of manufacturing, business, project management, and workflow management systems.

APPENDIX
DETAILED RESULTS

Dataset	Black-Box Attack on DTW Models		White-Box Attack on DTW Models		Black-Box Attack on FCN Models		White-Box Attack on FCN Models	
	Fraction of Success Adv.	MSE	Fraction of Success Adv.	MSE	Fraction of Success Adv.	MSE	Fraction of Success Adv.	MSE
AllGestureWiimoteX	0.12	0.123	0.12	0.126	0.679	0.12	0.68	0.119
AllGestureWiimoteY	0.14	0.128	0.111	0.121	0.65	0.122	0.333	0.006
AllGestureWiimoteZ	0.232	0.14	0.219	0.137	0.972	0.109	0.481	0.015
Car	0.45	0.058	0.45	0.316	0.13	0.113	0.917	0.073
ChlorineConcentration	0.463	0.16	0.383	0.156	0.349	0.269	0.873	0.092
CinCECGTorso	0.111	0.048	0.125	0.051	0.281	0.049	0.843	0.067
DodgerLoopDay	0.556	0.193	0.556	0.188	0.238	0.095	0.75	0.177
DodgerLoopGame	0.073	0.164	0.109	0.157	0.265	0.026	0.5	0.172
DodgerLoopWeekend	0.046	0.141	0.062	0.143	0.032	0.137	0.766	0.12
Earthquakes	0.187	0.113	0.165	0.123	0.024	0.122	0.476	0.118
ECG200	0.216	0.141	0.216	0.226	0.222	0.138	0.543	0.254
ECG5000	0.05	0.183	0.048	0.196	1	0.153	0.949	0.146
ECGFiveDays	0.155	0.172	0.176	0.231	0.121	0.083	0.173	0.19
EOGHorizontalSignal	0.288	0.064	0.28	0.068	0.016	0.058	0.889	0.049
EOGVerticalSignal	0.297	0.065	0.297	0.063	1	0.064	0.462	0.063
FordA	0.233	0.117	0.231	0.131	0.1	0.113	0.455	0.091
FordB	0.207	0.112	0.194	0.122	1	0.119	0.532	0.113
FreezerRegularTrain	0.443	0.185	0.448	0.163	0.245	0.149	0.977	0.029
FreezerSmallTrain	0.117	0.15	0.114	0.131	0.994	0.131	0.506	0.128
Fungi	0.452	0.172	0.479	0.158	0.096	0.205	1	0.22
GesturePebbleZ1	0.239	0.134	0.224	0.125	0.161	0.129	0.385	0.159
GesturePebbleZ2	0.2	0.127	0.18	0.133	0.357	0.13	0.3	0.108
InsectWingbeatSound	0.605	0.181	0.608	0.167	0.979	0.158	0.663	0.068
ItalyPowerDemand	0.342	0.145	0.252	0.161	0.122	0.08	0.3	0.208
Lightning2	0.125	0.095	0.125	0.099	0.714	0.133	0.438	0.124
Lightning7	0.308	0.185	0.269	0.136	0.833	0.132	0.714	0.123
MoteStrain	0.084	0.334	0.083	0.134	0.168	0.143	0.12	0.264
NonInvasiveFetalECGThorax1	0.927	0.101	0.929	0.092	1	0.084	1	0.021
NonInvasiveFetalECGThorax2	0.078	0.002	0.863	0.097	0.708	0.087	1	0.008
Phoneme	0.263	0.074	0.255	0.074	0.231	0.075	0.737	0.068
PickupGestureWiimoteZ	0.357	0.15	0.286	0.144	0.091	0.135	0.909	0.107
PigAirwayPressure	0.714	0.038	0.429	0.039	0.111	0.04	0.556	0.039
PigArtPressure	0.333	0.043	0.167	0.044	0.011	0.038	0.989	0.037
PigCVP	0.273	0.044	0.273	0.042	0.043	0.041	0.955	0.039
Plane	0.353	0.201	0.373	0.207	0.157	0.273	0.863	0.243
ShakeGestureWiimoteZ	0.111	0.142	0.111	0.144	0.125	0.122	0.813	0.125
SonyAIBORobotSurface1	0.058	0.137	0.079	0.132	0.436	0.168	0.451	0.155
SonyAIBORobotSurface2	0.1	0.151	0.118	0.184	0.16	0.179	0.02	0.042
StarLightCurves	0.196	0.313	0.154	0.449	0.889	0.064	0.588	0.059
Trace	0	0.136	0	0.111	0	0.078	1	0.117
TwoLeadECG	0.307	0.194	0.378	0.201	0.497	0.108	0.575	0.29
Wafer	0.589	0.204	0.414	0.182	1	0.133	0.991	0.42