Generating Adversarial Samples on Multivariate Time Series using Variational Autoencoders

Samuel Harford, *Student Member, IEEE*, Fazle Karim, and Houshang Darabi, *Senior Member, IEEE*

***Abstract*—Classification models for multivariate time series have drawn the interest of many researchers to the field with the objective of developing accurate and efficient models. However, limited research has been conducted on generating adversarial samples for multivariate time series classification models. Adversarial samples could become a security concern in systems with complex sets of sensors. This study proposes extending the existing gradient adversarial transformation network (GATN) in combination with adversarial autoencoders to attack multivariate time series classification models. The proposed model attacks classification models by utilizing a distilled model to imitate the output of the multivariate time series classification model. In addition, the adversarial generator function is replaced with a variational autoencoder to enhance the adversarial samples. The developed methodology is tested on two multivariate time series classification models: 1-nearest neighbor dynamic time warping (1-NN DTW) and a fully convolutional network (FCN). This study utilizes 30 multivariate time series benchmarks provided by the University of East Anglia (UEA) and University of California Riverside (UCR). The use of adversarial autoencoders shows an increase in the fraction of successful adversaries generated on multivariate time series. To the best of our knowledge, this is the first study to explore adversarial attacks on multivariate time series. Additionally, we recommend future research utilizing the generated latent space from the variational autoencoders.**

***Index Terms*—Adversarial machine learning, deep learning, multivariate time series, perturbation methods.**

I. Introduction

MACHINE learning drives many facets of society including online search engines, content analysis on social networks, and smart appliances such as thermostats. In computer vision, machine learning is commonly used for recognizing objects in images or video [1], [2]. In natural language processing for transcribing speech into text, matching news articles, and selecting relevant results [3], [4]. In healthcare, to diagnose and predict patient's survival [5], [6].

Time series classification is a subfield of machine learning that has received a lot of attention over the past several decades [7], [8]. A time series can be univariate or multivariate. Univariate time series are a time ordered collection of measurements from a single source. Multivariate time series are time ordered collections of measurements from at least two sources [9]. While most time series research focused on univariate time series [10]–[17], research in the area of multivariate time series has increased over the past decade [18]–[21]. Multivariate time series classification is applied in fields including healthcare [22], manufacturing [23], and action recognition [24]. Time series classification models aim to capture the underlying patterns of the training data and generalize the findings to classify unseen testing data. The field of multivariate time series classification has primarily focused on more traditional algorithms, including 1-nearest neighbor dynamic time warping (1-NN DTW) [25], WEASEL+MUSE [19], and Hidden-Unit Logistic Model [18]. With the surge in computational power, deep neural networks (DNNs) are increasingly being applied in machine learning applications [26], [27]. Due to their simplicity and effectiveness, DNN are becoming excellent methods for time series classification [20], [28], [29].

While machine learning and deep learning techniques allow for many important and practical problems to be automated, many of the classifiers have proven to be vulnerable to adversarial attacks [30], [31]. An adversarial example is a sample of input data that has been slightly modified in a way that makes the classifier mislabel the input sample. Similar examples can be created by generative adversarial networks (GANs), which look to generate data instances similar to that of the modeling data but not aim to manipulate classifiers [32]–[34]. In the field of computer vision, it has been shown that image recognition models can be tricked by adding information to an image that is not noticeable to the human eye [31]. Although DNNs are powerful models for a number of classification tasks, they have proven to be vulnerable to adversarial attacks when minor carefully crafted noise is added to an input [35], [36]. These vulnerabilities have a harmful impact on real-world applicability where classification models are incorporated in the pipeline of a decision making process [37]. A significant amount of research in adversarial attacks has focused on computer vision. Papernot *et al.*'s work has shown that it is easy to transfer adversarial attacks on a particular classifier to other similar classifiers [38]. Recent years have shown an increased focus on adversarial attacks in the field of time series classification [39]–[42]. However, these studies have been limited to attacks on univariate time series.

Many strategies have been developed for the generation of

Manuscript received January 1, 2021; revised February 27, 2021 and April 5, 2021; accepted May 7, 2021. Recommended by Associate Editor MengChu Zhou. *(Corresponding author: Samuel Harford.)*

Citation: S. Harford, F. Karim, and H. Darabi, " Generating adversarial samples on multivariate time series using variational autoencoders," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 9, pp. 1523–1538, Sept. 2021.

The authors are with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: sharfo2@uic.edu; karim1@uic.edu; hdarabi@uic.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2021.1004108

adversarial samples to trick DNN models. Most techniques work by targeting the gradient information of DNN classifiers [43]–[45]. In time series classification, classifiers used to monitor the electrocardiogram (ECG) signals of a patient can be manipulated to misclassify important changes in a patient's status. When attacking traditional time series classifiers, it is important to note that the model mechanics are non-differentiable. For this reason, attacks are not able to directly utilize the gradient information from traditional models. There are two main types of attacks. Black-box (BB) attacks rely only on the models output information, the training process and architecture of the target models. White-box (WB) attacks gives the attacker all information about the attacked model, including the training data set, the training algorithm, the model's parameters and weights, and the model architecture [45].

This study proposes extending the gradient adversarial transformation network (GATN) methodology to attack multivariate time series and utilize different adversarial generators [40]. GATN is extended by exploring the use of adversarial autoencoders to generate adversarial samples under both black-box and white-box attacks. The GATN methodology works by training a student model with the objective of replicating the output behavior of a targeted multivariate time series classifier. The targeted model is referred to as a teacher model. Once the student model has learned to mimic the behavior of the teacher model, the GATN model can learn to attack the student model. This study uses the 1-NN DTW and fully convolutional network (FCN) as the teacher models. Given a trained student model, the proposed multivariate gradient adversarial transformation network (MGATN) is then trained to attack the student model. Our methodologies are applied to 30 multivariate time series bench marks from the University of East Anglia (UEA) and the University of California, Riverside (UCR) [46]. To the best of our knowledge, this is the first study to conduct adversarial attacks on multivariate time series.

The remainder of this paper is structured as follows: Section II provides a background on the utilized multivariate time series classification models and techniques for creating adversaries. Section III details our proposed methodologies. Section IV presents the experiments conducted on the benchmark multivariate time series models. Section V illustrates and discusses the results of the experiments. Section VI concludes the paper and proposes future work.

II. Definitions and Background

For the task of multivariate time series classification, each instance is a set of time series which may be of varying lengths.

Definition 1: **Univariate Time Series** $T = t_1; t_2; \ldots; t_n$ is a time ordered set of length n continuous values. The length of a time series is equal to the number of values. A time series dataset is a collection of time series instances.

This work focuses on multivariate time series.

Definition 2: **Multivariate Time Series** consist of M univariate time series, where the M is the number of dimensions and $M \geq 2$

$$\begin{aligned} T_1 &= t_{1,1}; t_{2,1}; t_{3,1}; \ldots t_{n,1} \\ T_2 &= t_{1,2}; t_{2,2}; t_{3,2}; \ldots t_{n,2} \\ &\cdots \\ T_M &= t_{1,M}; t_{2,M}; t_{3,M}; \ldots t_{n,M}. \end{aligned}$$

Each multivariate time series instance has a corresponding class label. The objective of multivariate time series classification is to develop models that can accurately identify the class label of an unseen instance.

A. Multivariate Time Series Classifiers

1) Multivariate 1-Nearest Neighbor Dynamic Time Warping

Dynamic time warping (DTW) has proven to be an effective distance metric in time series applications [47], [48]. In univariate time series classification, the value of k in k-NN DTW is typically set to 1 [49]. For benchmarking purposes, multivariate time series classification algorithms compare themselves to 1-NN DTW instead of optimizing the value of k or each dataset [50]. The 1-nearest neighbor algorithm leverages a distance metric to assign the label of the closest training sample to an unknown testing sample. The distance between two multivariate time series, Q and C, is calculated as the cumulative distances of all dimensions independently measured under DTW [51]. The DTW distance of the mth dimension of Q and the mth dimension of C is written as DTW (Q_m,C_m). The lengths of time series Q and C are denoted as q and c, respectively. As defined by Shokoohi-Yekta *et al.* [51], the DTW matrix of dimension m is calculated as

$$\begin{aligned} D_m(i,j) = (Q_{i,m} - C_{j,m})^2 + \min[&D_m(i-1, j-1), \\ &D_m(i-1, j), D_m(i, j-1)] \end{aligned} \quad (1)$$

where i refers to the position on time series Q and j refers to the position on time series C. The warping matrix is initialized as

$$D_m(1,1) = 0;\ D_m(1, 2 \ldots q) = \infty;\ D_m(2 \ldots c, 1) = \infty. \quad (2)$$

Given the calculated warping matrix, the final distance for each dimension is $D_m(q,c)$. The distance between the multivariate time series is then calculated as

$$DTW_M(Q,C) = \sqrt{\sum_{m=1}^{M} D_m(q,c)}. \quad (3)$$

2) Multi Fully Convolutional Network

Inspired by their success in the fields of computer vision and natural language processing, deep learning models have been successfully applied to the task of time series classification [20], [28], [52]. The multivariate fully convolutional network (Multi-FCN) is one of the first deep learning networks used for the task of multivariate time series classification. Fig. 1 illustrates the Multi-FCN network. The three convolutional layers output filters of 128, 256, and 128 with kernels sizes of 8, 5, and 3, respectively. The model outputs a class probability for use in class labeling.

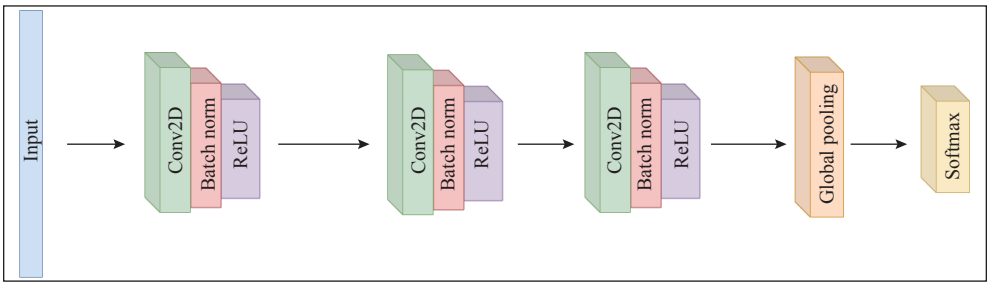

Fig. 1. The Multi-FCN architecture.

B. Adversarial Transformation Network

Multiple approaches for generating adversarial samples have been proposed to attack classification models. These methods have focused on classification tasks in the field of computer vision. Most methods use the gradient information with respect to the input sample or directly solving an optimization problem on the input sample. Baluja and Fischer [53] introduce adversarial transformation network (ATN), a neural network that transforms an input into an adversarial example by using a target network. ATNs may be trained for black-box or white-box attacks. ATNs work by first using a self-supervised method to train a feed-forward neural network. The model works by taking an original input sample and making slight modifications to the classifier output with the objective of matching the adversarial target. An ATN is defined as a neural network

$$g_f(x) : x \in \mathcal{X} \rightarrow x' \tag{4}$$

where g is defined as an ATN, f is the target network which outputs a probability distribution across class labels, x is an original data point in $\mathcal{X}$, x' is the generated adversary, and $x' \sim x$, but argmax $f(x) \neq$ argmax $f(\hat{x})$. To find g_f, the following loss is used:

$$L = \beta \times L_x(g_f(\mathbf{x}), \mathbf{x}) + L_y(f(g_f(\mathbf{x})), f(\mathbf{x})) \tag{5}$$

where L_x is a loss function for the inputs (e.g., L_2 loss function), L_y is the specially formed loss function for the output of f to avoid learning the identity function, and β is a weight to balance L_x and L_y. L_y determines whether or not the ATN is targeted. The target class refers to the class for which the adversary will cause the classifier to output the maximum value. Baluja and Fischer [53] define L_y as $L_y(y', y) = L_2(y', r(y, t))$, where t represents the target class, $y = f(x)$, $y' = f(g_f(x))$, and $r(\cdot)$ is a reranking function that modifies y such that $y_k < y_t$, $\forall k \neq t$. Baluja and Fischer [53] propose that the reranking function $r(y, t)$ can either be a simple one hot encoding function or can take advantage of the information in $\mathbf{y}$ to optimize based on reconstruction. They the define this reranking function as

$$r_\alpha(\mathbf{y}, t) = norm\left(\left\{ \begin{array}{ll} \alpha \times \max(\mathbf{y}) & \text{if } k = t \\ y_k & \text{otherwise} \end{array} \right\}_{k \in \mathbf{y}}\right) \tag{6}$$

where α is a weighting parameters. An $\alpha > 1$ defines how much larger y_t should be than the current max classification. The $norm(\cdot)$ function rescales its input to be a valid probability distribution.

C. Transferability Property

The transferability property of an adversarial sample is the property that the same adversary produced to mislead a targeted model f can mislead another model s, regardless of model architecture [54], [55]. Papernot *et al.* [38] further study this property and propose a black-box attack by training a local substitute network, s, to replicate the target model, f. The local substitute network s is trained using generated samples and the targeted model f is used to get the output label of generated samples. The transferability property is utilized in adversarial attacks by exploiting the full local model s on the targeted model f with the objective of achieving misclassifications. Papernot *et al.* [38] show that this method can be applied on both DNN and traditional machine learning classifiers.

D. Knowledge Distillation

A key strategy for reducing the cost of inference is model compression, also known as knowledge distillation [56]. The idea of knowledge distillation is to replace the original, computationally expensive model with a smaller model that requires less memory, parameters and computational time. This idea works by training a smaller student model s to mimic the behavior of the larger teacher model f. The teacher model f has its knowledge distilled into the student model s by minimizing a loss function between the set of networks. This loss function aims to output the same class probability vector on the student model s as that generated by the teacher model f. Hinton *et al.* [57] state that the commonly used softmax function results in a skewed probability distribution where the correct probability class is very close to 1 and the remaining classes are close to 0. To reduce the resulting skewness in the probability class vector, Hinton *et al.* [57] recommends adjusting the output such that

$$q_i = \sigma(z; T) = \exp(z_i/T) \Big/ \sum_j \exp(z_j/T) \tag{7}$$

where q_i is the adjusted probability of class i, σ is the adjusted output formula, z_i is the output logit, T is a temperature factor normally set to 1, and j is the list of possible classes. Higher values of T produce softer probability distributions over classes.

E. Gradient Adversarial Transformation Network

The GATN is a method for black-box and white-box adversarial attacks on univariate time series [40]. GATNs utilize an adversarial transformation network (ATN) as the

basic structure to generate adversarial samples. The ATN is defined as $g_f(x): x \rightarrow \hat{x}$, where f is the targeted model, x is the input time series and $\hat{x}$ is the generated adversary. The GATN modifies the ATN by utilizing the gradient information of the input sample for the target class prediction from the targeted classifier. The GATN is defined as $g_f(x;\tilde{x}): [x;\tilde{x}] \rightarrow \hat{x}$, where $\tilde{x}$ is the gradient information for the input sample. The gradient information is calculated as

$$\tilde{x} = \frac{\partial f_t}{\partial x} \tag{8}$$

where x is the input time series, f_t is the probability of the input being classified as class t, and the $[;]$ operation concatenates two vectors.

When training the student model, the loss function defined as

$$L_{\text{transfer}} = \gamma \times L_{\text{distillation}} + (1-\gamma) \times L_{\text{student}} \tag{9}$$

$$L_{\text{distillation}} = \mathcal{H}(\sigma(z_f; T=\tau), \sigma(z_s; T=\tau)) \tag{10}$$

$$L_{\text{student}} = \mathcal{H}(y, \sigma(z_s; T=1)) \tag{11}$$

where $\mathcal{H}$ is the cross entropy loss function, z_s and z_f are the unnormalized logits of the student and teacher networks, respectively, $\sigma(\cdot)$ is the scaled-softmax operation described in (7), y is the ground truth labels, γ is a balancing parameter between the two losses, and τ is a temperature parameter to scale the the student s and teacher f models (τ is kept constant at 10). White-box attacks use a γ value of 0.5, where the attack assigns equal weight to $L_{\text{distillation}}$ and L_{student}. Black-box attacks use a γ value of 1, where the attack is dependent only on $L_{\text{distillation}}$.

F. Adversarial Autoencoders

Makhzani *et al*. [58] propose the adversarial autoencoder (AAE), which is a probabilistic autoencoder that utilizes the knowledge learned from generative adversarial networks (GANs) [59]. The GANs are used to perform variational inference by matching the aggregated posterior of the hidden code vector of the autoencoder with an arbitrary prior distribution. AAE are similar to standard autoencoders [60], where the objective is to accurately reconstruct the original input, subject to a limited amount of added noise.

In addition to simple autoencoders, Makhzani *et al*. [58] propose the use of variational autoencoders (VAE) [61]. VAEs provide a formulation in which the encoding z is interpreted as a latent variable in a probabilistic generative model, a probabilistic decoder is defined by a likelihood function $p_\theta(x|z)$ and parameterized by θ. Alongside a prior distribution $p(z)$ over the latent variables, the posterior distribution $p_\theta(z|x) \propto p(z)p_\theta(x|z)$ can then be interpreted as a probabilistic encoder. Fig. 2 illustrates this process. Ideally, the trained latent vector creates clusters that are as close as possible to each other while still being distinct, allowing smooth interpolation, and enabling the construction of new samples. To better create the latent vector a Kullback-Leibler (KL) divergence is introduced into the loss function [62]. The KL divergence between two probability distributions simply measures how much they diverge from each other.

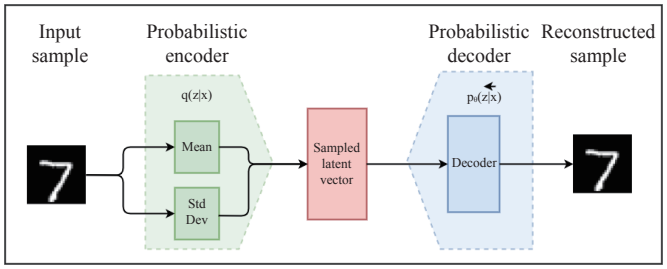

Fig. 2. The VAE architecture.

Minimizing the KL divergence here means optimizing the probability distribution parameters to closely resemble that of the target distribution. The KL divergence defined as

$$D_{KL} = -\sum_{i=1}^{n} P(i) \times \log\left(\frac{Q(i)}{P(i)}\right) \tag{12}$$

where n is the length of the output, P is the probabilistic distribution of the original data, and Q is the probabilistic distribution of the adversarial data. Due to the limited information about the actual probabilistic distributions, Variational Inference is used to simplify the divergence calculation using known information [63]. The KL diverence is simplified to

$$D_{KL} = \sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1 \tag{13}$$

where μ is the mean output of the encoding layer, and σ is the standard deviation output of the encoding layer.

III. Proposed Methodology

A. Multivariate Gradient Adversarial Transformation Network

The use of the GATN has proven to be a successful method for developing adversarial samples on univariate time series [40]. However, GATN and other adversarial attacks on time series have not been applied on multivariate time series. In this work, we purpose multivariate gradient adversarial transformation network (MGATN) which extends the GATN model for use in generating multivariate adversarial time series. In addition, we explore the use of Variational Encoders and Convolution Variational Encoders as alternative generator functions.

MGATN is defined as $g_f(X,\tilde{X}): [X;\tilde{X}] \rightarrow \hat{X}$, where X is an input multivariate time series, $\tilde{X}$ is the gradient information for each dimension of the multivariate input, and $\hat{X}$ is a multivariate adversarial sample. Fig. 3(c) shows how the architecture of this simple generator uses the input and gradient information to generate an adversarial sample. In this work, we explore the use of variational autoencoders and convolutional variational autoencoders (CVAE) as models to generate adversaries on multivariate time series. Figs. 3(d) and 3(e) show how a VAE and CVAE can be utilized to generate adversarial samples from an input sample and gradient information. MGATN method with a VAE generator is referred to as MGATN_V. In addition to a VAE generator, we propose the use of a CVAE. The CVAE generator has a

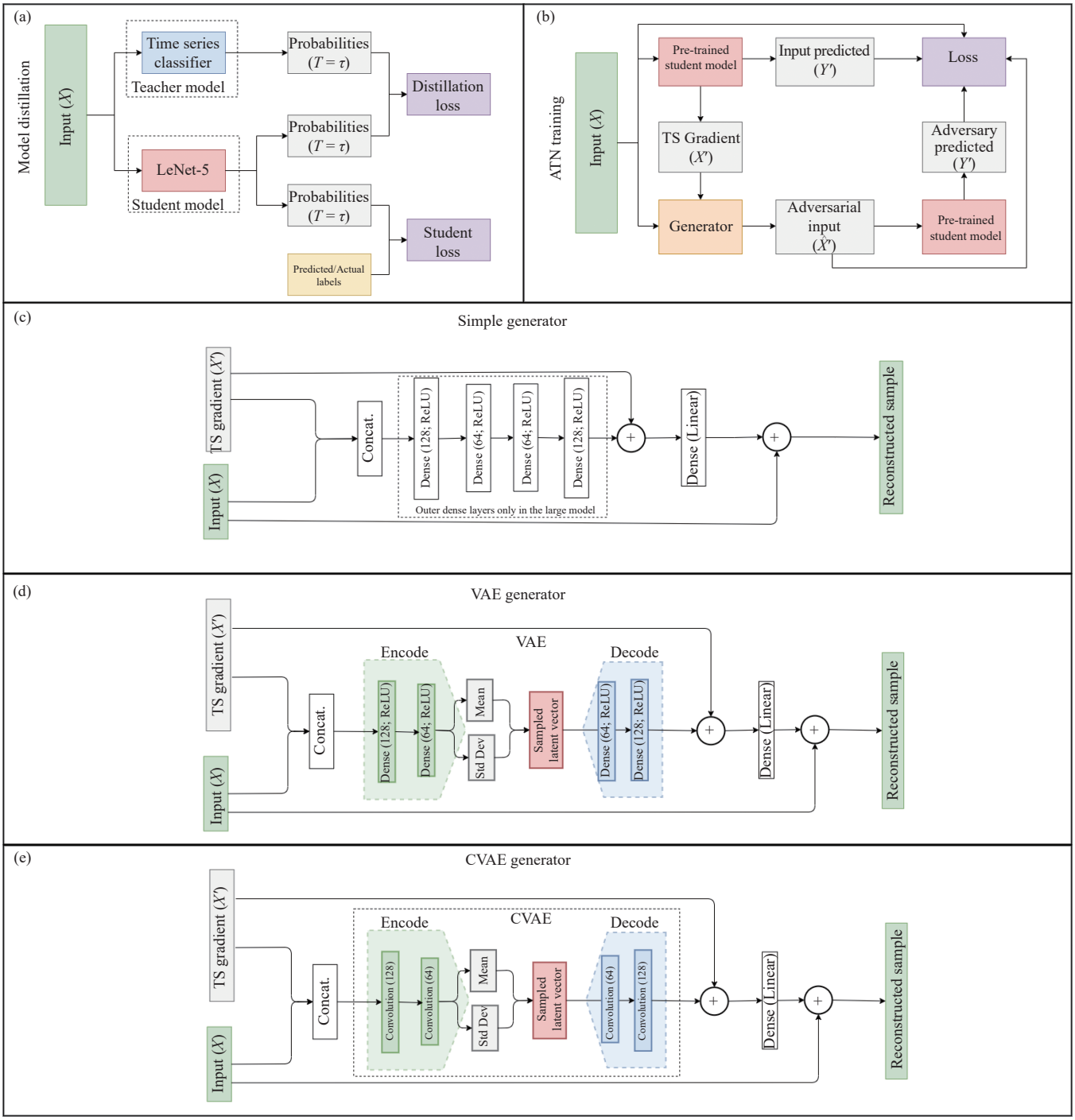

Fig. 3. The subfigures (a) and (b) illustrate the methodology of training the model distillation used the attacks. The subfigures (c) and (d) illustrate the methodology utilized to attack a time series classifier.

similar architecture to the VAE generator, where the dense layers of the encoder and decoder are modified to 1-D convolutional layers. MGATN method with a CVAE generator is referred to as MGATN$_{CV}$. The MGATN$_{CV}$ generator encodes the samples by first passing through an intermediate convolutional layer of filter size 32 and kernel size 5, then feeds to two branched convolutional layers of filter size 16 and kernel size 5. The two convolutional layers then pass through a sampling layer. The final step decodes the sampling layer with two convolutional layers of filter size 32 and the original time series shape, both with kernel sizes or 5.

B. Training Methodology

This subsection discusses the training purpose for the MGATN methodology. Figs. 3(a) and 3(b) illustrates the full framework architecture defined in [15]. The framework architecture is similar to that of GATN, with notable modifications. The input to the framework is multivariate, where the shape of inputs changes from (batch size, length, value) to (batch size, channel, length, value). This increase in shape requires the student model to be modified to a LeNet-5 with 2D Conv layers. In addition, we explore different

generator functions to create adversarial samples. The tested generator functions include a fully connected network, a variational autoencoder, and a convolutional variational autoencoder. Figs. 3(c)–3(e) illustrate the generator architectures. Fig. 3(c) illustrates the simple generator where the MGATN uses two dense layers in the autoencoder block. For a fair comparison of model parameters, a MGATN_L is included in the experiments and results. MGATN_L and MGATN_V have about the same amount of parameters as they have the same number and size of dense layers. Figs. 3(d) and 3(e) illustrate both the VAE and CVAE generators where the VAE utilizes dense layers in the encoder/decoder blocks and CVAE utilize convolutional layers in the encoder/decoder blocks.

The loss calculation is dependent upon the choice the the reranking function, as discussed in Section II-B. The simplest form of the reranking function is a one hot encoding of the desired class label. However, the one hot encoding reranking fuction is not the best choice when the objective is to minimize perturbations per class label. Other options for reranking functions require the use of the class probability vector. This class probability vector is not available when performing black-box attacks or when attacking most traditional models. We utilize the transferability property and knowledge distillation to train a student network s, where the class probability vector is not available. The student neural network s is trained to mimic the classification output for the targeted model f. The student model of the proposed architectures (MGATN, MGATN_L, MGATN_V, and MGATN_{CV}) uses the same knowledge distillation loss function described in Section II-E. The knowledge learned from the student model is then used in substitute of the unknown information, such as the class probability vector.

The MGATN framework aims to optimize the adversarial samples by balancing the loss on the input space and the loss of the prediction output. This loss function is described in (5). When training with VAE and CVAE generators, the KL divergence must be factored into the loss equation. The following loss function is optimized:

$$L = \beta \times [L_x(g_f(\mathbf{X}_i), \mathbf{X}_i)] - \beta_0 \times L_{KL} + L_y(f(g_f(\mathbf{X}_i)), f(\mathbf{X}_i)) \tag{14}$$

$$L_{KL} = \sigma^2 + \mu^2 - \log(\sigma) - 1 \tag{15}$$

where $\mathbf{X}$ is a multivariate time series, L_x is the loss function in the input space, L_y is the loss function on the output space of f, L_{KL} is the loss from the encoder, β is a weight to balance L_X, β_0 is a weight to balance L_{KL} (set to 0.005), σ is the standard deviation output of the VAE encoder, and μ is the mean output of the VAE encoder.

A key difference between MGATN, MGATN_L and the proposed architectures, MGATN_V and MGATN_{CV}, is the addition of a KL component in the loss function. Intuitively, the KL component measures the divergence between two distributions. MGATN_L attempts to determine the $P(X|z)$, where $P(X)$ is the probability distribution of the input time series data, and $P(z)$ is the distribution of the latent variable. The addition of the KL component, allows MGATN_V and MGATN_{CV} to estimate $P(z|X)$ as close as possible. In addition, the KL component is used as a regularization term which separates MGATN_V and MGATN_{CV} from standard MGATN and MGATN_L. The KL component in the loss function forces the model to minimize the cross-entropy between the aggregated posterior, which improves sample quality and latent features. This regularization term in addition to the reconstruction error promotes the quality of the encoder block to learn underlying patterns. While this has not been applied in the field of time series adversaries, the improvements of VAEs over standard autoencoders are well reseachered in the field of computer vision [58], [64]. The pioneers of VAE, Kingma and Welling, provide further theoretical reasoning on the added benefits of VAE [61].

IV. Experiments

A. Multivariate Time Series Benchmarks

All methodologies presented in this work are tested on 30 multivariate time series benchmark datasets. These benchmarks are compiled and provided by University of East Anglia (UEA) and the University of California, Riverside (UCR) in the Multivariate Time Series Classification Archive [46]. Table I provides information about the test benchmarks. These benchmark datasets are from a variety of fields, including medical care, speech recognition and motion recognition.

B. Black-Box and White-Box Restrictions

The experiments to evaluate all versions of MGATN follow specific restrictions for black-box and white-box attacks. All versions of MGATN require the use of gradient information to attack the target classifier. In this study, black-box attacks are limited to the discrete class label of the model output, and not the class probability vector that is output from a neural network with a softmax output.

The available multivariate time series archive [46] provides researchers with a training and testing set for model development. In this study, we split the testing set of the available data into two sets, D_{eval} and D_{test}. This split is a class balanced equally sized split of the available testing data. The training data is used to develop the attacked models f. MGATN is then trained on the D_{eval} dataset. We evaluate MGATN on the model development set D_{eval}, and the unseen testing set D_{test}. We further restrict black-box attacks to generate adversaries using unlabeled training data. This restriction utilizes the attacked classifier to determine the labels of training data prior to attacking the classifier.

C. Experimental Setup

This work explores black-box and white-box attacks with different generators on traditional and neural network time series classifiers. We compare the different generator functions to present how the proposed generator functions perform in comparison to established generator functions. Details for these classifiers were explained in Section II-A. Based on the restrictions of the attacks and traditional time series classifiers, we utilize a student model for all attacks

TABLE I
DATASET DESCRIPTION FOR UEA AND UCR MULTIVARIATE TIME SERIES BENCHMARKS

Dataset	Train cases	Test cases	Dimensions	Length	Classes
ArticularyWordRecognition	275	300	9	144	25
AtrialFibrillation	15	15	2	640	4
BasicMotions	40	40	6	100	4
CharacterTrajectories	1422	1436	3	182	20
Cricket	108	72	6	1197	12
DuckDuckGeese	50	50	1345	270	5
EigenWorms	131	128	6	17 984	5
Epilepsy	137	138	3	206	4
EthanolConcentration	261	263	3	1751	4
ERing	30	30	4	65	6
FaceDetection	5890	3524	144	62	2
FingerMovements	316	100	28	50	2
HandMovementDirection	320	147	10	400	4
Handwriting	150	850	3	152	26
Heartbeat	204	205	61	405	2
InsectWingBeat	25 000	25 000	200	30	10
JapaneseVowels	270	370	12	29	9
Libras	180	180	2	45	15
LSST	2459	2466	6	36	14
MotorImagery	278	100	64	3000	2
NATOPS	180	180	24	51	6
PEMS-SF	267	173	963	144	7
PenDigits	7494	3498	2	8	10
Phoneme	3315	3353	11	217	39
RacketSports	151	152	6	30	4
SelfRegulationSCP1	268	293	6	896	2
SelfRegulationSCP2	200	180	7	1152	2
SpokenArabicDigits	6599	2199	13	93	10
StandWalkJump	12	15	4	2500	3
UWaveGestureLibrary	120	320	3	315	8

except the white-box attack on the Multi-FCN classifier. White-box attacks on traditional classifiers, such as 1-NN DTW, do not provide the required class probability information for MGATN to be utilized. Additionally, black-box attacks for both traditional and neural network do not allow the attacker access to the internal information of the classifier, such as the neural network weights. In these cases, the student model s is utilized to train MGATN. The output of MGATN is then used to test if the teacher model f is vulnerable to the adversarial sample. As discussed in Section IV-B, the initial testing set is split into two equally sized sets for evaluation, D_{eval} and D_{test}. When evaluating the different variations of MGATN, we compute the fraction of successful adversaries on the targeted model f generated on the evaluation set D_{eval}. For a multivariate time series to be a successful adversarial example, we first input the time series into the classification model to see if the classifier results in the correct class label. If the classifier is able to correctly determine the class label of the time series, then we check to see if the corresponding adversarial sample of this time series is incorrectly labeled by the targeted classifier. This definition ensures that only correctly labeled samples can result in a possible adversarial sample. To evaluate our models ability to attack multivariate time series classification algorithms, we utilize both a traditional distance based classifier and a neural network classifier. The attacked time series classifiers are the Multivariate 1-Nearest Neighbor Dynamic Time Warping and the Multivariate Fully Convolutional Network. We evaluate based on the number of adversarial samples generated and the amount of perturbation added. The perturbation is measured by comparing the mean squared error of the original input sample compared to the output adversarial sample. The objective is to minimize the amount of perturbation and maximize the fraction of the training samples that result in successful adversaries. All experiments use a reranking weight α that is set to 1.5. The target class is specified to class 0. For benchmarks that have non-numeric class labels, the labels are encoded and the corresponding class label can be found in our

codebase. The reconstruction weight parameter β is selected by grid searching over a set of possibilities, such that $\beta \in \{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00001\}$. This work performs several experiments to show how multivariate time series classifiers can be attacked (comparing fraction of successful adversaries in Section V-A), how well the proposed architectures can generalize onto unseen multivariate time series data (generalization in Section V-B) and how well can these attacks be defended by retraining on these adversaries (defense in Section V-C).

D. Modeling Parameters

All tested attacks utilize a student network to mimic the target network except white-box attacks that target the Multi-FCN classifier. ATNs can directly utilize the gradient information of the Multi-FCN model that is available to the attacker in the white-box case. White-box attacks on Multi-FCNs are evaluated directly on the target classification model.

All student models utilize a simple LeNet-5 architecture that is modified for a multivariate input [65]. The LeNet-5 architecture is selected because it is one of the earliest and simplest convolutional neural networks, which has shown to work effectively on attacking univariate time series classifiers [40]. The architecture is a convolution neural network with two 2-dimensional convolutional layers with filter of size 6 and 16, kernels of size 5 and 5, ReLU activation functions, and valid padding. Each convolutional layer is passed to a 2-D Max Pooling layer. This is then passed to a flattening layer. The network then has two dense layers of 120 and 84 with tanh activation functions. Finally the network ends with a dense softmax layer consisting of the desired number of classes.

We use the multivariate version of 1-NN DTW to evaluate adversarial attacks on traditional time series classifiers. While this classifier has proven to be an effective method for both univariate and multivariate time series, the distance based restriction requires some modifications to utilize probabilistic outputs. This is not an issue when conducting black-box attacks on the classifier. White-box attacks have access to the output class probability distribution for each sample. Karim *et al*. [40] introduced a method of generating a class probability distribution that can generate the same discrete class result as the original classification. This method can be extended to accept a set of distance matrices as inputs, as opposed to a single distance matrix.

Our experiments evaluate the use of four different methods networks for adversary generation. The first is a simple fully connected network, which passes the original and gradient information to two dense layers with ReLU activation functions and the output is a gradient with matching input shape and a linear activation function. The second is an extended simple generator with two additional dense layers. The third network is a variational autoencoder with an intermediate dimension of 32 and latent dimension of 16. The final network is a convolutional variational autoencoder which alters the original dense layers of the VAE with convolutional layers. This network has the same intermediate dimension of 32 and latent dimension of 16. More detailed architecture parameters can be explored in our codebase.

V. Results

A. Fraction of Successful Adversaries

Fig. 4 illustrates the fraction of successful adversaries generated during black-box and white-box attacks on the 30 tested multivariate time series benchmarks. The fraction of successful adversaries is defined as the number of adversaries captured by the attack divided by the number of possible adversaries that could be generated. As an additional requirement, we select our models based on a mean squared error (MSE) limit of 0.1 between the original and adversarial time series. This requirement is used to determine the model based on the stated range of beta values β. Fig. 4 also illustrate the differences in results for the tested generators. These include MGATN, MGATN$_L$, MGATN$_V$, and MGATN$_{CV}$. The target class for all experiments is set to class 0, where string labels are encoded and these class encoding can be found in our codebase for reproducibility. The detailed results for all experiments can be found in Appendix.

There are a total of 120 experiments for each generator function, 30 datasets on 4 different attack-target combinations. We compare these experiments across generators by evaluating the number of wins a method has across all experiments. A win is defined as one method having a greater fraction of successful adversaries than any other method, where ties are not collected. The number of wins for MGATN, MGATN$_L$, MGATN$_V$, and MGATN$_{CV}$ are 16, 10, 38, and 27, respectively. MGATN$_V$ has the most wins across all the experiments where 27 of the 38 wins occur in black-box attacks. However, MGATN$_{CV}$ has a high number of white-box wins (19 wins) compared to the other methods. When looking at the target model instead of attack, we see that MGATN$_V$ has the most wins on Multivariate 1-NN DTW (14 wins) and has the most wins on Multi-FCN (24 wins). We postulate the AutoEncoder component of MGATN$_V$ and MGATN$_{CV}$ to be the reason why the performance is increasing and not the size of the network. This is because MGATN$_L$ and MGATN$_V$ have about the same size and parameters, yet MGATN$_V$ outperforms MGATN$_L$, significantly. These findings demonstrate the significant improvement achieved with the modification of the generator functions. Fig. 5 illustrates adversarial samples with all attacks.

We use a Wilcoxon sign-rank test (WSRT) to compare the fraction of successful adversaries generated by white-box and black-box attacks on Multivariate 1-NN DTW and Multi-FCN classifiers. WSRT is a nonparametric statistical hypothesis test used to compare two related samples when the distribution of the two samples means cannot be assumed to be normally distributed. Table II shows the p-values for the MGATN variations compared to the fast gradient signed method (FGSM), a good baseline for generating adversaries [31]. This table compares MGATN variants with FGSM where significant improvements (at a p-value of 0.05) are green and experiments with no significant improvements are red. Our results indicate that, with the exception of white box attacks on 1-NN DTW classifiers, the MGATN variants all achieve

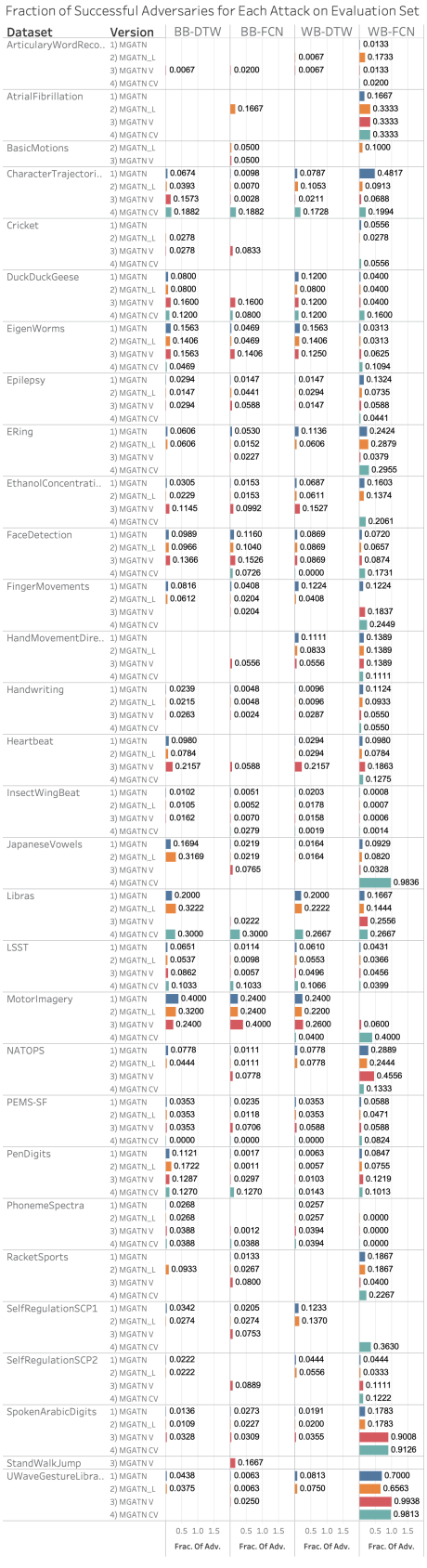

Fig. 4. Black-box and white-box attacks on FCN and 1-NN DTW classifiers that are tested on D_{eval}. Blank bars mean that the adversarial attacks did not result in any adversaries for a specific attack and dataset.

higher fractions of successful adversaries then the FGSM method. It should be noted that MGATN performs similarly on white box attacks with 1-NN DTW, and never statistically under performs.

Table III summarizes the results of MGATN_V and MGATN_{CV} compared to the original MGATN and MGATN_L. Table cells in green show a significant improvement (at a p-value of 0.05) over the original MGATN. These results show MGATN_V is superior to MGATN and MGATN_L when applied on both tested black-box attacks. Even with approximately the same number of parameters, MGATN_V statistically outperforms MGATN_L for black-box attacks. However, MGATN_V does not perform the original MGATN or MGATN_L on white-box attacks. This difference in adversarial results comes from the VAEs objective of regularizing the latent space. MGATN_{CV} outperforms MGATN and MGATN_L when applying white-box attack on the Multi-FCN classifier. The MGATN_{CV} performs the best for white-box on the neural network classifier, which we postulate is because the convolutional component provides superior encoding for this attack. This indicates the importance of the AutoEncoders in the generator and the significant improvement in the resultant fraction of adversaries achieved.

In order to test the distribution of the generated adversary a Cramer test is used to compare generated adversaries with the original multivariate time series. The Cramer test is a non-parametric two sampled test of the underlying distributions between multivariate sets of data [66]. The Cramer test has a null hypothesis of the two samples belonging to the same distribution and an alternative hypothesis that they are drawn from different distributions. All generated adversaries have a p-value below 0.05. These results prove that all adversaries are drawn from the same distribution as their original multivariate time series.

B. Generalization

In this subsection we evaluate the trained MGATN models on the unseen testing data, D_{test}. This evaluation is important when the situation does not allow time for model retraining. This is the case when sensor data only has the time for a forward pass through the network. This evaluation is not common when generating adversaries. However, time series applications often require real-time analysis of collected data. For this reason, it is beneficial that MGATN is able to generate adversarial samples without the need for retraining. Fig. 6 illustrates the fraction of successful adversaries for all testing sets and on all attacks. These results show a similar fraction for each type of attack. The white-box attack on the Multi-FCN classifier obtains the highest fraction of successful adversaries across most datasets. Additionally, we see that the black-box attack on Multi-FCN results in the lowest fraction of successful adversaries. The black-box limitation along with the increased complexity of neural networks compared to distance based models makes it difficult to generate adversaries on black-box attacks on Multi-FCNs. This generalization shows the potential to generate adversaries based on pretrained models. This allows for MGATN to be applied on edge devices with limited computational resources and still be able to generate successful adversaries.

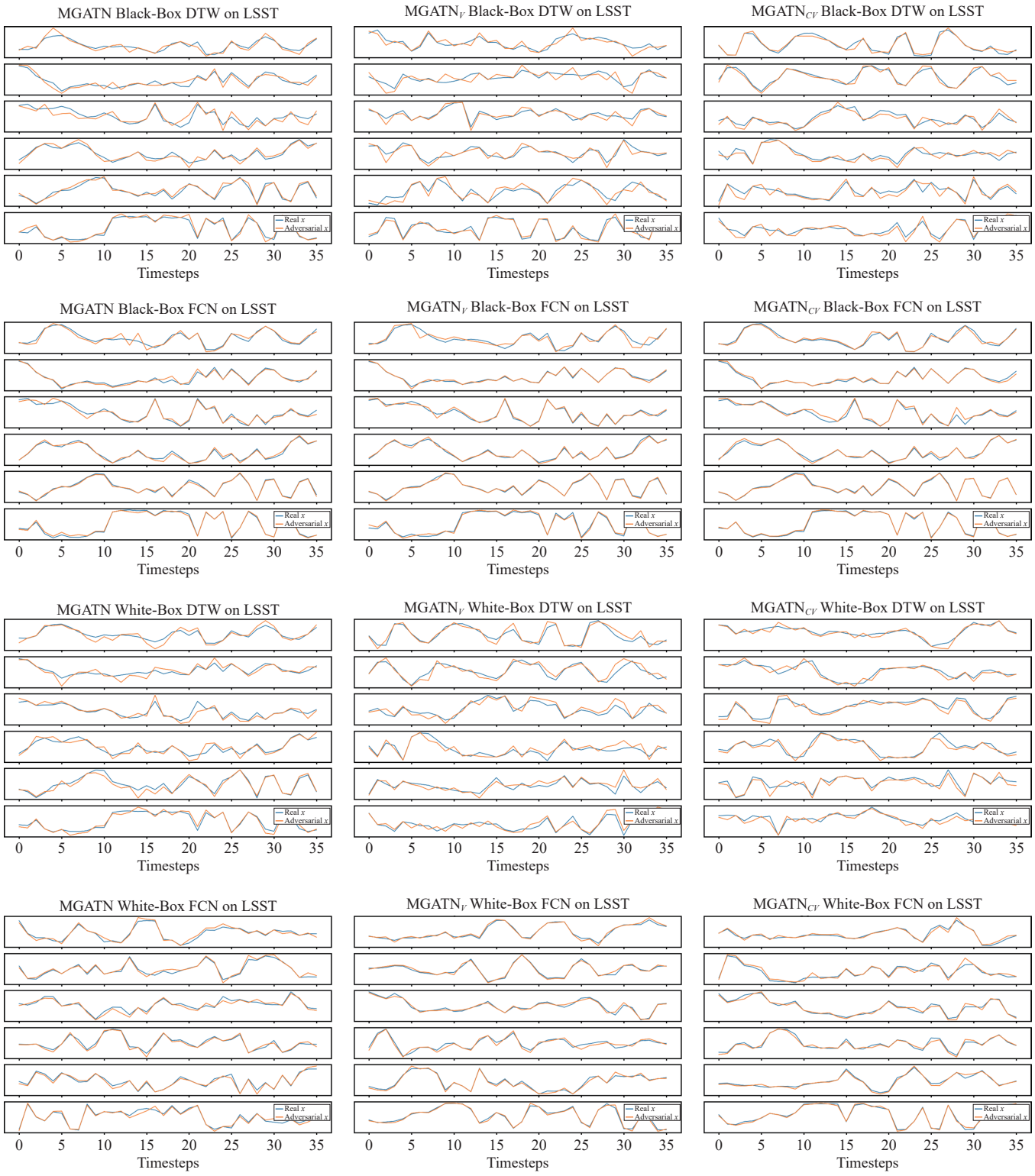

Fig. 5. Sample Adversarial Samples on the LSST dataset.

C. Defense

The ability to defend against an adversarial attack is an important post-evaluation that utilizes the findings of MGATN. In this work we explore a simple defense process that utilizes the pretrained MGATN models. Given a trained MGATN, we output the successful adversarial samples for an attack. These adversarial samples are appended onto our original training data. The classifier and MGATN are then retrained using this new training set. Based on this defense process, we evaluate MGATN by analyzing the change in testing accuracy and the fraction of successful adversaries. Table IV shows a Wilcoxon Sign-Rank Test for a comparison of testing accuracy of developed models. The testing set D_{test} is defined in Section IV-B. Table cells that are green show a significant improvement in testing accuracy (at a p-value of 0.05). This analysis evaluates whether the use of adversarial

TABLE II
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} OF OUR METHODS COMPARED TO FGSM (P-VALUE SHOWN)

	MGATN	MGATN_L	MGATN_V	MGATN_{CV}
BB w/ 1-NN DTW	1.24E–02	1.11E–02	3.75E–03	5.41E–03
WB w/ 1-NN DTW	9.34E–02	8.79E–02	6.78E–02	6.43E–02
BB w/ Multi-FCN	3.87E–03	3.12E–03	8.64E–05	7.39E–03
WB w/ Multi-FCN	5.22E–03	4.23E–03	7.38E–03	3.21E–05

TABLE III
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} COMPARISON OF MGATN GENERATORS (P-VALUE SHOWN)

	MGATN_V	MGATN_{CV}
MGATN on BB w/ 1-NN DTW	4.46E–02	7.22E–02
MGATN on WB w/ 1-NN DTW	5.42E–02	5.13E–02
MGATN on BB w/ Multi-FCN	7.34E–04	9.59E–02
MGATN on WB w/ Multi-FCN	8.79E–02	1.08E–02
MGATN_L on BB w/ 1-NN DTW	4.52E–02	7.24E–02
MGATN_L on WB w/ 1-NN DTW	5.62E–02	5.52E–02
MGATN_L on BB w/ Multi-FCN	8.42E–04	9.72E–02
MGATN_L on WB w/ Multi-FCN	8.12E–02	9.78E–03

sample in addition to training data results in significantly higher testing accuracy. The results show that only white-box samples from Multi-FCN generated from MGATN_V and MGATN_{CV} results in a significant increase in model accuracy. The remainder of the experiments show that the adversarial samples do not result in an improved testing accuracy when the models are retrained. Table V shows a WSRT for the comparison of the fraction of successful adversaries generated. These results show that there is a significant decrease in the fraction of adversaries generated when the model is retrained with adversaries in the training data (at a p-value of 0.05). The one exception is the MGATN_{CV}, where the fraction of adversaries is statistically large compared to the previous tests. This shows that both distance based and neural network classifiers become more robust when retrained with new samples.

D. Latent Space

The MGATN_V and MGATN_{CV} methods make use of variational autoencoders to generate adversarial samples. Variational autoencoders provides a probabilistic manner for describing an observation in latent space. VAEs formulate the encoder to describe a probability distribution for each latent attribute. To understand the implications of a variational autoencoder, we visualize the latent space. Fig. 7 illustrates the latent space of evaluation samples on the CharacterTrajectories dataset generated by MGATN_V and MGATN_{CV} using a t-distributed stochastic neighbor embedding (TSNE) data reduction. TSNE is a statistical method for visualizing high-dimensional data in a two or three dimensional space using a stochastic neighbor embedding [67]. This figure shows that MGATN_{CV} method is able to learn clear differences between classes. The MGATN_V method results non-separable latent spaces using the TSNE

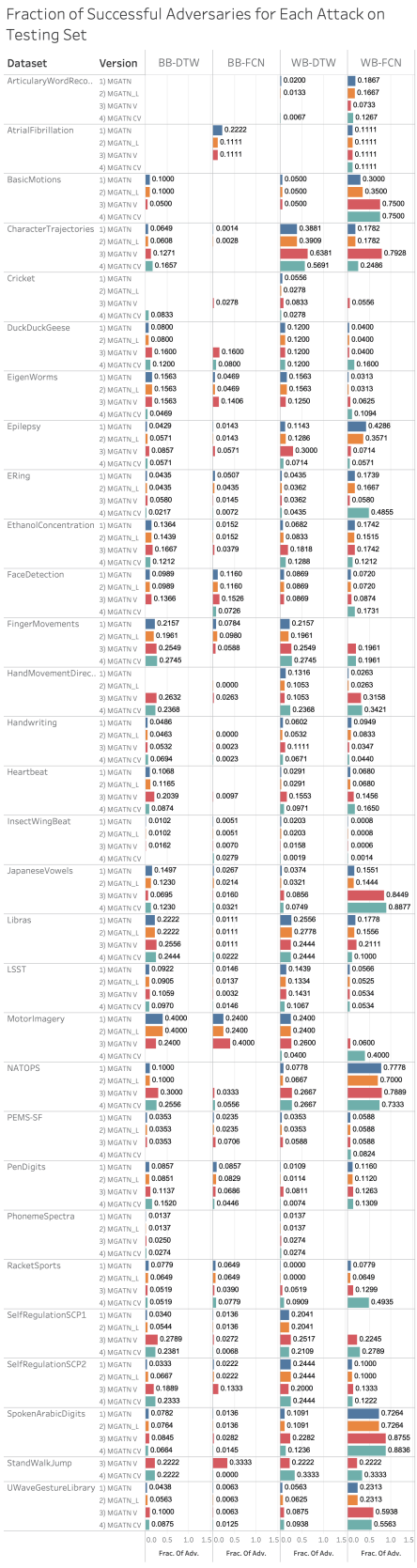

Fig. 6. Black-box and white-box attacks on FCN and 1-NN DTW classifiers that are tested on D_{test}. Blank bars mean that the adversarial attacks did not result in any adversaries for a specific attack and dataset.

dimensionality reduction. However, this does not mean

TABLE IV
WSRT COMPARING TESTING ACCURACY OF MODELS DEVELOPED ON ORIGINALTRAINING DATA VS TRAINING DATA WITH ADVERSARIAL SAMPLES (P-VALUE SHOWN)

	MGATN	$MGATN_L$	$MGATN_V$	$MGATN_{CV}$
BB DTW	5.88E–01	5.76E–01	4.36E–01	6.28E–01
BB FCN	7.13E–01	6.92E–01	9.16E–01	1
WB DTW	6.14E–01	6.12E–01	2.32E–01	4.18E–01
WB FCN	1.41E–01	1.32E–01	1.13E–02	5.34E–03

TABLE V
WSRT COMPARING FRACTION OF SUCCESSFUL ADVERSARIES FOR MODELS DEVELOPED ON ORIGINAL TRAINING DATA VS TRAINING DATA WITH ADVERSARIAL SAMPLES (P-VALUE SHOWN)

	MGATN	$MGATN_L$	$MGATN_V$	$MGATN_{CV}$
BB DTW	2.32E–02	2.22E–02	2.26E–02	1.55E–01
BB FCN	3.29E–02	3.20E–02	2.73E–02	3.74E–02
WB DTW	3.45E–03	3.43E–03	5.43E–04	1.28E–02
WB FCN	5.01E–05	5.32E–05	6.36E–06	2.32E–06

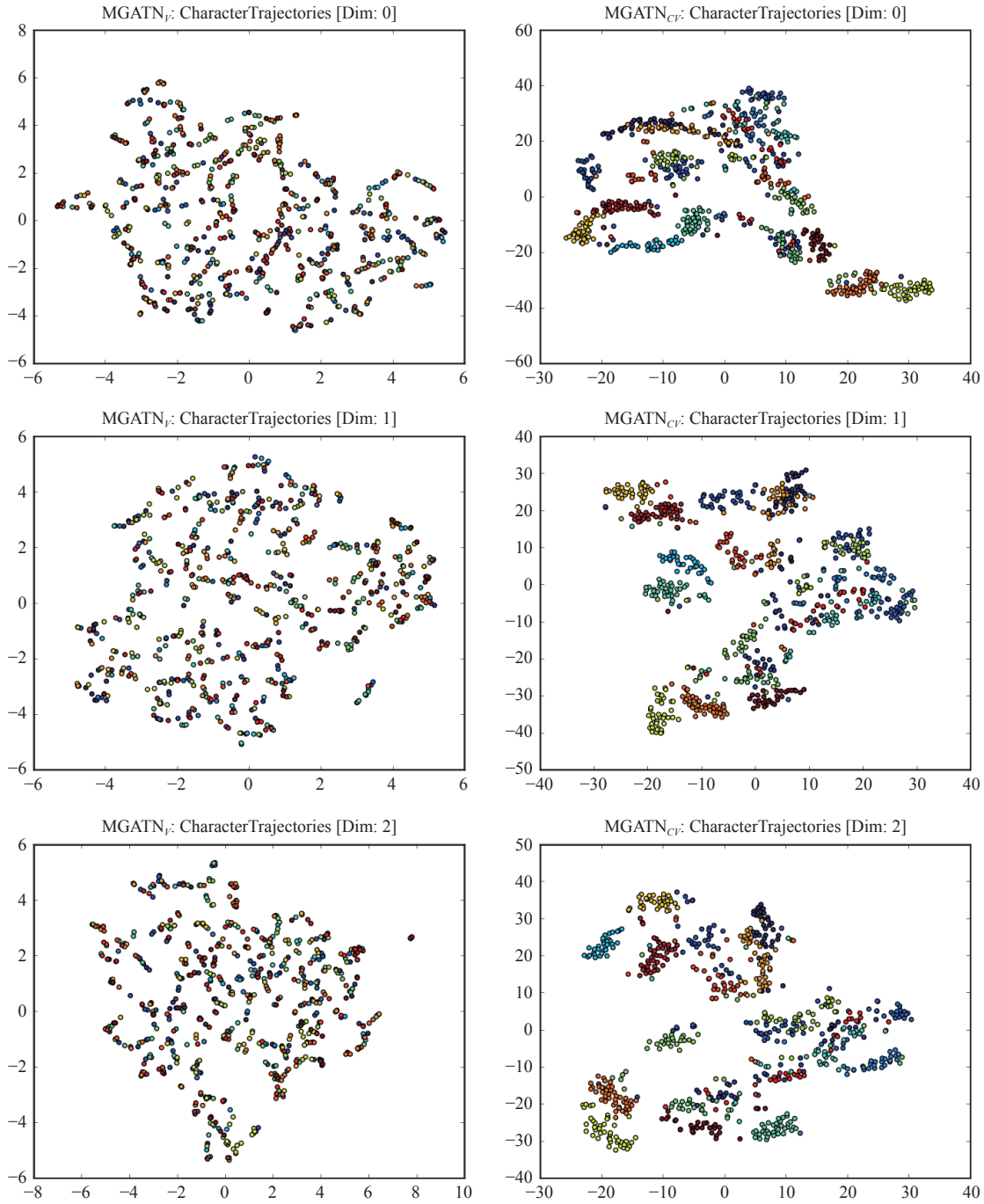

Fig. 7. Example illustrations of Latent Dimensions using TSNE reduction.

$MGATN_V$ has no linearly separable classes on different different or reduction techniques. Further research is required to understand the latent spaces of $MGATN_V$ Models that generate a clearly defined latent space can be used to form a generative model capable of creating new data similar to what was observed during training. New data generated from an interpretable latent space can be used to retrain classifiers to increase accuracy, further improve defense against adversarial attacks, and many other data mining applications. Further, the embeddings from the latent space can also be utilized for other classification tasks and anomaly detection.

VI. Conclusion

This work extends the GATN by modifying the generated function to significantly improve the generated adversaries on multivariate time series. We evaluate the different generation methods on 30 multivariate time series datasets. These evaluations test both black-box and white-box attacks on multivariate 1-NN DTW and Multi-FCN classifiers. Our results show that the most vulnerable model is the Multi-FCN attacked with white-box information. We further prove the generated adversaries are from the same distribution as the original series using a Cramer test. Utilizing an unseen testing set, we see that our MGATN models are able to generate adversaries on data without the need for retraining. A simple defense procedure shows that the use of generating adversaries when retraining our models makes them less vulnerable to future attacks while maintaining the same level of testing accuracy. Finally, we see that the latent space modeled by $MGATN_{CV}$ results in a clear class separation that can be used for future data generation. Future research in this area should explore the development of targeted adversarial attacks that misclassify input to a specific class. Finally, the developed latent space information can be exploited to better understand the underlying patterns of the time series classes.

Acknowledgments

We acknowledge Somshubra Majumdar for his assistance and insightful comments that laid the foundation to the research work. Further, we would like to thank all the researchers that spent their time and effort to create the data we used.

Appendix
Detailed Results

TABLE VI
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} FOR THE BLACK-BOX ATTACK ON MULTI 1-NN DTW

Dataset	MGATN	$MGATN_L$	$MGATN_V$	$MGATN_{CV}$
ArticularyWord-Recognition	0.0000	0.0000	0.0067	0.0000
AtrialFibrillation	0.0000	0.0000	0.0000	0.0000
BasicMotions	0.0000	0.0000	0.0000	0.0000
CharacterTrajectories	0.0674	0.0393	0.1573	0.1882
Cricket	0.0000	0.0278	0.0278	0.0000
DuckDuckGeese	0.0800	0.0800	0.1600	0.1200
EigenWorms	0.1563	0.1406	0.1563	0.0469

TABLE VI
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} FOR THE BLACK-BOX ATTACK ON MULTI 1-NN DTW (CONTINUED)

Dataset	MGATN	$MGATN_L$	$MGATN_V$	$MGATN_{CV}$
Epilepsy	0.0294	0.0147	0.0294	0.0000
ERing	0.0606	0.0606	0.0000	0.0000
EthanolConcentration	0.0305	0.0229	0.1145	0.0000
FaceDetection	0.0989	0.0966	0.1366	0.0000
FingerMovements	0.0816	0.0612	0.0000	0.0000
HandMovement-Direction	0.0000	0.0000	0.0000	0.0000
Handwriting	0.0239	0.0215	0.0263	0.0000
Heartbeat	0.0980	0.0784	0.2157	0.0000
InsectWingBeat	0.0102	0.0105	0.0162	0.0000
JapaneseVowels	0.1694	0.3169	0.0000	0.0000
Libras	0.2000	0.3222	0.0000	0.3000
LSST	0.0651	0.0537	0.0862	0.1033
MotorImagery	0.4000	0.3200	0.2400	0.0000
NATOPS	0.0778	0.0444	0.0000	0.0000
PEMSSF	0.0353	0.0353	0.0353	0.0000
PenDigits	0.1121	0.1722	0.1287	0.1270
PhonemeSpectra	0.0268	0.0268	0.0388	0.0388
RacketSports	0.0000	0.0933	0.0000	0.0000
SelfRegulationSCP1	0.0342	0.0274	0.0000	0.0000
SelfRegulationSCP2	0.0222	0.0222	0.0000	0.0000
SpokenArabicDigits	0.0136	0.0109	0.0328	0.0000
StandWalkJump	0.0000	0.0000	0.0000	0.0000
UWaveGestureLibrary	0.0438	0.0375	0.0000	0.0000

TABLE VII
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} FOR THE BLACK-BOX ATTACK ON MULTI-FCN

Dataset	MGATN	$MGATN_L$	$MGATN_V$	$MGATN_{CV}$
ArticularyWord-Recognition	0.0000	0.0000	0.0200	0.0000
AtrialFibrillation	0.0000	0.1667	0.0000	0.0000
BasicMotions	0.0000	0.0500	0.0500	0.0000
CharacterTrajectories	0.0098	0.0070	0.0028	0.1882
Cricket	0.0000	0.0000	0.0833	0.0000
DuckDuckGeese	0.0000	0.0000	0.1600	0.0800
EigenWorms	0.0469	0.0469	0.1406	0.0000
Epilepsy	0.0147	0.0441	0.0588	0.0000
ERing	0.0530	0.0152	0.0227	0.0000
EthanolConcentration	0.0153	0.0153	0.0992	0.0000
FaceDetection	0.1160	0.1040	0.1526	0.0726
FingerMovements	0.0408	0.0204	0.0204	0.0000
HandMovementDirection	0.0000	0.0000	0.0556	0.0000
Handwriting	0.0048	0.0048	0.0024	0.0000
Heartbeat	0.0000	0.0000	0.0588	0.0000
InsectWingBeat	0.0051	0.0052	0.0070	0.0279
JapaneseVowels	0.0219	0.0219	0.0765	0.0000
Libras	0.0000	0.0000	0.0222	0.3000
LSST	0.0114	0.0098	0.0057	0.1033

TABLE VII
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} FOR THE BLACK-BOX ATTACK ON MULTI-FCN (CONTINUED)

Dataset	MGATN	MGATN_L	MGATN_V	MGATN_{CV}
MotorImagery	0.2400	0.2400	0.4000	0.0000
NATOPS	0.0111	0.0111	0.0778	0.0000
PEMSSF	0.0235	0.0118	0.0706	0.0000
PenDigits	0.0017	0.0011	0.0297	0.1270
PhonemeSpectra	0.0000	0.0000	0.0012	0.0388
RacketSports	0.0133	0.0267	0.0800	0.0000
SelfRegulationSCP1	0.0205	0.0274	0.0753	0.0000
SelfRegulationSCP2	0.0000	0.0000	0.0889	0.0000
SpokenArabicDigits	0.0273	0.0227	0.0309	0.0000
StandWalkJump	0.0000	0.0000	0.1667	0.0000
UWaveGestureLibrary	0.0063	0.0063	0.0250	0.0000

TABLE VIII
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} FOR THE WHITE-BOX ATTACK ON MULTI 1-NN DTW

Dataset	MGATN	MGATN_L	MGATN_V	MGATN_{CV}
ArticularyWord-Recognition	0.0000	0.0067	0.0067	0.0000
AtrialFibrillation	0.0000	0.0000	0.0000	0.0000
BasicMotions	0.0000	0.0000	0.0000	0.0000
CharacterTrajectories	0.0787	0.1053	0.0211	0.1728
Cricket	0.0000	0.0000	0.0000	0.0000
DuckDuckGeese	0.1200	0.0800	0.1200	0.1200
EigenWorms	0.1563	0.1406	0.1250	0.0000
Epilepsy	0.0147	0.0294	0.0147	0.0000
ERing	0.1136	0.0606	0.0000	0.0000
EthanolConcentration	0.0687	0.0611	0.1527	0.0000
FaceDetection	0.0869	0.0869	0.0869	0.0000
FingerMovements	0.1224	0.0408	0.0000	0.0000
HandMovementDirection	0.1111	0.0833	0.0556	0.0000
Handwriting	0.0096	0.0096	0.0287	0.0000
Heartbeat	0.0294	0.0294	0.2157	0.0000
InsectWingBeat	0.0203	0.0178	0.0158	0.0019
JapaneseVowels	0.0164	0.0164	0.0000	0.0000
Libras	0.2000	0.2222	0.0000	0.2667
LSST	0.0610	0.0553	0.0496	0.1066
MotorImagery	0.2400	0.2200	0.2600	0.0400
NATOPS	0.0778	0.0778	0.0000	0.0000
PEMSSF	0.0353	0.0353	0.0588	0.0000
PenDigits	0.0063	0.0057	0.0103	0.0143
PhonemeSpectra	0.0257	0.0257	0.0394	0.0394
RacketSports	0.0000	0.0000	0.0000	0.0000
SelfRegulationSCP1	0.1233	0.1370	0.0000	0.0000
SelfRegulationSCP2	0.0444	0.0556	0.0000	0.0000
SpokenArabicDigits	0.0191	0.0200	0.0355	0.0000
StandWalkJump	0.0000	0.0000	0.0000	0.0000
UWaveGestureLibrary	0.0813	0.0750	0.0000	0.0000

TABLE IX
FRACTION OF SUCCESSFUL ADVERSARIES ON D_{EVAL} FOR THE WHITE-BOX ATTACK ON MULTI-FCN

Dataset	MGATN	MGATN_L	MGATN_V	MGATN_{CV}
ArticularyWord-Recognition	0.0133	0.1733	0.0133	0.0200
AtrialFibrillation	0.1667	0.3333	0.3333	0.3333
BasicMotions	0.0000	0.1000	0.0000	0.0000
CharacterTrajectories	0.4817	0.0913	0.0688	0.1994
Cricket	0.0556	0.0278	0.0000	0.0556
DuckDuckGeese	0.0400	0.0400	0.0400	0.1600
EigenWorms	0.0313	0.0313	0.0625	0.1094
Epilepsy	0.1324	0.0735	0.0588	0.0441
ERing	0.2424	0.2879	0.0379	0.2955
EthanolConcentration	0.1603	0.1374	0.0000	0.2061
FaceDetection	0.0720	0.0657	0.0874	0.1731
FingerMovements	0.1224	0.0000	0.1837	0.2449
HandMovementDirection	0.1389	0.1389	0.1389	0.1111
Handwriting	0.1124	0.0933	0.0550	0.0550
Heartbeat	0.0980	0.0784	0.1863	0.1275
InsectWingBeat	0.0008	0.0007	0.0006	0.0014
JapaneseVowels	0.0929	0.0820	0.0328	0.9836
Libras	0.1667	0.1444	0.2556	0.2667
LSST	0.0431	0.0366	0.0456	0.0399
MotorImagery	0.0000	0.0000	0.0600	0.4000
NATOPS	0.2889	0.2444	0.4556	0.1333
PEMSSF	0.0588	0.0471	0.0588	0.0824
PenDigits	0.0847	0.0755	0.1219	0.1013
PhonemeSpectra	0.0000	0.0000	0.0000	0.0000
RacketSports	0.1867	0.1867	0.0400	0.2267
SelfRegulationSCP1	0.0000	0.0000	0.0000	0.3630
SelfRegulationSCP2	0.0444	0.0333	0.1111	0.1222
SpokenArabicDigits	0.1783	0.1783	0.9008	0.9126
StandWalkJump	0.0000	0.0000	0.0000	0.0000
UWaveGestureLibrary	0.7000	0.6563	0.9938	0.9813

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[2] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[3] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. 12th Annu. Conf. Int. Speech Communication Association*, 2011.

[4] X. Zhang, J. Zhao, and Y. LeCun, " Character-level convolutional networks for text classification," *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[5] M. Pishgar, F. Karim, S. Majumdar, and H. Darabi, "Pathological voice classification using mel-cepstrum vectors and support vector machine," arXiv preprint arXiv: 1812.07729, 2018.

[6] S. Harford, H. Darabi, M. Del Rios, S. Majumdar, F. Karim, T. V. Hoek, K. Erwin, and D. P. Watson, "A machine learning based model for out of hospital cardiac arrest outcome classification and sensitivity analysis," *Resuscitation*, vol. 138, pp. 134–140, 2019.

[7] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The UCR time series archive," *IEEE/CAA J. Autom. Sinica* , vol. 6, no. 6, pp. 1293–1305, 2019.

[8] H. Darabi, G. Ifrim, P. Schafer, and D. F. Silva, " Guest editorial for special issue on time series classification," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 6, pp. 1291–1292, 2019.

[9] P. M. Papadopoulos, V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, " Scalable distributed sensor fault diagnosis for smart buildings," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 3, pp. 638–655, 2020.

[10] K. Sirisambhand and C. A. Ratanamahatana, " A dimensionality reduction technique for time series classification using additive representation," in *Proc. 3rd Int. Congr. Information and Communication Technology*. Springer, 2019, pp. 717–724.

[11] 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 Trans. Systems*, *Man*, *and Cybernetics*: *Systems*, vol. 47, no. 10, pp. 2688–2703, 2017.

[12] 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*, vol. 57, no. 2, pp. 359–388, 2018.

[13] A. Sharabiani, A. Sharabiani, and H. Darabi, " A novel bayesian and chain rule model on symbolic representation for time series classification," in *Proc. IEEE Int. Conf. Automation Science and Engineering*, 2016, pp. 1014–1019.

[14] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proc. 23rd ACM Int. Conf. Machine Learning*, 2006, pp. 1033–1040.

[15] F. Karim, H. Darabi, S. Harford, S. Chen, and A. Sharabiani, "A framework for accurate time series classification based on partial observation," in *Proc. IEEE 15th Int. Conf. Automation Science and Engineering*, 2019, pp. 634–639.

[16] F. Karim, S. Majumdar, H. Darabi, and S. Chen, " LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2017.

[17] F. Karim, S. Majumdar, and H. Darabi, " Insights into lstm fully convolutional networks for time series classification," *IEEE Access*, vol. 7, pp. 67718–67725, 2019.

[18] W. Pei, H. Dibeklioğlu, D. M. Tax, and L. van der Maaten, "Multivariate time-series classification using the hidden-unit logistic model," *IEEE Trans. Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 920–931, 2017.

[19] P. Schäfer and U. Leser, " Multivariate time series classification with WEASEL+MUSE," arXiv preprint arXiv: 1711.11343, 2017.

[20] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," *Neural Networks* , vol. 116, pp. 237–245, 2019.

[21] T. Lintonen and T. Raty, "Self-learning of multivariate time series using perceptually important points," *IEEE/CAA J. Autom. Sinica* , vol. 6, no. 6, pp. 1318–1331, 2019.

[22] H. Kang and S. Choi, " Bayesian common spatial patterns for multisubject EEG classification," *Neural Networks* , vol. 57, pp. 39–50, 2014.

[23] S. Doltsinis, M. Krestenitis, and Z. Doulgeri, " A machine learning framework for real-time identification of successful snap-fit assemblies," *IEEE Trans. Automation Science and Engineering*, 2019.

[24] Y. Fu, *Human Activity Recognition and Prediction*. Springer, 2016.

[25] S. Seto, W. Zhang, and Y. Zhou, "Multivariate time series classification using dynamic time warping template selection for human activity recognition," in *Proc. IEEE Symp. Series on Computational Intelligence*, 2015, pp. 1399–1406.

[26] Y. LeCun, Y. Bengio, and G. Hinton, " Deep learning," *Nature* , vol. 521, no. 7553, p. 436, 2015.

[27] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2018.

[28] Z. Wang, W. Yan, and T. Oates, " Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Networks*, 2017, pp. 1578–1585.

[29] S. Hashida and K. Tamura, " Multi-channel MHLF: LSTM-FCN using MACD-histogram with multi-channel input for time series classification," in *Proc. IEEE 11th Int. Workshop on Computational Intelligence and Applications*, 2019, pp. 67–72.

[30] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Joint European Conf. Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 387–402.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv: 1412.6572, 2014.

[32] J. Yoon, D. Jarrett, and M. van der Schaar, " Time-series generative adversarial networks," *Advances in Neural Information Processing Systems*, 2019, pp. 5508–5518.

[33] N. Yang, M. Zhou, B. Xia, X. Guo, and L. Qi, " Inversion based on a detached dual-channel domain method for StyleGAN2 embedding," *IEEE Signal Processing Letters*, vol. 28, pp. 553–557, 2021.

[34] H. Han, W. Ma, M. C. Zhou, Q. Guo, and A. Abusorrah, "A novel semi-supervised learning approach to pedestrian re-identification," *IEEE Internet of Things Journal*, 2020. DOI: 10.1109/JIOT.2020.3024287

[35] K. R. Mopuri, A. Ganeshan, and V. B. Radhakrishnan, "Generalizable data-free objective for crafting universal adversarial perturbations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2452–2465, 2019.

[36] K. Reddy Mopuri, U. Ojha, U. Garg, and R. Venkatesh Babu, "Nag: Network for adversary generation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 742–751.

[37] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, " Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[38] 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.

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

[40] F. Karim, S. Majumdar, and H. Darabi, " Adversarial attacks on time series," *IEEE Trans. Pattern Analysis and Machine Intelligence* , 2020. DOI: 10.1109/TPAMI.2020.2986319

[41] Q. Ma, W. Zhuang, S. Li, D. Huang, and G. W. Cottrell, "Adversarial dynamic shapelet networks." in *Proc. AAAI*, 2020, pp. 5069–5076.

[42] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Adversarial attacks on deep neural networks for time series classification," in *Proc. Int. Joint Conf. Neural Networks*, 2019, pp. 1–8.

[43] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access* , vol. 6, pp. 14410–14430, 2018.

[44] 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.

[45] 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.

[46] A. J. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. J. Keogh, " The UEA multivariate time series classification archive, 2018," *CoRR* , vol. abs/1811.00075, 2018. [Online]. Available: http://arxiv.org/abs/1811.00075

[47] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[48] P. Papapetrou, V. Athitsos, M. Potamias, G. Kollios, and D. Gunopulos, "Embedding-based subsequence matching in time-series databases," *ACM Trans. Database Systems* (*TODS*), vol. 36, no. 3, pp. 1–39, 2011.

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

[50] A. P. Ruiz, M. Flynn, and A. Bagnall, "Benchmarking multivariate time

series classification algorithms," arXiv preprint arXiv: 2007.13156, 2020.

[51] M. Shokoohi-Yekta, J. Wang, and E. Keogh, " On the non-trivial generalization of dynamic time warping to the multi-dimensional case," in *Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 289–297.

[52] J. Grabocka and L. Schmidt-Thieme, " Neuralwarp: Time-series similarity with warping networks," arXiv preprint arXiv: 1812.08306, 2018.

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

[54] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, " Practical black-box attacks against deep learning systems using adversarial examples," arXiv preprint arXiv: 1602.02697, vol. 1, no. 2, p. 3, 2016.

[55] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv preprint arXiv: 1312.6199, 2013.

[56] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2006, pp. 535–541.

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

[58] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," arXiv preprint arXiv: 1511.05644, 2015.

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, " Generative adversarial nets," *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[60] P. Baldi, " Autoencoders, unsupervised learning, and deep architectures," in *Proc. ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 37–49.

[61] D. P. Kingma and M. Welling, " Auto-encoding variational Bayes," arXiv preprint arXiv: 1312.6114, 2013.

[62] P. Mirowski, H. Steck, P. Whiting, R. Palaniappan, M. MacDonald, and T. K. Ho, " KL-divergence kernel regression for non-Gaussian fingerprint based localization," in *Proc. IEEE Int. Conf. Indoor Positioning and Indoor Navigation*, 2011, pp. 1–10.

[63] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[64] D. P. Kingma and M. Welling, " An introduction to variational autoencoders," arXiv preprint arXiv: 1906.02691, 2019.

[65] Y. LeCun *et al.*, " LeNet-5, convolutional neural networks," [Online]. Available: http://yann.lecun.com/exdb/lenet, vol. 20, p. 5, 2015.

[66] C. Franz and M. C. Franz, "Package 'cramer'," 2019.

[67] G. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Proc. NIPS*, vol. 15. Citeseer, 2002, pp. 833–840.

Samuel Harford (S' 18) received the B.S. and M.S. degrees in industrial engineering from the University of Illinois at Chicago, USA, in 2021 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago. His research interests lie in the domain of machine learning, time series classification, and heath care data mining.

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, and the Ph.D. degree in industrial engineering from the University of Illinois at Chicago in 2019. Currently, he is a Senior Research Specialist at Prominent Laboratory, the foremost research facility in process mining at the University of Illinois at Chicago and his research interests include health care classification and time series classification.

Houshang Darabi (S'98–A'00–M'10–SM'14) received the Ph.D. degree in industrial and systems engineering from Rutgers University, USA, in 2000. He is currently a Professor with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago. His research has been supported by several agencies, such as the National Science Foundation, the National Institute of Standard and Technology, and the National Institute of Occupational Safety and Health. His current research interests include the application of data mining, process mining, and time series classification in design and analysis of healthcare, safety, and education systems.