NEW DIRECTIONS IN ROBUST TIME-SERIES MACHINE LEARNING:

THEORY, ALGORITHMS, AND APPLICATIONS

By

TAHA BELKHOUJA

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2024

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of TAHA BELKHOUJA find it satisfactory and recommend that it be accepted.

Janardhan Rao Doppa, Ph.D., Chair

Yan Yan, Ph.D.

Diane J. Cook, Ph.D.

Ganapati Bhat, Ph.D.

ACKNOWLEDGMENT

This work wouldn't have been accomplished if not for a lot of guidance, assistance, and inspiration from many people to whom I am eternally grateful, and I would like to express my gratitude:

First and foremost, I would like to express my deepest gratitude to Prof. Jana Doppa, who has played a vital role in making this journey possible since Day one. From the beginning of my journey as a PhD student, Prof. Doppa helped me grow so much as a scientist and the critical thinker I am now, and always pushed me to aim higher. Without his guidance, continuous support and dedicated involvement, this work would not have been printed on this manuscript you are reading. His uncompromising support has been a guiding light, especially during times when I had my doubts about myself and my work. During the many moments of uncertainty, he always allowed me the time to take a deep breath and address any issues, all this while ensuring I kept a positive outlook.

Secondly, I want to express my gratitude to the excellent collaborators I was fortunate to work with. Prof. Ganapti Bhat was an exemplary collaborator. I am especially grateful for the generous time and expertise he shared with me while I was working on investigating the real-world impact of my work. I have enjoyed working with him and with Dina Hussein on the different challenges of the Human Activity Recognition area. To Prof. Yan Yan, thank you for contributing to my professional growth and continuous support to develop the theoretical side of my research. To Prof. Diane Cook, thank you for the insightful critiques that have shaped my research growth. Your insightful questions and constructive feedback have been instrumental in strengthening my dissertation.

My gratitude extends to my cherished friends and labmates, some of whom have shaped my time at Washington State University. Aryan Deshwal, thank you for your critical support and thoughtful insights and for being the humble, brilliant man you are. Alaleh Ahmadian and Iman Mirzadeh, your friendship and support throughout my time at WSU have made it

the amazing experience I have now in my memory. Syrine, my partner, my family, and my labmate, words cannot express my gratitude and appreciation for your tremendous support.

Finally, to my family, my pillars of strength, I owe an immeasurable debt of gratitude.

NEW DIRECTIONS IN ROBUST TIME-SERIES MACHINE LEARNING: THEORY, ALGORITHMS, AND APPLICATIONS

Abstract

by Taha Belkhouja, Ph.D.
Washington State University
May 2024

Chair: Janardhan Rao Doppa

Despite the rapid progress in research on the robustness of deep neural networks (DNNs) for images and text, there is little principled work for the time-series domain. Since time-series data arises in diverse applications, including mobile health, finance, and smart grid, it is important to verify and improve the robustness of DNNs for the time-series domain. Safe deployment of time-series DNNs for real-world applications relies on their ability to be resilient against natural/adversarial perturbations and anomalous inputs that may affect their predictive performance. This dissertation studies the design of robust machine learning (ML) algorithms that aim to minimize both the risk and uncertainty of wrongful decisions made by time-series-based ML systems from both theoretical and algorithmic perspectives. First, we investigate the robustness against adversarial time-series inputs. Adversarial examples were shown to be successful in exposing fundamental blind spots in ML models. While adversarial examples expose how to break the models, the process of creating adversarial examples can itself improve the robustness of ML models by adding them to the training set. The time-series modality poses unique challenges for studying adversarial robustness that are not seen in images and text. The key challenge is how to assess the similarity in the

time-series input space to efficiently create valid time-series adversarial examples. Second, we investigate the challenge of Out-of-Distribution (OOD) detection, where the ML system is required to identify time-series inputs that do not follow the distribution of training data. This is a critical task as deep models often make predictions that are very confident yet incorrect on such examples. Detecting OOD examples is challenging, and the potential risks are high for sensitive applications. The key challenge for time-series inputs is how to identify the features that improve the separability between OOD examples and training examples.

Motivated by these goals, this dissertation proposes and evaluates a suite of novel solutions to push the frontiers of robust time-series ML: 1) The practical threats of adversarial examples to time-series ML systems; 2) The use of constraints on statistical features of the time-series data to construct adversarial examples, and providing formal robustness certificates for time-series data; 3) The use of elastic measures such as Dynamic Time Warping to quantify the similarity between time-series examples and developing theoretically-sound algorithms to efficiently construct valid adversarial examples, and to train robust ML models by explicitly solving a min-max optimization problem; 4) Adapting and applying the developed algorithms to real-world applications including wearable sensors enabled ML systems for healthcare to handle both natural perturbations and missing sensor data; and 5) A novel OOD detection algorithm based on deep generative models for the time-series domain and explain why prior OOD methods from the other domains perform poorly.

TABLE OF CONTENTS

Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

Dedication

To my parents, my guiding shining lights, whose faith in me always sustains,

To my brothers and sister, your solidarity along my journey forever remains,

To my wife, my inspiration, my anchor in life, whose support never abstains.

CHAPTER ONE

INTRODUCTION

In recent years, rapid progress in the fields of machine learning (ML) and artificial intelligence (AI) has yielded significant breakthroughs across a wide range of challenging application domains, including, but not limited to, computer vision, natural language processing, autonomous decision-making, and strategic planning. Such progress has brought excitement about the great potential for AI to upgrade and confidently automate areas such as healthcare (Rajpurkar et al., 2022), science (H. Wang et al., 2023), and finance (L. Cao, 2022). However, this rapid evolution and quick integration of AI and ML systems into societal and personalized applications have materialized a wide range of critical concerns. Privacy issues arising from the extensive data requirements of ML models raise questions about the ethical collection, handling, and protection of personal information. Similarly, the security vulnerabilities inherent in AI systems (Hu et al., 2021) stress the need for robust deployment of these systems against potential infringements or misuse. Additional concerns also arise around the fairness and bias of algorithmic decisions, economic implications, and fully autonomous systems. The diverse progress in AI technologies promises to be extensively beneficial for humanity. Still, it is also noteworthy to seriously consider potential challenges and risks at an early stage.

There is a growing literature in the machine learning field highlighting issues related to the safety of AI, including robustness, risk sensitivity, and safe exploration. However, as machine learning systems are deployed in increasingly large-scale, autonomous, open-domain situations, it is worth reflecting on the scalability of such approaches and understanding what challenges remain to reduce accidental risk in modern machine learning systems. Overall, a careful dissection is needed to investigate concrete technical problems relating to safety assurance in machine learning systems for high-stake applications.

To efficiently address a few concrete safety and robustness problems of machine learning

that impact substantial real-world applications, we first define safety in machine learning as the minimization of both risk and uncertainty of wrongful decisions made by the system. As such, we aim to achieve an ML system that is resilient to real-world deployment vulnerabilities. A significant vulnerability such systems face is the uncertainty around the inputs which would be provided to them. Variations in data quality, such as acquisition quality, noise, or missing values, can significantly degrade the performance of ML models that were trained on curated datasets. Additionally, these systems may face scenarios of concept drift once deployed in the highly dynamic and complex real-world scenarios, where the statistical properties of the target variable change over time, leading to decreased accuracy and reliability. Moreover, the presence of the data that significantly deviates from the training distribution, known as outliers or anomalies, can further complicate the task of making accurate predictions, necessitating robust and adaptive models capable of handling such variability continuously. This dissertation focuses on the robustness of machine learning systems in the complex time-series input space during their deployment (aka research directions in robust time-series ML). From financial market forecasting (Ozbayoglu, Gudelek, and Sezer, 2020) to health monitoring (Ignatov, 2018) and predictive analytics in industrial settings (Z. Zheng et al., 2017), time-series analysis enables the extraction of meaningful patterns and trends over time, facilitating proactive decision-making and insights.

Deep neural networks (DNNs) have shown great success in learning accurate predictive models from time-series data (Zhiguang Wang, W. Yan, and Oates, 2017). In spite of their success, very little is known about their robustness. Most of the prior work on robustness for DNNs is focused on image domain (Z. Kolter and Madry, 2018) and natural language domain (W. Y. Wang, Singh, and J. Li, 2019). Time-series domain poses unique challenges (e.g., sparse peaks, fast oscillations) that are not encountered in both image and natural language processing domains. There is limited prior work on filtering methods in the signal processing literature to *automatically* identify invalid time-series inputs.

Vulnerability against adversarial (and natural) noise. Adversarial attacks (I. J. Good-

fellow, Shlens, and Szegedy, 2014) is a widely-known threat for DNNs that relies on subtle manipulations of the input data to craft inputs that will cause a highly-performing AI system to make erroneous predictions. Most of the prior work on adversarial robustness for DNNs is focused on the semantics of image domain (Z. Kolter and Madry, 2018) and natural language domain (W. Y. Wang, Singh, and J. Li, 2019). Adversarial methods rely on small perturbations to create worst possible scenarios from a learning agent's perspective. These perturbations are constructed by bounding l_p-norm (with p=2 or ∞, and sometimes p=1) and depend heavily on the input data space: they can be a small noise to individual pixels of an image or word substitutions in a sentence. Adversarial examples expose the brittleness of DNNs and motivate methods to improve their robustness. The time-series modality poses unique challenges for studying adversarial robustness that are not seen in images nor in text. The standard approach of imposing an l_p-norm bound to create worst possible scenarios from a learning agent's perspective does not capture the true similarity between time-series instances. Consequently, l_p-norm constrained perturbations can potentially create adversarial examples that correspond to a completely different class label. Hence, adversarial examples from prior methods based on l_p-norm will confuse the learner when they are used to improve the robustness of DNNs. In other words, the accuracy of DNNs will degrade on real-world data after adversarial training (Belkhouja and Doppa, 2020a).

Prior work on improving the robustness of DNNs has largely focused on training these models to withstand adversarial attacks or input perturbations within a defined l_p-norm constraint. Strategies for improving DNN robustness can be categorized into two principal approaches: the first involves adversarial training using data augmentation techniques, such as the incorporation of adversarial examples or noisy inputs into the training set. The second strategy entails the optimization of a specific loss function tailored to a robustness criterion, for instance, ensuring that similar input images yield comparable predictions from the DNN. However, it is noteworthy that the majority of these methods are optimized for image data and rely on the l_p norm to measure the imperceptibility of the attack. Given the distinct

attributes of time-series data, such measures can rarely capture the true similarity between time-series pairs and prior methods are likely to fail on time-series data. Consequently, a pivotal question needs to be addressed: How can we develop effective methodologies for training DNNs to be robust in the time-series domain, taking into account its specific challenges?

Vulnerability against unknown distributions. To ensure reliable and safe prediction from an ML model, minimizing the generalization error is not enough. One of the failure scenarios in the AI safety domain is confident predictions on Out-Of-Distribution (OOD) examples. Such examples, not observed during the training phase or outside the intended context of deployment, pose a significant risk of leading to unsafe decision-making outcomes. Safe and reliable deployment of machine learning systems require the ability to detect time-series data that do not follow the distribution of training data, also known as the in-distribution (ID). For example, a circumstantial event for an epilepsy patient or a sudden surge in one branch of a smart grid will result in sensor readings that deviate from the training data distribution. Another important application of OOD detection for the time-series domain is synthetic data generation. Many time-series applications suffer from limited or imbalanced data, which motivates methods to generate synthetic data (K. E. Smith and A. O. Smith, 2020). A key challenge is to automatically assess the validity of synthetic data, which can be alleviated using accurate OOD detectors.

There is a growing body of work on OOD detection for the image domain (Dan Hendrycks, 2017; W. Liu et al., 2020; S. Liang, Y. Li, and Srikant, 2018; Z. Xiao, Q. Yan, and Amit, 2020; Y.-Y. Yang et al., 2020; T. Cao et al., 2020) and other types of data such as genomic sequences (Ren et al., 2019). These methods can be categorized into

- Supervised methods that fine-tune the ML system or perform specific training to distinguish examples from ID and OOD.
- Unsupervised methods that employ Deep Generative Models (DGMs) on unlabeled data to perform OOD detection.

However, time-series data with its unique characteristics pose unique challenges that are not encountered in the image domain:

- Spatial relations between pixels are not similar to the temporal relations across different time-steps of time-series signals.
- Pixel variables follow a categorical distribution of values $\{0, 1, \cdots, 255\}$ where as time-series variables follow a continuous distribution.
- The semantics of images (e.g., background, edges) do not apply to time-series data.
- Humans can identify OOD images for fine-tuning purposes, but this task is challenging for time-series data. Hence, prior OOD methods are not suitable for the time-series domain.

1.1 Summary of Dissertation Research

This dissertation develops a suite of novel algorithms and associated theory to significantly push the frontiers of robust machine learning for the time-series domain. First, we propose different frameworks that address the challenges of creating adversarial examples that are suited for the time-series input space. The key idea is to create adversarial algorithms that look beyond l_p norms to assess the similarity between time-series examples using statistical features of time-series signals and elastic measures such as dynamic time warping. Second, we address the challenges of training robust ML models beyond the conventional adversarial training (Rawat, Wistuba, and Nicolae, 2017) by explicitly solving a min-max optimization problem. The goal is that during the training process, we automatically make deep models learn robust feature embedding without requiring a well-curated data augmentation procedure (Stephan Zheng et al., 2016). Third, we address the out-of-distribution detection challenge for time-series data. This is the first work on solving OOD detection for time-series data. Finally, in collaboration with Dina Hussein and Prof. Ganapti Bhat, we demonstrate

the real-world impact of the developed robust time-series ML algorithms by applying them to wearable sensors-enabled mobile applications by addressing natural perturbations and missing sensor data.

1.2 Summary of Technical Contributions

The main contribution of this dissertation is the development of suits of novel algorithms and theory to significantly improve the robustness of machine learning systems for time-series data. Specific contributions include:

- The design of a novel framework referred as **M***ultivariate* **T***ime-***S***eries* **Ad***versarial* **Lens** *(MTS-AdLens)* (Belkhouja and Doppa, 2020a) to analyze deep models for predictive analytics of time-series data in real-world IoT and mobile applications in the adversarial setting. *To the best of our knowledge, this was the first principled study of adversarial robustness of DL methods for analytics on multivariate time-series data in IoT and mobile applications by considering the real-world constraints.* The framework also includes a novel defense method that can be deployed at the inference stage in IoT and mobile applications.

- The development of two novel frameworks to create adversarial examples suitable for time-series data by addressing the similarity challenges.

  - The first is referred to as **T***ime-***S***eries* **A***ttacks via* **STAT***istical Features (TSA-STAT)* (Belkhouja and Doppa, 2022; Belkhouja and Doppa, 2023). TSA-STAT creates adversarial examples by imposing constraints on statistical features of the clean time-series signal. Additionally, TSA-STAT formulation allows the derivation of a theoretical certified bound for the robustness of ML models due to adversarial attacks.

  - The second referred as *Dynamic Time Warping for Adversarial Robustness (DTW-*

AR) (Belkhouja, Yan Yan, and Doppa, 2023a). DTW-AR employs the dynamic time warping measure (Sakoe, 1971; Müller, 2007) to measure a realistic similarity between two time-series signals (Berndt and Clifford, 1994; Müller, 2007).

Both frameworks and the TSA-STAT certification guarantees are designed to be applicable to any DNN for time-series domain with different neural network structures.

- The design of a principled framework referred to as **RO***bust Training for* **T***ime-***S***eries (RO-TS)* (Belkhouja, Yan Yan, and Doppa, 2022) to train robust DNNs for time-series data. We formulate a novel *min-max* optimization problem to reason about the robustness criteria in terms of disturbances to time-series using the global alignment kernel (GAK) measure (M. Cuturi et al., 2007); and provide a theoretically-sound optimization algorithm to solve it.

- The development of a novel OOD detection algorithm for the time-series domain referred to as Seasonal Ratio Scoring (SRS) (Belkhouja, Yan Yan, and Doppa, 2023b). SRS employs the Seasonal and Trend decomposition using Loess (STL) (Cleveland et al., 1990) and deep generative models to enhance the separability between in-distribution and OOD examples.

- Two concrete instantiations[1] of robust time-series ML algorithms to wearable sensors-enabled mobile applications by addressing natural perturbations and missing data.

  - *StatOpt* (Hussein, Belkhouja, et al., 2022) framework, a concrete instantiation of the statistical optimization approach TSA-STAT (Belkhouja and Doppa, 2022), to enable reliable ML classifiers on low-power wearable devices.

  - *Accuracy-Preserving Imputation (AIM)* (Hussein, Belkhouja, et al., 2023), a novel and energy-efficient search-based approach to produce accuracy-preserving imputations for missing sensor data at runtime.

[1]These works were developed in close collaboration with Dina Hussein and Ganapti Bhat

1.3 Outline of the Thesis

The remaining part of the dissertation is organized as follows.

In Chapter Two, we first provide an overview of the problem setup for the different threats on robust Time-Series deep learning settings studied in this dissertation. Next, we discuss the necessary background material on these threats. Finally, we define the evaluation metrics to measure the efficacy of any proposed algorithms for time-series data that aim to enhance the robustness of Time-Series deep learning models.

In Chapter Three, we describe a primary setting for analyzing deep models for predictive analytics of time-series data in real-world IoT and mobile applications through adversarial lens, namely **M***ultivariate* **T***ime-***S***eries* **Ad***versarial* **Lens** *(MTS-AdLens)*. We provide in this chapter a principled study of the adversarial robustness of DL methods for analytics on multivariate time-series data in IoT and mobile applications by considering the real-world constraints and novel defensive methods against adversarial threats motivated by deployment constraints. We first describe the key challenges in both hardware, data, and algorithms to study the adversarial robustness of deep models for multivariate time-series data. Then, we develop effective attacks that expose significant vulnerabilities of deep models for multivariate time-series sensor data in realistic settings. Finally, we propose a novel defense method at the inference stage to improve the adversarial robustness of deep models.

In Chapter Four, we describe the proposed framework referred to as **T***ime-***S***eries* **A***ttacks via* **STAT***istical Features (TSA-STAT)* to create adversarial examples for time-series data. We describe TSA-STAT as a principled approach to create targeted adversarial examples for the time-series domain using statistical constraints and polynomial transformations. Then, we provide theoretical analysis to prove that polynomial transformations expand the space of valid adversarial examples over additive perturbations. Additionally, we derive theoretically certified bounds (Cohen, Rosenfeld, and J Zico Kolter, 2019) for adversarial robustness of TSA-STAT that is applicable to any deep model for time-series domain.

In Chapter Five, we describe a different adversarial framework for time-series domain referred as *Dynamic Time Warping for Adversarial Robustness (DTW-AR)*. We describe the theoretical and empirical analysis to demonstrate the effectiveness of DTW over the standard l_2 distance metric for adversarial robustness studies. We also describe the principled algorithm using the Dynamic Time Warping measure that efficiently creates diverse adversarial examples justified by theoretical analysis.

In Chapter Six, we address the challenge of training robust DNNs for time-series domain by proposing **RO***bust Training for* **T***ime-***S***eries (RO-TS)* framework. RO-TS employs additive noise variables to formulate a *min-max* optimization problem to reason about the robustness criteria in terms of disturbances to time-series inputs by minimizing the worst-case risk. We develop a principled theoretical framework *stochastic compositional alternating gradient descent ascent (SCAGDA)* algorithm to solve compositional min-max optimization problems, to which RO-TS belongs.

In Chapter Seven, we transition to the out-of-distribution threat and describe the proposed novel OOD detection algorithm for the time-series domain referred to as Seasonal Ratio Scoring (SRS). We describe the principled algorithm based on STL decomposition and deep generative models to compute the Seasonal Ratio (SR) score to detect OOD time-series examples. Additionally, we provide a novel time-series alignment algorithm based on dynamic time warping to improve the effectiveness of the SR score-based OOD detection. Finally, we provide a formulation of the experimental setting for time-series OOD detection and the evaluation of SRS algorithm on real-world datasets and comparison with state-of-the-art baselines.

In Chapters Eight and Nine, we describe the two concrete instantiations[2] of robust deep learning algorithms for the Human Activity Recognition (HAR) field.

- In Chapter Eight, we describe *StatOpt*, the concrete instantiation of the TSA-STAT approach in Chapter 4, to enable reliable ML classifiers for HAR on low-power wear-

[2]These works were developed in close collaboration with Dina Hussein and Ganapti Bhat

able devices. We characterize and evaluate the real-world mismatch between sensor data distributions from training and real-world deployment for ML-based HAR applications. Then, we describe the StatOpt framework to create additional training examples capturing the statistical properties of sensor disturbances for training reliable HAR classifiers.

- In Chapter Nine, we describe the novel and energy-efficient search-based *Accuracy-Preserving Imputation (AIM)* approach to producing accuracy-preserving imputations for missing sensor data at runtime. We describe how AIM identifies the most likely data patterns for imputing missing sensor data through a novel formulation for this search problem. Additionally, we provide experimental validation on four diverse wearable datasets to demonstrate that AIM enables reliable imputation of missing sensor data with minimal overhead.

Finally, in Chapter Ten, we provide a summary of the dissertation.

CHAPTER TWO

PROBLEM SETUP AND RELATED WORK

Time-series predictive analytical tasks correspond to finding a mapping from synchronous temporal observations data to semantic labels representing the state of the system. Let $X \in \mathbb{R}^{n \times T}$ be a multi-variate time-series signal, where n is the number of channels (representing synchronous sensors) and T is the window-size of the signal. Given a set of training examples in the form of input time-series example X and output label y pairs, our goal is to learn a function approximator F_θ whose output predictions have high accuracy on unseen input time-series examples. We consider F_θ as a DNN classifier that maps $\mathbb{R}^{n \times T} \rightarrow \mathcal{Y}$, where θ stands for parameters of the model to be estimated using training data and $\mathcal{Y}$ is the set of classification labels y. For example, in a health monitoring application using physiological sensors for patients diagnosed with cardiac arrhythmia, we use the measurements X from wearable devices to predict the likelihood of a cardiac failure for a set of potential diagnosis $\mathcal{Y}$.

Adversarial threat. Adversarial examples expose the brittleness of DNNs and motivate methods to improve the robustness of classifiers. The goal of am adversarial attack strategy is to tamper with the input data from sensors to create an adversarial input X_{adv} such that:

- The tampered input is highly similar to the original input: $\|X_{adv} - X\|_p \leq \epsilon$, where ϵ is minimal given an l_p norm $\|.\|_p$
- The adversarial example is misclassified by the DNN: $F_\theta(X_{adv}) \neq F_\theta(X)$

We define the space of potential attack threat on an input X as:

$$\left\{ X_{adv} \;\middle/\; \|X_{adv} - X\|_p \leq \epsilon \text{ and } F_\theta(X) \neq F_\theta(X_{adv}) \right\}$$

where ϵ defines the neighborhood of highly-similar examples for input X depending on the deployment application.

This threat affect the reliability of deployed classifier as the model will output wrong predictions with high confidence on adversarial examples. To mitigate this threat, we aim to design and train a robust classifier F_θ^* such that:

$$\text{Given a user-defined } \epsilon,\ \forall X_{adv} \text{ such that } \|X_{adv} - X\|_p \leq \epsilon \Rightarrow F_\theta^*(X) = F_\theta^*(X_{adv}) \quad (2.1)$$

Out-of-distribution threat. Let $\mathcal{D}_{in}$ be an in-distribution (ID) time-series set with d examples $\{(X, y)\}$ sampled from the distribution P^* defined on the joint space of input-output pairs $(\mathcal{X}, \mathcal{Y})$. Each $X \in \mathbb{R}^{n \times T}$ from $\mathcal{X}$ is a multi-variate time-series input. We consider the time-series classifier F_θ to be trained using $\mathcal{D}_{in}$. Out-of-distribution (OOD) samples (X, y) are typically generated from a distribution other than P^*. Specifically, we consider a sample (X, y) to be OOD if the class label y is different from the set of in-distribution class labels, i.e., $y \notin \mathcal{Y}$. By default of the design, the classifier F_θ learned using $\mathcal{D}_{in}$ will assign one of the class labels in $\mathcal{Y}$ when encountering an OOD sample (X, y). This threat affect the reliability of deployed classifier as the model will output wrong predictions with high confidence on OOD samples.To mitigate this threat, we aim to design a detector D that flags OOD examples and discard them from the machine learning system pipeline:

$$D(X, y) = \begin{cases} 1 & \text{if } X \text{ is OOD} \\ 0 & \text{if } X \text{ is ID} \end{cases} \quad (2.2)$$

2.1 Background and Related Work

Adversarial threat. Prior work for creating adversarial examples mostly focus on image and natural language processing (NLP) domains (Z. Kolter and Madry, 2018; W. Y. Wang, Singh, and J. Li, 2019). For the image domain, such methods include general attacks such as Carlini & Wagner (CW) attack (Carlini and D. Wagner, 2017) and universal attacks (Moosavi-Dezfooli, A. Fawzi, O. Fawzi, et al., 2017). CW is an instance-specific attack that relies on solving an optimization problem to create adversarial examples by controlling the

adversarial confidence score to fool the target deep model. Universal attacks are a class of adversarial methods that are not input-dependent. The goal of universal attacks is to create a universal perturbation that can be added to any input to create a corresponding adversarial example. The Frank-Wolfe attack (J. Chen, D. Zhou, et al., 2020) improves the optimization strategy for adversarial examples to overcome the limitations of projection methods. Recent work regularizes adversarial example generation methods to obey intrinsic properties of images. The work of (Laidlaw and Feizi, 2019b) enforces a smoothness regularizer on the adversarial output such that similar-color pixels are perturbed following the same direction. Other works have employed spatial transformation within a perceptual threshold (C. Xiao et al., 2018) or a semantic-preserving transformation (Hosseini et al., 2017) to regularize the output. These methods exploit the intrinsic characteristics of images to control and regularize the algorithm to create adversarial examples. Expectation Over Transformation (EOT) (Athalye, Engstrom, et al., 2018) approach creates robust adversarial examples that are effective over an entire distribution of transformations by maximizing an expectation of the log-likelihood given transformed inputs. These transformations include perceptual distortion of a given image such as rotation or texture modification. RayS method (J. Chen and Gu, 2020) was also proposed to improve the search over adversarial examples using a sanity check that is specific for the image domain. (Baluja and Fischer, 2018) proposed to use Adversarial Transformation Network (ATN) to automatically create adversarial examples for any given input. (Karim, Majumdar, and Darabi, 2020) investigated the use of ATNs for time-series data. The main findings include ATN fails to find adversarial examples for many inputs and not all targeted attacks are successful to fool DNNs. While adversarial attacks perturb pixel values in the image domain, they perturb characters and words in the NLP domain. For example, adversarial attacks may change some characters to obtain an adversarial text which seems similar to the reader, or change the sentence structure to obtain an adversarial text which is semantically similar to the original input sentence (e.g., paraphrasing). One method to fool text classifiers is to employ the saliency map of input words to generate adversarial

examples while preserving meaning under the white-box setting (Samanta and Mehta, 2017). A second method named DeepWordBug (J. Gao et al., 2018) employs a black-box strategy to fool classifiers with simple character-level transformations.

The most successful empirical defense known so far is adversarial training (Tramer et al., 2020) that employs adversarial algorithms to augment training data. This method is intuitive as it relies on feeding DNNs with adversarial examples in order to be robust against adversarial attacks. Other defense methods have been designed to overcome the injection of adversarial examples and the failure of deep models. (Athalye, Carlini, and D. Wagner, 2018) proposed different attack techniques to show that a defense method such as obfuscated gradients is unable to create a robust deep model. Distillation technique (Papernot, McDaniel, X. Wu, et al., 2016) has also been proposed as a defense against adversarial perturbations, where a smaller neural network is used as an auxiliary network to improve the robustness of the overall learned model. It was shown empirically that such techniques can reduce the success rate of adversarial example generation. However, (Papernot, McDaniel, I. Goodfellow, et al., 2017) proved experimentally that this defense method is useful only when the attack is performed directly on the distilled model (i.e., white-box settings). For black-box settings, it was found that the adversarial attacks evade the distillation scheme used by the target. (Tramèr et al., 2018) analyzed adversarial training and its transferability property to explain how robust deep models should be attained. To improve adversarial training through a min-max optimization formulation, (Xiong and Hsieh, 2020) tries to learn a recurrent neural network to guide the optimizer to solve the inner maximization problem of the min-max training objective.

To further improve empirical defenses, the concept of *certifiable robustness* was introduced. A deep model is certifiably robust for a given input X, if the prediction of X is guaranteed to be constant within a small neighborhood of X, e.g., l_p ball. Raghunathan, Steinhardt, and P. Liang, 2018 provides certificates for one-hidden-layer neural networks using semi-definite relaxation. In Hein and Andriushchenko, 2017, certification is an instance-

specific lower bound on the tampering required to change the classifier's decision with a small loss in accuracy. In a recent work (Cohen, Rosenfeld, and J Zico Kolter, 2019; Bai Li et al., 2019), the robustness of deep models against adversarial perturbation is connected to random noise. These methods certify adversarial perturbations for deep models under the l_2 norm. Cohen, Rosenfeld, and J Zico Kolter, 2019 defined two families for certification methods: 1) *Exact* methods report the existence or the absence of a possible adversarial perturbation within a given bound. This goal has been achieved using feed-forward multi-layer neural networks based on Satisfiability Modulo Theory (X. Huang et al., 2017) or modeling the neural network as a 0-1 Mixed Integer Linear Program (Fischetti and Jo, 2018). However, these methods suffer from scalability challenges. 2) *Conservative* methods either confirm that a given network is robust for a given bound or report that robustness is inconclusive (Bai Li et al., 2019). Our proposed robustness certificate for TSA-STAT falls in the conservative category and extends the recent method based on random noise (Bai Li et al., 2019).

There is little to no principled prior work on adversarial methods for time-series domain. (H Ismail Fawaz et al., 2019) employed the standard Fast Gradient Sign method with l_2-norm bound (Kurakin, I. Goodfellow, and Bengio, 2016) to create adversarial noise with the goal of reducing the confidence of deep convolutional models for classifying *uni-variate* signals. Network distillation is also employed to train a student model for creating adversarial attacks (Karim, Majumdar, and Darabi, 2020). In an orthogonal work, the study from (Siddiqui et al., 2019) concluded that time-series signals are highly complex, and their interpretability is ambiguous. Adversarial examples can also be studied for regression tasks over time-series data. However, there is very limited work in this direction as explained by (Siddiqui et al., 2019). These methods consider Euclidean distance and employ standard methods from the image domain such as FGSM (Mode and Hoque, 2020). In an orthogonal/complementary direction, generative adversarial networks are used to impute missing values in time-series data (Luo, Zhang, et al., 2019; Luo, Cai, et al., 2018).

Since characteristics of time-series (e.g., fast-pace oscillations, sharp peaks) are different

from images and text, most existing adversarial algorithms are not efficient or not applicable to time-series data. To overcome the limitations of the existing work, we propose:

1. MTS-AdLens framework that analyzes the adversarial threat when the input space is a multi-variate time-series data.

2. TSA-STAT, DTW-AR and RO-TS frameworks that overlook the standard l_p-norm distance as a similarity measure between inputs and investigate more efficient time-series-based similarity measures: Statistical features and elastic measures. The proposed frameworks overcome the challenges of the high complexity of time-series-based similarity measures by providing efficient algorithmic solutions to create a practical framework.

3. The development of theoretical guarantees for adversarial robustness through the TSA-STAT framework, where we propose a new derivation of a certified bound for adversarial robustness that is applicable to any deep model for time-series domain.

Out-of-distribution threat. The work on OOD detection can be mainly classified into the following categories:

1. **OOD detection via pre-trained models.** Employing pre-trained deep neural networks (DNNs) to detect OOD examples was justified by the observation that DNNs with ReLU activation can produce arbitrarily high softmax confidence for OOD examples (Dan Hendrycks, 2017). Maximum probability over class labels has been used (Dan Hendrycks, 2017) to improve the OOD detection accuracy. Building on the success of this method, temperature scaling and controlled perturbations were used (S. Liang, Y. Li, and Srikant, 2018) to further increase the performance. The Mahalanobis-based scoring method (Kimin Lee, Kibok Lee, et al., 2018) is used to identify OOD examples with class-conditional Gaussian distributions. Gram matrices (Sastry and Oore, 2020)

were used to detect OOD examples based on the features learned from the training data.

2. **OOD detection via synthetic data.** During the training phase, it is impossible to anticipate the OOD examples that would be encountered during the deployment of DNNs (Hendrycks et al., 2019). Hence, unsupervised methods (Yu and Aizawa, 2019) are employed or synthetic-data-based on generative models is created (Kimin Lee, H. Lee, et al., 2018; Z. Lin et al., 2020) to explicitly regularize the DNN weights over potential OOD examples.

3. **OOD detection via deep generative models.** The overall idea of using deep generative models (DGMs) for OOD detection is as follows: 1) DGMs are trained to directly estimate the in-distribution P^*; and 2) The learned DGM identifies OOD samples when they are found lying in a low-density region. Prior work has used auto-regressive generative models (Ren et al., 2019) or GANs (Ziyu Wang et al., 2020) and proposed scoring metrics such as likelihood estimates to obtain good OOD detectors. DGMs are shown to be effective in evaluating the likelihood of input data and estimating the data distribution, which makes them a good candidate to identify OOD examples with high accuracy. However, as shown by (Nalisnick et al., 2018), DGMs can assign a high likelihood to OOD examples. Likelihood ratio (Ren et al., 2019) and likelihood regret (Z. Xiao, Q. Yan, and Amit, 2020) are proposed to improve OOD detection.

4. **OOD detection via time-series anomaly detection.** Generic Anomaly Detection (AD) algorithms (Pang, Shen, L. Cao, et al., 2021; Ruff et al., 2021) can be employed to solve OOD problems for time-series data. Anomaly detection is the task of identifying observed points or examples that deviate significantly from the rest of data. Anomaly detection relies on different approaches such as distance-based metrics or density-based approaches to quantify the dissimilarities between any example and the rest of the data. Current methods using DNNs (e.g., Generative Adversarial Net-

works, auto-encoders) showed higher performance in anomaly detection as they can capture more complex features in high dimensional spaces (Pang, Shen, L. Cao, et al., 2021; Pang, Shen, and Hengel, 2019). We note that there exist some AD methods that can cover the same setting as the OOD problem for time-series domain. However, both settings are still considered as two different frameworks with two different goals (Jingkang Yang et al., 2021). By definition, AD aims to detect and flag anomalous samples that deviate from a pre-defined normality (Laptev, Amizadeh, and Flint, 2015; Canizo et al., 2019) estimated during training. Under the AD assumption of normality, such samples only originate from a covariate shift in the data distribution (Ruff et al., 2021). Semantically, such samples do not classify as OOD samples (Jingkang Yang et al., 2021). For example, consider an intelligent system trained to identify the movement of a person (e.g, run, stand, walk, swim), where stumbling may occur during running. Such an event would be classified as an anomaly as the activity running is still taking place, but in an irregular manner. However, if the runner slips and falls, such activity should be flagged as OOD due to the fact that it does not belong to any of the pre-defined activity classes. In other words, OOD samples must originate from a different class distribution ($y_{\text{OOD}} \notin \mathcal{Y}$) than in-distribution examples, while anomalies typically originate from the same underlying distribution but with anomalous behavior. Open-set recognition methods can be applicable for this setting (D.-W. Zhou, Ye, and Zhan, 2021; D.-W. Zhou, Y. Yang, and Zhan, 2021) as it has been shown that they are effective in detecting unknown categories without prior knowledge. However, OOD Detection encompasses a broader spectrum of solution space and does not require the complexity of identifying the semantic class of the anomalies. Additionally, anomalies can manifest as a single time-step, non-static window-length, but not generally a complete time-series example in itself. Such differences can be critical for users and practitioners, which necessitates the study of separate algorithms for AD and OOD. Unlike anomaly detection, OOD detection focuses on identifying test samples with non-

overlapping labels with in-distribution data and can generalize to multi-class setting (Jingkang Yang et al., 2021).

In summary, the limitations of the existing work for the time-series OOD detection setting are:

- The effectiveness of pre-trained models as OOD detectors depends critically on the availability of a highly-accurate DNN for the classification task. However, this requirement is challenging for the time-series domain as real-world datasets are typically small and exhibit high class imbalance, resulting in inaccurate DNNs (Wen et al., 2020; Chen Huang et al., 2016).

- Synthetic data for the time-series domain is a highly challenging task and remains an open research problem due to the limited data and their ambiguity to be validated by human experts.

- Most generative models for OOD detection rely on assumptions specific to the input domain space (e.g., the data contains background pixels). There is no prior work on OOD detection via deep generative models for time-series data.

- OOD samples cannot be used as labeled anomalous examples during training due to the general definition of the OOD space. For various AD methods such as nearest neighbors and distance-based, the fine-tuning of the cut-off threshold between "normal" and "anomalous" examples requires anomaly labels during training. Mainly, window-based techniques(Chandola, Banerjee, and Kumar, 2010) require both normal and anomalous sequences during training, and if there are none, anomalous examples are randomly generated. Such a requirement is not practical for OOD problem settings as the distribution is ambiguous to define and sample from.

- AD assumes that normal samples are homogeneous in their observations. This assumption helps the AD algorithm to detect anomalies. Such an assumption cannot

hold for different classes of the in-distribution space for multi-class settings. Therefore, time-series AD algorithms are prone to fail at detecting OOD samples. Indeed, our experiments demonstrate the failure of state-of-the-art time-series AD methods.

Therefore, we propose the Seasonal Ratio Scoring (SRS) framework, the first work on OOD detection over time-series data (To the best of our knowledge). SRS overcomes the aforementioned challenges by dissecting the inputs using time-series decomposition tools and employing time-series-based DGMs to estimate the score used to separate ID from OOD examples.

2.2 Evaluation Protocol

Adversarial threat. The evaluation of adversarial attacks using the efficiency metric $\alpha_{Eff} \in [0, 1]$ over the created adversarial examples. α_{Eff} (higher means better attacks) measures the capability of adversarial examples to fool a given DNN F_θ to output the target class-label. α_{Eff} is calculated as the fraction of adversarial examples that are predicted correctly by the classifier:

$$\alpha_{Eff} = \frac{\#\text{ Adv. examples s.t.} F(X) == y_{target}}{\#\text{ Adv. examples}} \tag{2.3}$$

The evaluation of the robustness of models designed to withstand different adversarial attack strategies employs the prediction accuracy of each model (via ground-truth labels of time-series) as the metric. A DNN classifier is robust if it is successful in predicting the true label of any given adversarial example. To ensure robustness, DNN models should be least sensitive to different types of perturbations over original time-series signals.

Out-of-distribution threat. The evaluation of OOD detectors employs in general the following two standard metrics.

1. AUROC score: The area under the receiver operating characteristic curve is a threshold-independent metric that measures the separability between ID and OOC classes using

a given OOD detector. This metric (higher is better) is equal to 1.0 for a perfect detector and 0.5 for a random detector.

2. F1 score: It is the harmonic mean of precision and recall. Precision measures the ratio of the OOD examples that were correctly identified out of all the OOD examples that were identified by the detector. On the other hand, Recall measures the ratio of the OOD examples that were correctly identified out of all the actual OOD examples in the test set. Due to the threshold dependence of F1 score, the highest F1 score obtained with a variable threshold can be used to evaluate the method. This score has a maximum of 1.0 in the case of a perfect precision and recall scores.

CHAPTER THREE

ANALYZING DEEP LEARNING FOR TIME-SERIES DATA THROUGH ADVERSARIAL LENS IN MOBILE AND IOT APPLICATIONS

T. Belkhouja, J. Doppa. "Analyzing Deep Learning for Time-Series Data through Adversarial Lens in Mobile and IoT Applications." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 39(11): 3190-3201*, 2020.

Originally published in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

Attributions:

T. Belkhouja has contributed to this work by formulating the problem setting, investigating the state of the art, formulating the theoretical contribution and the algorithmic solution, implementing the algorithm and running the required empirical analysis to highlight the performance improvement of the proposed solution compared to the state of the art.

J. Doppa has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

ANALYZING DEEP LEARNING FOR TIME-SERIES DATA THROUGH ADVERSARIAL LENS IN MOBILE AND IOT APPLICATIONS

In this chapter, we describe the **M***ultivariate* **T***ime-***S***eries* **Ad***versarial* **Lens** *(MTS-AdLens)* framework that analyzes deep models for predictive analytics of time-series data in real-world IoT and mobile applications through adversarial lens. This chapter is a principled study of adversarial robustness of Deep Learning methods for analytics on multivariate time-series data in IoT and mobile applications that considers the real-world constraints. MTS-AdLens investigates the real-world hardware constraints and creates adversarial examples using a small number of channels. Motivated by this threat, we propose a novel defense method that can be deployed at the inference stage by leveraging the critical channel analysis. The key idea is to exploit the fact that the deep model relies on a small number of critical channels to make its prediction. MTS-AdLens comprehensive analysis:

1. Identifies the key challenges in hardware, data, and algorithms to study adversarial robustness of deep models for multivariate time-series data.

2. Develops highly effective attacks that expose significant vulnerabilities of deep models for multivariate time-series sensor data in realistic settings using critical channel analysis.

3. Uses a novel defense method at the inference stage to improve the adversarial robustness of deep models.

3.1 Problem Setup

Predictive analytical tasks in IoT and mobile applications correspond to finding a mapping from synchronous sensor data to semantic labels representing the state of the system. Suppose $X = [X_1, X_2, \cdots, X_n]$ is the input multivariate time-series example that is generated by a set of n heterogeneous synchronous sensors. Each sensor generates a one-dimensional

time-series X_i related to the event it is sensing. Given a set of training examples in the form of input time-series example and output label pairs, our goal is to learn a function approximator $F_\theta(X)$ whose output predictions have high accuracy on unseen input time-series examples, where θ stands for parameters of the model to be estimated using training data. Convolutional neural networks (CNNs) via 1D convolutions have been shown to perform very well for classifying multivariate time-series data including human activity recognition (Ignatov, 2018) and smart home monitoring (G. Chen et al., 2018). CNNs are inherently well-suited to recognize various patterns from synchronous sensor readings. For example, in human activity recognition tasks, lower layers capture the local emphatics of the signals that characterize the general nature of the human movement, while higher layers capture the patterns describing the combination of different movements (Jianbo Yang et al., 2015). Therefore, in this work, we focus on analyzing state-of-the-art CNN models for time-series classification in adversarial settings. Towards this goal, we consider two related problems.

Problem 1: Adversarial attack strategies. The goal of attack strategy is to tamper with the transmitted data from sensors to create an adversarial input X_{adv} such that $\|X_{adv} - X\| \leq \epsilon$ and $F(X, \theta) \neq F(X_{adv}, \theta)$, where ϵ is the difference between original and adversarial inputs, and should be minimal to be perceptible that the input has been changed. There are two main attack strategies:

1. **Input-specific adversarial attacks.** This type of attack constructs an adversarial input based on a *single* input example and output label pair: it creates an adversarial example that is very similar, but classified into a different class label using the deep neural network:

$$X_{adv} = X + \epsilon \cdot sign(\nabla_X J(\theta, X, y)) \tag{3.1}$$

   where $\nabla_X J$ is the gradient of the deep model's loss function with respect to image X, y is the true label of image X, and θ stands for model parameters.

2. **Universal adversarial attacks.** This type of attack will analyze a *collection* of input

example and output label pairs to construct a universal adversarial pattern. This pattern when applied to any input example will most likely result in deep network mis-classifying the given input (Moosavi-Dezfooli, A. Fawzi, O. Fawzi, et al., 2017). The goal is to find the minimal perturbation δ such that for any given input X, $F(X,\theta) \neq F(X+\delta,\theta)$ for $\|\delta\|_p \leq \epsilon$, where ϵ is the maximum threshold for human perception. To generate the universal adversarial pattern (Moosavi-Dezfooli, A. Fawzi, and Frossard, 2016; Moosavi-Dezfooli, A. Fawzi, O. Fawzi, et al., 2017) for multivariate time-series data, the following equation is solved:

$$\mathcal{P}_{X\sim\mathcal{D}}\left(F(X+v,\theta) \neq F(X,\theta)\right) \geq 1-\eta \text{ s.t. } \|v\|_p \leq \epsilon \tag{3.2}$$

where $X \sim \mathcal{D}$ represents that an input X follows the given data distribution $\mathcal{D}$, ϵ is the maximum magnitude of perturbation, and η is the minimal fooling rate desired to be induced for the target deep model.

Since our focus is on realistic vulnerability assessment of IoT and mobile systems, only black-box attacks (no knowledge is assumed about the target model) are feasible. To create black-box attacks (Rawat, Wistuba, and Nicolae, 2017), the attacker has the ability to query the target model to get predicted labels for any candidate input X: $y_X{=}F_\theta(X)$. This training data from querying the target model is used to create a *proxy* model $F'_{\theta'}(X)$ to devise strong attacks. The effectiveness of an attack strategy $\mathcal{AS}$ is measured by the *degradation in accuracy* of the target model $F_\theta(X)$ on adversarial inputs created by $\mathcal{AS}$. In this work, we focus on practical adversarial attacks that meet the real-world constraints of IoT and mobile applications. A concrete example of such an adversarial attack is the tampering with infrastructure signals for smart grids. If the sensors' readings that are collected centrally in smart-grid management are altered maliciously, the attack can lead the system to equipment failures or black-outs.

Problem 2: Defense methods for adversarial robustness. The goal of defense methods is to improve the robustness of deep model $F_\theta(X)$ against adversarial attacks. The

effectiveness of a defense strategy $\mathcal{DS}$ is measured by the *improvement in accuracy* of the model $F_\theta(X)$ with respect to the adversarial inputs created by a specific attack strategy $\mathcal{AS}$. In this work, we primarily focus on defense mechanisms at the inference stage due to their utility in critical applications such as smart health, where re-training of deep model to improve robustness is not practical due to privacy concerns. For example, adversarial robustness via defense methods can help epileptic patients to trust the Electroencephalographic sensors' readings in predicting seizures effectively.

3.2 MTS-AdLens: Multivariate Time-Series Adversarial Lens Framework

In this section, we described the details of our proposed *MTS-AdLens* framework for multivariate time-series data in IoT and mobile applications illustrated by Figure 3.1. To explain the figure better, we provide the following example for smart-health monitoring using wearable physiological sensors. For patients with heart conditions, they are equipped with sensors to monitor their heart signals. These sensors will represent the physical domain. The sensors will generate electrocardiographic signals in the form of time-series data, which is the digital representation of the heart condition. These signals are fed to a pre-trained deep model to infer if the condition of the heart is healthy or requires attention. The result of this inference mechanism represents the predictive analytics of the system. If attackers target this system, they will aim to tamper with the inferred condition. An example of such tampering is to prevent the detection of an irregular heart condition that could lead to a fatality. First, we explain how to create black-box attacks on sensor-based mobile systems ignoring the real-world constraints. Second, we discuss the real-world constraints imposed by these systems that limit the space of black-box attacks. Third, we propose some novel practical attack strategies based on identifying critical channels of time-series data to meet the physical constraints. Lastly, inspired by our vulnerability analysis of deep models for multivariate time-series data in these systems, we describe a novel defense technique at the

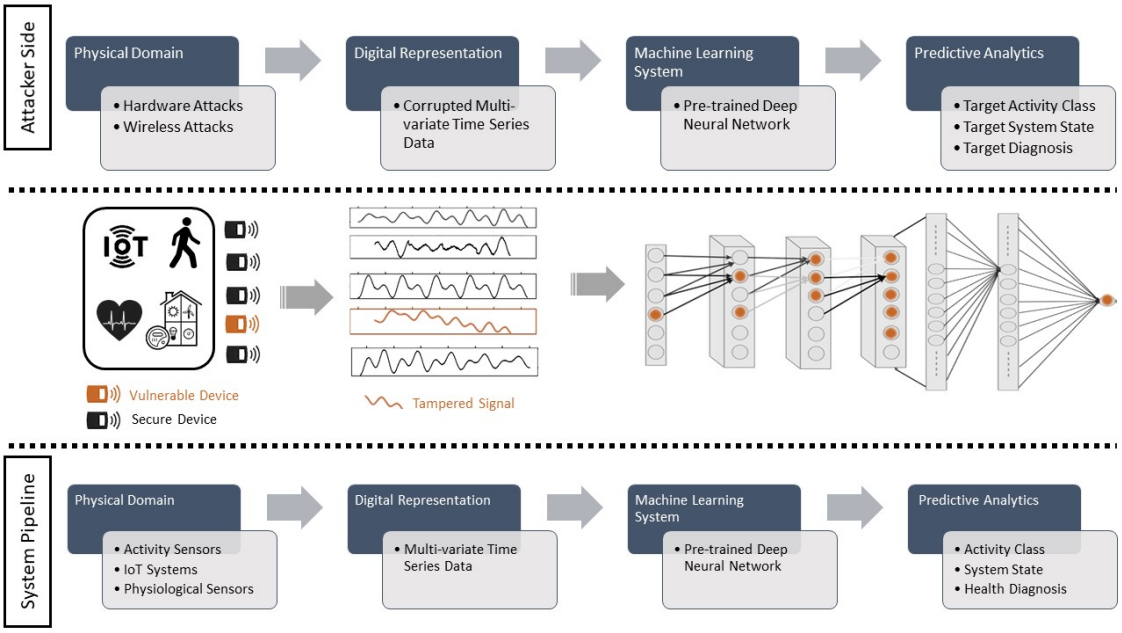


Figure 3.1 MTS-AdLens framework and the generic data processing pipeline for IoT and mobile applications. The system data processing pipeline (bottom row) is illustrated in parallel to the attacker's pipeline (top row) to attack the system. Data acquisition is the first step in the physical domain through IoT and mobile sensors. The sensors will record the activities sensed in the smart-home and generate time-series data to represent them digitally. In this step, the attacker can intervene through hardware and wireless attacks to tamper with the recorded data by injecting the adversarial attack created by MTS-AdLens framework. Digital representation of the recorded data is the second step. This is achieved by generating multivariate time-series data. In this step, the attacker employs MTS-AdLens framework to create the adversarial attack. The next step is to feed the adversarial example to a pre-trained deep neural network for time-series data, which represents the black-box model for MTS-AdLens framework. The proposed framework does not depend on the deployment location of the DL (local, edge or cloud).Finally, predictive analytics task is executed using the deep model to produce predicted output labels. However, due to adversarial input, deep learning model may make wrong predictions as desired by the attacker.

inference stage to improve the adversarial robustness of these models.

3.2.1 Unconstrained Black-box Attacks

Recall that black-box attacks do not assume any knowledge about the target deep model. However, the attacker has the ability of querying the target model to get the predicted label

for any candidate input time-series example. The data collected from querying the target model is used to create a *proxy* deep model to mimic the behavior of the target model. To create black-box adversarial attacks for IoT and mobile systems, we can employ FGSM/PGD and DeepFool methods to construct adversarial examples.

To create the *proxy* deep model, the attacker needs to use the data generated by the target system. This can be done in one or more of the following ways:

- Eavesdropping the target system for a given time to record all the data traces. Therefore, the attacker will be using the data that is seen by the target deep model. This helps the proxy deep model to better mimic the behavior of the target deep model.
- Employ synthetic data to augment the acquired data. We have used two different techniques for this purpose: 1) The dynamic time warping barycentric averaging technique (Hassan Ismail Fawaz et al., 2018) creates new time-series signals by averaging a set of random signals in a dynamic time warping space. This technique allows us to synthesize additional time-series signals that are very similar to the original ones. 2) Create additional data by running the FGSM/PGD algorithm to create adversarial examples using minimal perturbation that are assumed to belong to the same semantic label space as the original time-series signal.
- The attacker can potentially acquire similar sensors and create the training data for the proxy model.

3.2.2 Real-world Constraints in IoT and Mobile Systems

Real-world constraints. Since the target deep model for IoT and mobile systems rely on synchronous sensors, it is critically important to account for the limitations imposed by the hardware on plausible attack strategies. We list three important constraints for a sensor-enabled mobile system.

- *Constrained computational resources:* Sensors are basic hardware designed to meet specific goals and lack computational resources to execute sophisticated algorithms.

- *Hard traceability:* Sensors and wearable systems rely on sparse devices that are not centrally controlled. These devices are expected to operate dynamically and function autonomously. Their main objective is to acquire signals and send them to a central data acquisition system (Lara and Labrador, 2012). As a result, we cannot map the collected time-series data to the sensors that generated it.

- *Real-time execution:* The overall system needs to execute in real-time to meet the functional requirements of its applications. Therefore, practical security attacks should satisfy these real-time needs to delay induction and possible ensuing investigations.

Practical hardware threats. Let us consider a mobile system enabled by multiple synchronous sensors. The goal of the attacker is to target all those sensors to be able to tamper them. Due to limited hardware resources and open connectivity of sensors, the attacker can employ several hardware attack approaches to tamper with the time-series signals. Side-channel attacks (Nawir et al., 2016) can be potentially used to monitor sensors and to learn how to infiltrate and tamper with their acquisition. Fault injection attacks (Zabib et al., 2017) may allow the attacker to result in failure of sensors and gain access to them. Since sensors are mostly wireless, another potential threat is the ability of shadow attack, or jam one or more sensors, and masquerade his own emitters as a legitimate acquisition device for the target system (Nawir et al., 2016).

One of the main constraints the attacker will face is that of real-time signal acquisition. To create an adversarial instance, the attacker needs to acquire the original input in order to tamper with it. Therefore, methods to create adversarial instances (e.g., FGSM) can solely target systems with data storage. In this case, the attacker can try to maliciously get access to it and copy the acquired time-series data to construct corresponding adversarial

instances to be sent to the system. If this constraint exists, the attacker won't be able to maneuver around it for creating those adversarial instances. To overcome this challenge, universal adversarial attack approach can be employed. This technique will permit the attacker to create the pattern that needs to be embedded in the sensors. Once the pattern is applied to the recorded signal, the resulting time-series data is most likely to be incorrectly classified. Additionally, this technique allows the attacker to overcome the second obstacle: *sparsity of sensors*. Since the target systems rely on multiple sensors, it will be tedious or near-impossible for the attacker to maliciously gain access to all of them discretely. This constraint will limit the number of channels that can be perturbed. Therefore, from all the channels that produce one single multivariate time-series input example, the attacker will have access to only a small number of channels. A single input example is composed of synchronous channels acquired from multiple heterogeneous sensors.

3.2.3 Practical Attacks via Critical Channels Analysis

In our work, we show that even with the above-mentioned real-world constraints, it is possible to create an adversarial pattern that will result in most inputs to be mis-classified. In such cases, the attacker would not have a reason to infiltrate all sensors at the same time. The key insight to creating such adversarial patterns relies on identifying *critical channels* of the time-series data. Our fine-grained analysis of the inference behavior of the deep model showed that it relies on some channels more than others to predict the output labels. To compute the sorted order of different channels of the input $[X_1, X_2, \cdots, X_n]$, we propose an iterative *forward-search* algorithm.

Algorithm 1 shows the pseudo-code of creating adversarial attacks using critical channel analysis. First, data from the target model is collected as illustrated in 3.2.1, where the attacker will eavesdrop on the mobile system to record data seen by the target model and employ synthetic data to augment it (Line 1 and 2). Subsequently, the attacker labels the overall data acquired by querying the target model (Line 3) and splits the data into a training

Algorithm 1 MTS-AdLens Attack: Practical Attacks via Critical Channel Analysis

Input: $F(\theta, X)$, target deep model; Attack Strategy $\mathcal{AS}$.

1: Collect a subset of target data: $D_{sub} \leftarrow$`EavesDrop`(D)

2: Create proxy data: $D_{proxy} \leftarrow D_{sub} \oplus$`Augment`$(D_{sub})$

3: Create proxy labels: $L_{proxy} \leftarrow F(D_{proxy}, \theta)$

4: Divide proxy data and labels into training data D and validation data V

5: Train proxy model: $F(\theta_{proxy}, X) \leftarrow$`Train`$(D)$

6: $n \leftarrow$ number of channels in the time-series data

7: Initialize the list of sorted critical channels $\mathcal{C} = [\,]$

8: Initialize list of channels $\mathcal{L} = [1, 2, \cdots, n]$

9: $A_{full} \leftarrow$ Compute accuracy of proxy model on data V

10: **repeat**

11: Compute the score of each channel i in list $\mathcal{L}$ as follows:

a) Set all channels in $\mathcal{C}$ and channel i to ZERO for all time-series examples in V

b) $A_{partial} \leftarrow$ Compute accuracy of proxy model on modified data V

c) Score of channel $i \leftarrow A_{full} - A_{partial}$

12: Add the channel in $\mathcal{L}$ with highest score to $\mathcal{C}$ and remove it from $\mathcal{L}$

13: **until** channel list $\mathcal{L}$ is empty

14: Create adversarial attacks using strategy $\mathcal{AS}$, proxy model $F(\theta_{proxy}, X)$, and sorted critical channels $\mathcal{C}$ to meet practical constraints

set and validation set (Line 4). The proxy model is created using the training data (Line 5). To create practical adversarial attacks, the sorted order of critical channels is computed as follows. We start with an empty list of sorted critical channels $\mathcal{C}$. In each iteration, we identify the most critical channel from the channels not in $\mathcal{C}$ and add it to $\mathcal{C}$. To compute the critical score of each channel not in $\mathcal{C}$, we do the following. We set the candidate channel along with all the channels in $\mathcal{C}$ to ZERO and perform inference with the target model $F_\theta(X)$

on a set of validation examples (Line 11a and 11b). The drop in accuracy is considered the critical score and we select the channel with highest score to add to $\mathcal{C}$ (Line 12). The sorted order of critical channels of the proxy model are used to create practical adversarial attacks using a specific attack algorithm (e.g., FGSM/PGD or DeepFool) to meet the real-world constraints.

In our experiments, we have also found that depending on the architecture of the deep (CNN) model, two different models can have different sorted order for critical channels. However, the attacker can target critical channels to create practical attacks. Indeed, our experiments show that targeting critical channels improves the effectiveness of attack in degrading the accuracy of target model. An attack that perturbs a small number of most critical channels is more effective than an attack that perturbs a larger number of channels chosen at a random.

3.2.4 Adversarial Defense via Dynamic Ensembles

In this section, we propose a novel defense algorithm referred as *Dynamic Ensembles for Error Detection (DEED)* to improve the adversarial robustness of the target deep model. DEED method is deployed at the inference stage and NOT the training stage of the model. Therefore, the security of these mobile systems is independent of the deep model provider and DEED can be directly deployed at the level of the system. As mentioned above, most defense mechanisms (studied for vision tasks) rely on re-training the model to improve robustness. However, there are at least two practical challenges to employ re-training based methods for defense: 1) Due to the inherent complexity and ambiguity of time-series signals for human perception, prior methods can deteriorate the accuracy of trained deep models. Indeed, we demonstrate this phenomenon in our experiments; and 2) In many IoT and mobile applications (e.g., smart health), it may not be possible to re-train deep models due to privacy concerns. This is the key motivation for us to study a defense method that can be directly deployed at the inference stage.

Algorithm 2 MTS-AdLens Defense: Dynamic Ensembles for Error Detection (DEED) Algorithm

Input: $F_\theta(X)$, target deep model; $\mathcal{C}$, set of n critical channels in sorted order; X, input multivariate time-series example; K, number of channels for defense; MAX, the size of ensemble.

Output: classification error or not

1: Predict the output from target model: $\hat{y} = F(X, \theta)$
2: Specify an exponential dropout probability distribution for channels based on $\mathcal{C}$: $\mathcal{P}$=$(p_1, p_2, \cdots, p_n)$
3: **for** i=1 to MAX **do**
4: Select K channels via sampling from distribution $\mathcal{P}$
5: X' $\leftarrow$ Set the unselected channels from X to ZERO
6: Predict the output from target model: $\hat{y}_i = F(X', \theta)$
7: **end for**
8: $\hat{y}_{vote} \leftarrow$ Majority vote of all MAX predictions
9: If $\hat{y}_{vote} \neq \hat{y}$ **return** classification error

The DEED algorithm is inspired by real-world constraints for mobile systems mentioned in the above section and our vulnerability analysis of the deep models to identify the critical channels. Since the complexity of attack to simultaneously tamper all sensors increases with the number of sensors used, our DEED approach for defense exploits this key limitation to detect erroneous classification from the target deep model. As explained in the previous section, for error detection, DEED relies on the fact that each deep (CNN) model depends on a small number of critical channels to predict output labels. The error-detection scheme employs the same learning model with dropout probabilities on each channel. The channels are given exponential dropout probabilities such that the most reliable channels have less probability of being dropped out when compared to other channels. As the exponential prob-

ability is a heavy-tailed distribution, the channels that are the least reliable will be dropped more often than the channels that are the most reliable. The probability distribution is constructed in such a way that the most critical channels have higher chance to remain, while the least critical channels have higher chance to be dropped. If n channels are ordered based on their criticality as c_1 (least critical), c_2,$\cdots$, c_n (most critical), then dropout probability for c_i will be generated using the exponential probability density function $\frac{1}{n}e^{-\frac{1}{n}x}$, x being a random variable. For example, with five channels, the probability vector of the channels to be dropped from the input by DEED is [0.797, 0.155, 0.035, 0.006, 0.007]. This process will improve the robustness of the deep model to make correct classification decisions while there is a positive probability of disregarding tampered or perturbed channels. To increase the accuracy of DEED scheme, we employ a voting algorithm to infer the label of the input multivariate time-series example. If this inferred label via voting is different than the predicted activity label from the target deep model, then an error in classification is detected. Algorithm 2 shows the pseudo-code of DEED method.

3.3 Experimental Results

In this section, we first describe the experimental setup and then discuss the results of our proposed MTS-AdLens framework along different dimensions.

3.3.1 Experimental Setup

Datasets. To evaluate the effectiveness of our black-box attacks and DEED defense method, we employed five diverse real-world multivariate time-series datasets.

1. *WISDM* dataset was collected from 36 human subjects performing six different activities (Kwapisz, Weiss, and Moore, 2011). This dataset was collected using the accelerometers of Android smartphones and contains three channels.

2. *Opportunity Activity Recognition (OAR)* dataset was collected using sensors configured on human subjects who performed 18 different classes of daily activities and contains 107 channels (Roggen et al., 2010).

3. *PAMAP2 Physical Activity Monitoring* dataset was collected from 9 human subjects. These subjects performed 18 different classes of physical activities and contains using 36 channels (Reiss and Stricker, 2012a).

4. *DLA digital life-assistant* dataset where 24 ultrasound sensors are used to track a robot in a room (Dua and Graff, 2017).

5. *SH smart-home monitoring dataset* where multiple environment sensors have been used to monitor the different conditions in a given home setting (Dua and Graff, 2017).

We use WISDM for evaluating unconstrained black-box attacks; and employ the other datasets for evaluating constrained black-box attacks and DEED defense method due to the presence of a large number of channels.

CNN models for multivariate time-series classification. We employ the CNN model from (Ignatov, 2018) to experiment on WISDM dataset. The input of the CNN model is a time-series signal with a window-size of 200. The architecture includes one convolutional/max-pooling layer using 196 convolutional filters and one fully-connected layer using 1024 neurons. For the remaining datasets, we employed the CNN architecture proposed in (Jianbo Yang et al., 2015) for our model. This architecture employs three convolutional layers of which two are followed by a max-pooling layer. Finally, the feature maps output is fully-connected to the output layer of the CNN. The input of this CNN model is a signal with window-size of 30. We employ cross-entropy loss with l_2 regularization to learn weights via Adam optimizer.

Evaluation metrics. The effectiveness of an attack strategy $\mathcal{AS}$ ($\eta_{\mathcal{AS}}$) is the error rate of the model on the adversarial data (higher the better). We only consider the data that has been originally classified by the target model correctly. The effectiveness of defense strategy

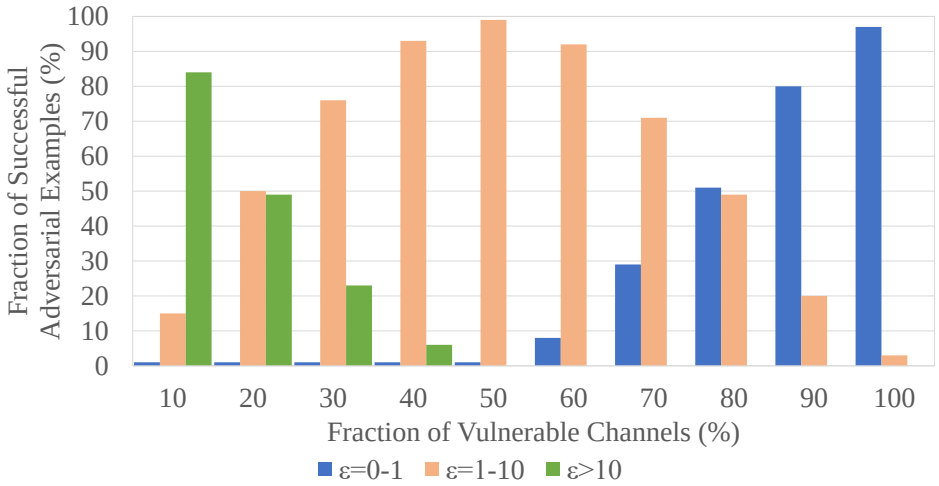


Figure 3.2 Histogram of the average minimal value of ϵ used to create a successful adversarial example using FGSM/PGD as a function of the fraction of vulnerable channels.

$\mathcal{DS}$ ($\eta_{\mathcal{DS}}$) is the recovered error-rate on the adversarial data (higher the better).

3.3.2 Results for Unconstrained Black-Box Attacks

For these experiments, we split the training data into two parts: one employed for training the target model and one employed for training the proxy model to enable attacks. We trained multiple CNNs that differ in architecture and parameters on the data created from querying the target model, and selected the model that performs best. used.

Results for FGSM/PGD attacks. Since ϵ (amount of perturbation) is a free variable, we can always create an adversarial example via FGSM/PGD for any input instance to fool the target model. Since sensors' signals are not human-perceptible, the attacker can create adversarial examples using the proxy model. Subsequently, it can be validated if the adversarial example fools the target model or not by querying it directly. If not, the attacker can modify ϵ until a successful adversarial example is created. Figure 3.2 shows the histogram of the average minimal ϵ value needed to create a successful adversarial example. It can be seen that ϵ value needed to succeed is inversely proportional to the fraction of vulnerable channels: a small number of vulnerable channels require large ϵ and vice versa.

Table 3.1 Transferability of $\eta_{\mathcal{AS}}$ from proxy model to the target model.

Proxy model	Target model
0.9	0.75
0.85	0.62
0.8	0.55
0.75	0.45
0.7	0.45

Results for universal attacks. Since the attacker has the ability to query the target model, we ran the DeepFool algorithm for several iterations to generate the universal adversarial pattern for a desired fooling rate. We compute $\eta_{\mathcal{AS}}$ to measure the efficiency of attack using this universal pattern. Table 3.1 shows the transferability rate of $\eta_{\mathcal{AS}}$ from the proxy model to the target model. This table shows that universal adversarial attacks are successful in black-box setting. Although $\eta_{\mathcal{AS}}$ is lower on the target model when compared to the proxy model, the attack still had a significant impact on the target model.

3.3.3 Results for Practical Attacks within Constraints

Recall that in practical monitoring systems, the attacker can only perturb a small number of channels. Therefore, we present the results of attacks with a varying fraction of vulnerable channels.

Results for FGSM attacks. Figure 3.3 shows the effectiveness of black-box attack via FGSM method for both constrained and unconstrained settings. We can see that attacks are capable of increasing the error rate of the target model up to 20 percent when the attacks are limited to 30 percent of the channels. Since OAR dataset consists of over 100 channels in contrast to 30 channels of PAMAP2, we see a gradual improvement in the effectiveness of attack with increasing vulnerable channels. We observe for sharp changes in the curves of SH and DLA due to their relatively small number of channels.

Results for PGD attacks. We can observe a high similarity of the attack performance between both input-specific attacks regarding the channel-access limitation. For the case of OAR and PAMAP2 data, we observe similar performance by both FGSM and PGD attacks in the accuracy drop. While for SH data, we can observe that PGD outperformed FGSM with a drop in accuracy of 80% for less than 40% channel access. This shows that input-specific attacks on time-series data are affected similarly by the limitation of channels access. While the limited access renders the attacks less effective, the attacker can still succeed in overcoming it using MTS-AdLens.

Figure 3.3 shows the importance of the proxy model in creating effective attacks. The figure shows the baseline of the same attack in a naive setting. This latter setting (Hassan Ismail Fawaz et al., 2018) transfers the attack directly to a black-box model with no prior tuning. When the attacker is unaware of such real-world constraints, the attacks are less effective. However, the attacker can still succeed in overcoming this limitation using our proposed MTS-AdLens framework. Through the comparison with the curve of the naive baseline, we observe that the framework through the proxy model can enhance the adversarial attack on black-box model in constrained settings. We conclude that the proposed framework using the proxy model can improve the effectiveness of adversarial attacks on black-box models in a constrained setting (i.e., access to limited channels).

Results for universal attacks. Figure 3.9 show the results for universal attacks for varying fraction of vulnerable channels, where vulnerable channels are picked based on the sorted order of critical channels. We also compare with a random selection of vulnerable channels to show the usefulness of critical channels analysis. The comparison to random selection aims to show that critical channels play a role in enhancing adversarial attacks. The proposed framework utilizes the identified critical channels to alter and create more effective adversarial examples in contrast to altering random channels. The figure illustrates that the proposed framework: 1) Create adversarial black-box attacks via selecting vulnerable channels are effective even in this constrained setting; and 2) Enhance the effectiveness of constrained

attacks via the selection of sorted critical channels in contrast of random selection. We conclude the high utility of critical channels analysis to create effective attacks by MTS-AdLens. These experimental results provide strong evidence that deep learning models for multivariate time-series data indeed rely on some channels more than others for the inference phase. This observation is important to thoroughly study the adversarial robustness of deep models for time-series data by identifying their vulnerabilities.

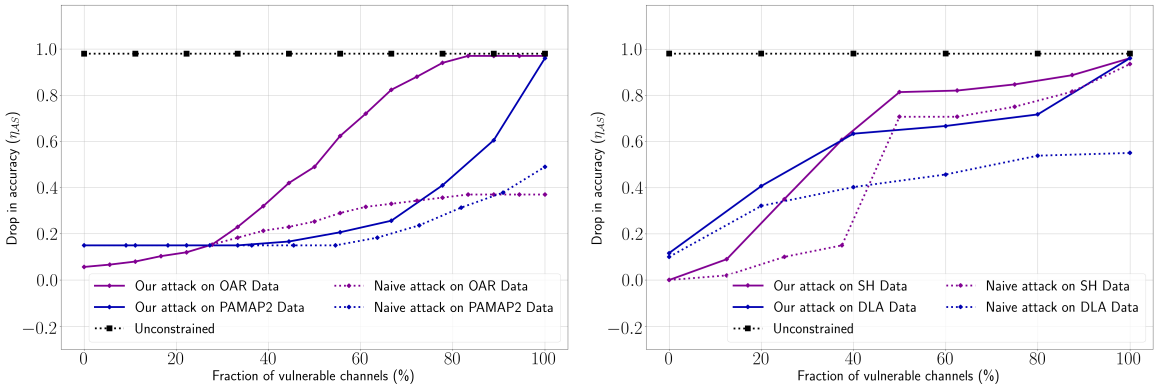


Figure 3.3 Results of black-box attacks using FGSM with unconstrained and constrained settings when compared to a naive attack baseline that does not use a proxy model.

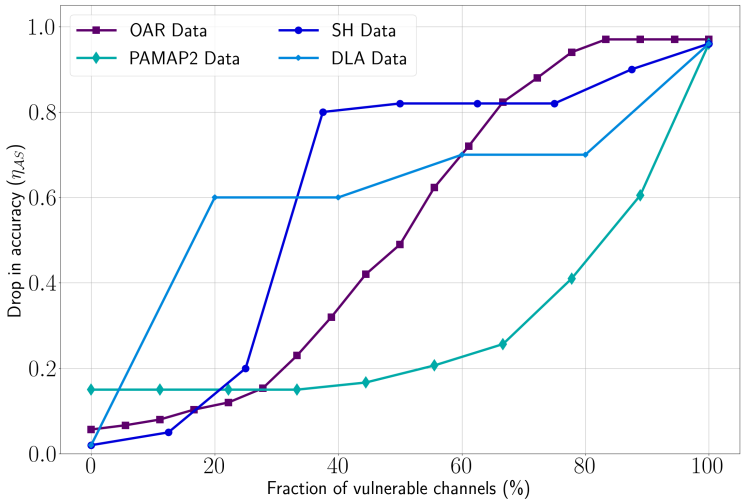


Figure 3.4 Results of black-box attacks using PGD with constrained settings.

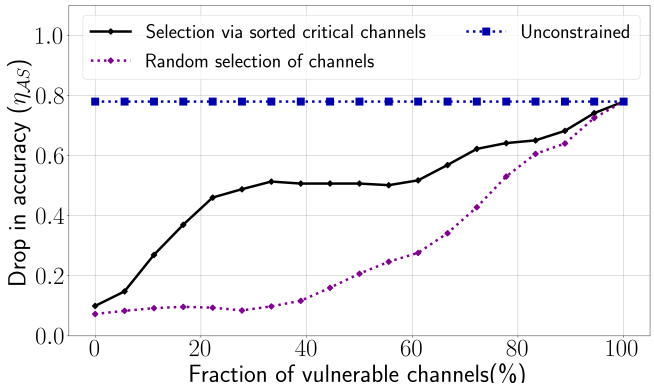


Figure 3.5 OAR Data

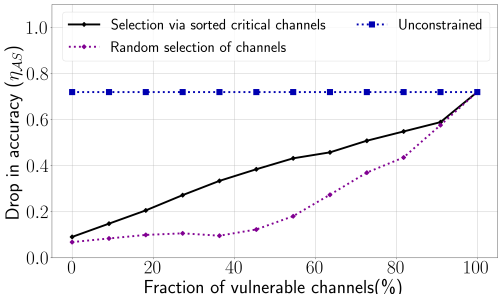


Figure 3.6 PAMAP2 Data

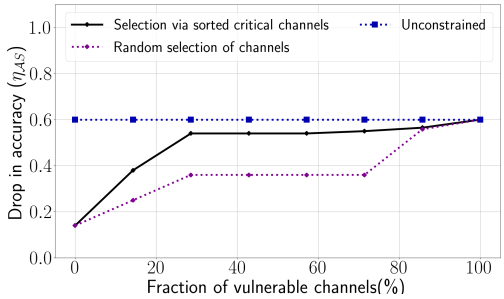


Figure 3.7 SH Data

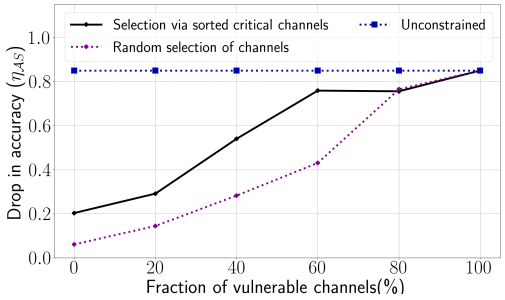


Figure 3.8 DLA Data

Figure 3.9 Results of universal attacks with constraints on vulnerable channels.

3.3.4 Results for Defense Mechanisms

Results for adversarial training. First, we test our hypothesis that the most popular defense method in image domain, namely, adversarial training will degrade the accuracy of deep model. Recall that one of the key challenges with adversarial time-series data is that

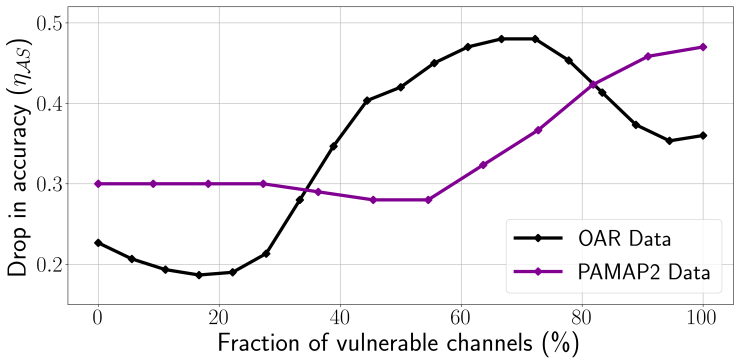


Figure 3.10 Performance of the classification model using adversarial training on black-box FGSM/PGD attacks.

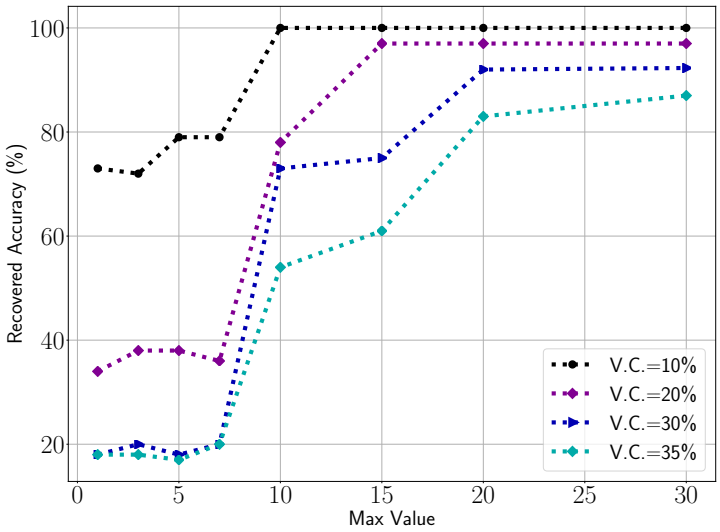


Figure 3.11 Performance of DEED algorithm as a function of MAX parameter (ensemble size) and the fraction of vulnerable channels (V.C.) allowed to create attacks.

human-eye-imperceptibility is not applicable. The defense technique relies on additional training data generated by FGSM and PGD attack (Hassan Ismail Fawaz et al., 2018). We have re-trained the target deep model using a fixed range for the parameter ϵ on WISDM dataset. Figure 3.10 shows the performance of the adversarially trained model for classifying FGSM/PGD generated adversarial inputs. The first observation is that the accuracy of the target deep model drops from **93%** to **78%** due to adversarial training. We explain this observation by the fact that time-series data have complex behavior: two signals that

are close in distance may belong to different true labels. Hence, some of the augmented adversarial examples are noisy/inconsistent and can potentially mislead deep models during the training phase. Additionally, the model is unstable in its performance when we observe the behavior on both datasets, and the drop in accuracy is relatively high to be considered as a functional defense. The instability of adversarial training to defend against adversarial attack is explained by the fact that the deep model is not trained for the possible scenarios, where limited channel attacks are effective. This result strongly corroborates our hypothesis.

Results for DEED defense method. Recall that DEED tries to detect whether the error in prediction by the target model is due to adversarial attacks in a constrained setting. The algorithm relies on the fact that under a black-box setting, the critical channels can be different from a proxy model that has been created to mimic its parameters and behavior. We tested with multiple values of MAX (ensemble size). Our results showed that error detection accuracy improves with increasing values of MAX and saturates at 30. This choice was concluded after running the DEED algorithm using different values of MAX. Figure 3.11 shows that the recovered accuracy is constant when reaching the value of MAX= 30. Hence, we used MAX=30 in all our DEED experiments. We set K (number of channels for defense) to be slightly more than the number of channels employed for attacks. For tuning the K parameter, this step is performed empirically by testing the full-range of different limited-channel attacks to choose the value at which the desired trade-off between security and efficiency is achieved for each dataset/application-setting. We performed experiments on both kinds of black-box attacks. Table 3.2 shows the error detection accuracy (fraction of detected errors that are correct) and Table 3.3 shows a comparison between the voting accuracy (accuracy of the predicted output $\hat{y}_{vote}$ from ensemble classifier in Algorithm 2 w.r.t the true label) of our proposed defensive algorithm and the accuracy of the learning model trained with adversarial data generated using FGSM/PGD . We make the following observations: 1) Error detection accuracy of DEED is very high when the fraction of vulnerable channels is small, but it drops gradually as the fraction of vulnerable channels employed for

Table 3.2 Results of DEED defense method in terms of error detection accuracy and accuracy of voting classifier.

Fraction of vulnerable channels (%)	Error detection accuracy(%)			
	PAMAP2	OAR	SH	DLA
0	100	100	100	100
10	96.8	94	95	97.2
20	82	88	85	90
30	78.6	82.5	85	89
35	61	62.5	84	80

attacks increases; and 2) Accuracy of the voting classifier is very high even for attacks with a relatively large fraction of vulnerable channels and exhibits robust behavior. The success of the proposed algorithm shows the importance of critical channels in deep learning models. While they create a vulnerability in accurate inference, the proposed framework employs them to improve the accuracy of the model, without the expense of re-training the model.

Finally, we run the DEED algorithm on an Android phone to quantify the cost of additional real-time overhead when compared to standard inference procedure. We employed a Samsung Galaxy S10 (Android 10, SoC Snapdragon 855) platform. The fraction of vulnerable channels are set to 20% for this experiment. In this setting, the deep learning model has already been shown to be vulnerable for limited channels attack. For example, for OAR data, the drop in accuracy $\eta_{\mathcal{AS}}$ is around 0.5. The runtime results for different datasets in real-time are shown in Figure 3.12. We use an ensemble size of 20. We make the following observations. 1) The overhead of DEED is proportional to the number of channels. They range between 15X and 17X with respect to the standard inference method. However, this overhead can be further reduced by exploiting the sparsity in the computation via zero channels; and 2) The recovered accuracy using DEED is shown to exceed 97% and justifies the additional overhead to improve robustness/security of deep models.

Table 3.3 Comparison of the Accuracy Results of DEED defense method vs Adversarial Training Defense.

	Adversarial Training accuracy(%)		DEED accuracy(%)	
Fraction of vulnerable channels (%)	PAMAP2	OAR	PAMAP2	OAR
0	70	78	100	100
10	70	80	100	100
20	70	81	97.04	98.02
30	71	75	92.3	92.5
35	72	72	87.4	87.8
	SH	DLA	SH	DLA
0	65	72	100	100
10	60	72	100	100
20	60	70	100	100
30	70	63	97.2	97.5
35	65	69	91.1	90.8

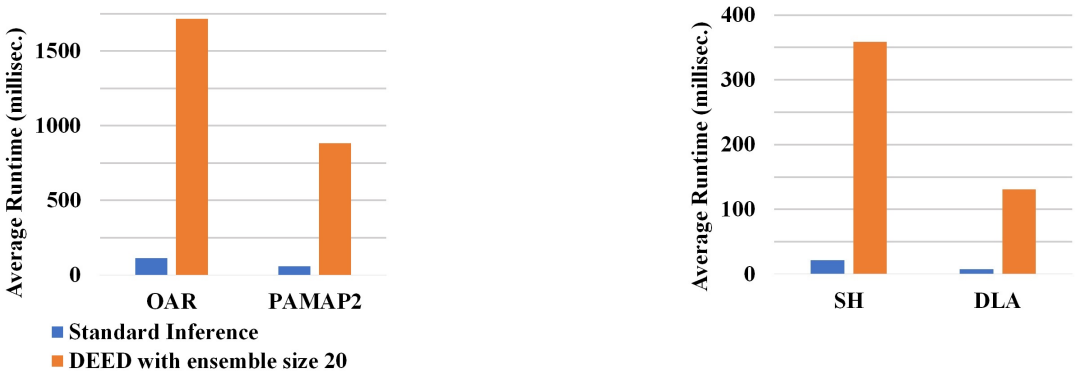


Figure 3.12 Runtime cost of DEED algorithm and standard inference without defense for the setting with 20% vulnerable channels on different datasets.

3.3.5 Summary of Experimental Findings

Key findings of our experimental evaluation include:

1. Black-box adversarial attacks (input-specific and universal attacks) are successful in fooling the deep model over multivariate time-series data for IoT and mobile applications without any real-world constraints.

2. With real-world constraints in the form of restricted access to channels for IoT and mobile systems, we showed that using critical channels analysis, we can develop successful attacks using the most critical channels meeting the constraints.

3. Using the critical channels and dynamic ensembles, our inference stage defense algorithm DEED achieved good results in detecting the constrained adversarial attacks. Additionally, the algorithm accurately predicted the original labels of adversarial signals.

3.4 Summary

In this chapter, we developed and evaluated a principled framework to analyze deep models for multivariate time-series classification in adversarial settings. Our comprehensive study shows that these deep learning methods are significantly vulnerable to adversarial attacks. We have exposed that these vulnerabilities remain existing even with limited access adversarial capacity, where the attacker cannot change all the channels of a multivariate time-series input at once. We have explained how the hardware architecture of sensors constrains the space of practically feasible adversarial attacks and how we can exploit the synchronous sensors architecture of monitoring systems to create effective and robust defense mechanisms.

CHAPTER FOUR

ADVERSARIAL FRAMEWORK WITH CERTIFIED ROBUSTNESS FOR TIME-SERIES DOMAIN VIA STATISTICAL FEATURES

T. Belkhouja and J. Doppa. "Adversarial Framework with Certified Robustness for Time-Series Data via Statistical Features". *Journal of Artificial Intelligence Research (JAIR), 73: 1435-1471*, 2022.

Originally published in the *Journal of Artificial Intelligence Research.*

Attributions:

T. Belkhouja has contributed to this work by formulating the problem setting, investigating the state of the art, formulating the theoretical contribution and the algorithmic solution, implementing the algorithm and running the required empirical analysis to highlight the performance improvement of the proposed solution compared to the state of the art.

J. Doppa has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

ADVERSARIAL FRAMEWORK WITH CERTIFIED ROBUSTNESS FOR TIME-SERIES DOMAIN VIA STATISTICAL FEATURES

In this chapter, we propose a novel framework referred to as **T***ime*-**S***eries* **A***ttacks via* **STAT***istical Features (TSA-STAT)* and provide certified bounds on robustness. TSA-STAT relies on three key ideas. First, we create adversarial examples by imposing *constraints on statistical features* of the clean time-series signal. This is inspired by the observation that time-series data are comprehensible using multiple statistical tools rather than the raw data (Ignatov, 2018; Christ, Kempa-Liehr, and Feindt, 2016; L. Ge and L.-J. Ge, 2016). The statistical constraints allow us to create valid adversarial examples that are much more similar to the original time-series signal when compared to l_p-norm constrained perturbations. Second, we employ *polynomial transformations* to create adversarial examples. For a given polynomial transformation with fixed parameters and an input time-series signal, we get an adversarial time-series as the output. We theoretically prove that polynomial transformations expand the space of valid adversarial examples over traditional additive perturbations, i.e., identify blind spots of additive perturbations. Our experiments demonstrate that polynomial transformation based attacks are more effective (in terms of successfully fooling time-series DNNs) than those based on additive perturbations. Third, to create attacks of different types, we solve an appropriate optimization problem to *identify the parameters of the polynomial transformation* via gradient descent. Certifiable robustness studies DNN classifiers whose prediction for any input X is verifiably constant within some neighborhood around X, e.g., l_p ball. We derive a certified bound for robustness of adversarial attacks using TSA-STAT. Our TSA-STAT framework and certification guarantees are applicable to DNNs for time-series domain with different network structures.

4.1 Challenges for time-series domain.

The standard l_p-norm based distance doesn't capture the unique characteristics (e.g., fast-pace oscillations, sharp peaks) and the appropriate notion of invariance for time-series signals. As a consequence, perturbations based on l_p-norm can lead to a time-series signal that semantically belongs to a different class-label. Indeed, our experiments demonstrate that small perturbations result in adversarial examples whose distance (l_2 and l_∞-norm) from the original time-series signal is greater than the distance between time-series signals from two different class labels. Therefore, there is a great need for studying adversarial methods focused on deep models for the time-series domain by exploiting the structure and unique characteristics of time-series signals.

4.2 The TSA-STAT Framework

In this section, we first provide a high-level overview of the TSA-STAT framework. Subsequently, we describe the key elements, and instantiate the framework to create white-box and black-box attacks.

Overview of TSA-STAT. Our framework creates targeted adversarial examples using polynomial transformations. For a given input time-series signal X, a target label y_{target}, a set of statistical features $\mathcal{S}$ and a DNN classifier F_θ, TSA-STAT generates adversarial examples using two key ideas: 1) *Constraints on the statistical features* to regularize the similarity of adversarial example X_{adv} to the original time-series X; and 2) A *polynomial transformation* that allows us to explore a larger space of adversarial examples over the traditional additive perturbations. Figure 4.1 provides a high-level overview of the TSA-STAT framework. The effectiveness of adversarial examples critically depends on the coefficients of the polynomial transformation. TSA-STAT solves an optimization problem over two different losses via gradient descent to find the parameters of the polynomial transformation. First, a statistical loss is employed to ensure that original time-series signal X and the generated adversarial

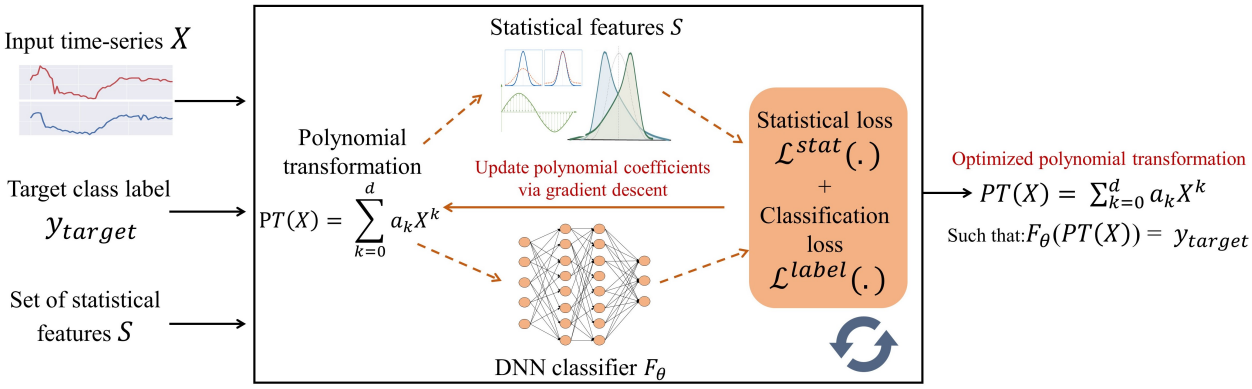


Figure 4.1 High-level overview of the TSA-STAT framework to create adversarial examples using optimized polynomial transformations. Given an input time-series signal X, a target label y_{target}, a DNN classifier F_θ, and a set of statistical features $\mathcal{S}$, TSA-STAT solves an optimization problem over two different losses to find the parameters of the polynomial transformation: 1) A statistical loss to ensure that original time-series signal X and the generated adversarial example X_{adv} are highly similar by imposing constraints on their statistical features; and 2) A classification loss to make sure that the DNN classifier F_θ classifies the generated adversarial example X_{adv} with the target class label y_{target}. The optimized polynomial transformation will take the time-series signal X as input and produce adversarial example X_{adv} as output.

example X_{adv} are highly similar by imposing constraints on their statistical features. Second, a classification loss to make sure that the DNN classifier F_θ classifies the generated adversarial example X_{adv} with the target class label y_{target}. The polynomial transformation with the optimized parameters will take the time-series signal X as input and produce adversarial example X_{adv} as output.

4.2.1 Key Elements

1) Statistical constraints. Time-series data is often analyzed using diverse statistical tools (Montgomery, Jennings, and Kulahci, 2015). Machine learning models have achieved good classification performance using statistical features of time-series data (Fulcher and Jones, 2014). These prior studies motivate us to use statistical features of time-series data to develop adversarial algorithms. We propose a new definition to create adversarial examples for time-series signals. Let $\mathcal{S}^m(X) = \{S_1(X), S_2(X), \cdots, S_m(X)\}$ be the set of statistical features of

a given input X (e.g., mean, standard deviation, kurtosis). We define an adversarial example X_{adv} derived from X as follows:

$$\begin{cases} \forall\ 1 \leq i \leq m,\ \|S_i(X_{adv}) - S_i(X)\|_\infty \leq \epsilon_i \\ \text{and}\ \ F_\theta(X) \neq F_\theta(X_{adv}) \end{cases} \tag{4.1}$$

where ϵ_i is the bound for the i^{th} statistical feature. Using this definition, we call to change the conventional l_p distance-based neighborhood-similarity to one based on statistical features for creating valid adversarial examples. We conjecture that this definition is better suited for adversarial examples in time-series domain. Indeed, our experiments strongly support this claim.

2) Polynomial transformation-based attacks. To explore larger and powerful space of valid adversarial examples when compared to traditional additive perturbations, we propose polynomial transformation based attacks. The aim of this approach is to find a transformation over the input space that creates effective adversarial attacks. This transformation considers the entire time-series input to decide the output for each channel and time-step of the adversarial example. Hence, we propose an adversarial transformation on the input time-series space. We define polynomial transformation $\mathcal{PT} : \mathbb{R}^{n \times T} \to \mathbb{R}^{n \times T}$ as follows:

$$X_{adv} = \mathcal{PT}(X) = \mathcal{PT}(X_{i,j}) \ \ \forall (i,j) \in [n] \times [T] \tag{4.2}$$

where $X \in \mathbb{R}^{n \times T}$ is the input time-series signal and X_{adv} is the corresponding adversarial example. The key idea is to create a threat model that does not require calling back the deep model for every new adversarial attack. Our goal is to preserve dependencies between features of the input space by having a transformation $\mathcal{PT}(\cdot)$ that depends on the input time-series X, unlike the standard additive perturbations. Inspired by power series (Drensky and Holtkamp, 2006), we approximate this transformation $\mathcal{PT}(\cdot)$ using a polynomial representation with a chosen degree d: $\mathcal{PT}(X) = \sum_{k=0}^{d} a_k\ X^k + \mathcal{O}(X^{d+1})$, where $a_k \in \mathbb{R}^{n \times T}$ denote the polynomial coefficients and $\mathcal{O}$ stands for Big O notation.

Theorem 1. *For a given input space* $\mathbb{R}^{n\times T}$ *and* $d \geq 1$*, polynomial transformations allow more candidate adversarial examples than additive perturbations in a constrained space. If* $X \in \mathbb{R}^{n\times T}$ *and* $\mathcal{PT} : X \rightarrow \sum_{k=0}^{d} a_k\ X^k$*, then* $\forall X_{adv}$ *s.t.* $\|S_i(X_{adv}) - S_i(X)\|_\infty \leq \epsilon_i$*:*

$$\left\{X_{adv} = \mathcal{PT}(X),\ \forall a_k\right\} \supsetneq \left\{X_{adv} = X + \delta,\ \forall \delta\right\}$$

, $S_i \in \mathcal{S}^m(X) \bigcup Identity.$

The above theorem states that polynomial transformations expand the space of valid adversarial examples and identify blind spots of additive perturbations. In other words, the theorem explains that *some* of the adversarial examples created using polynomial transformations are not possible using standard additive perturbations. We show through the proof provided in **Appendix A** that an example created using a standard additive perturbation can be created by a polynomial transformation, however, the inverse is not always true. This theorem motivates the use polynomial transformations within the TSA-STAT framework instead of additive perturbations in order to uncover more adversarial examples.

3) Optimization based adversarial attacks. To create powerful adversarial examples to fool the deep model $F_\theta(X)$, we need to find optimized coefficients a_k, $\forall$ k=0 to d, of the polynomial transformation $\mathcal{PT}(X)$. Our approach is based on minimizing a *loss function* $\mathcal{L}$ using gradient descent that **a)** Enforces an input signal X to be mis-classified to a target class y_{target} (different from true class label $y^* \in Y$); and **b)** Preserves close proximity to statistical features in the given set $\mathcal{S}^m$.

Classification loss. To achieve the mis-classification goal, we employ the formulation of (Carlini and D. Wagner, 2017) to define a loss function:

$$\mathcal{L}^{label}(\{a_k\}, X) = \max\left[\max_{y\neq y_{target}}\left(\mathcal{Z}_y\left(\sum_{k=0}^{d} a_k\ X^k\right)\right) - \mathcal{Z}_{y_{target}}\left(\sum_{k=0}^{d} a_k\ X^k\right),\ \rho\right] \quad (4.3)$$

where $\rho < 0$. This loss function will ensure that the adversarial example will be moving towards the space where it will be classified by the DNN as class y_{target} with a confidence $|\rho|$ using the output of the pre-softmax layer $\{\mathcal{Z}_y\}_{y\in Y}$.

Statistical loss. To satisfy the constraints on statistical features of the set $\mathcal{S}^m$, we propose another loss function. This loss function overcomes the impractical use of projection functions on the statistical feature space.

$$\mathcal{L}^{stat}(\{a_k\}, X, \mathcal{S}^m) \triangleq \sum_{S_i \in \mathcal{S}^m} \|S_i(\sum_{k=0}^{d} a_k \ X^k) - S_i(X)\|_\infty \tag{4.4}$$

Combined loss. The final loss function $\mathcal{L}$ that we want to minimize to obtain coefficients a_k of the polynomial transformation $\mathcal{PT}(\cdot)$ is as follows:

$$\mathcal{L}(\{a_k\}, X, \mathcal{S}^m) = \beta_{label} \times \mathcal{L}^{label}(\{a_k\}, X) + \beta_{stat} \times \mathcal{L}^{stat}(\{a_k\}, X, \mathcal{S}^m) \tag{★}$$

where β_{label} and β_{stat} are hyper-parameters that can be used to change the trade-off between the adversarial classification loss $\mathcal{L}^{label}$ and the statistical loss $\mathcal{L}^{stat}$. We note that our experiments showed good results with the simple configuration of β_{label}=1 and β_{stat}=1.

4.2.2 Instantiations of TSA-STAT

White-box setting. Our goal is to create targeted adversarial attacks on a classifier F_θ. Adversarial transformation X_{adv} for a single-instance X: $X_{adv} = \mathcal{PT}_{y_{target}}(X) = \sum_{k=0}^{d} a_k \ X^k$ s.t.:

$$\begin{cases} \|S_i(X_{adv}) - S_i(X)\|_\infty \leq \epsilon_i \ \forall S_i \in \mathcal{S}^m \\ F_\theta(X_{adv}) = y_{target} \end{cases}$$

where y_{target} is the target class-label of the attack.

We employ gradient descent based optimizer to minimize the loss function in Equation ★ over $\{a_k\}_{0 \leq k \leq d}$, where d is the polynomial degree for $\mathcal{PT}(\cdot)$. The parameter ρ introduced in Equation 4.3 plays an important role here. ρ will push gradient descent to minimize mainly the second term when the first one plateaus at ρ first. Otherwise, the gradient can minimize the general loss function by pushing $\mathcal{L}^{label}(\{a_k\}, X)$ to $-\infty$, which is counter-productive for our goal.

We can also extend this procedure to create adversarial examples under universal perturbations. A universal perturbation generates a single transformation that is applicable for any input $X \in \mathbb{R}^{n \times T}$. We introduce a targeted universal attack in this setting as:

$$X_{adv} = \mathcal{PT}_{y_{target}}(X) = \sum_{k=0}^{d} a_k \; X^k \text{ s.t. } \mathcal{PT}(F(X_{adv}) = y_{target}) > (1 - e_t) \tag{4.5}$$

where e_t represents the error probability of creating an adversarial example that F_θ would classify it with label $y \neq y_{target}$. Our proposed algorithm analyzes a given set of inputs to find coefficients $\{a_k\}_{0 \leq k \leq d}$ that would push image of multiple inputs $\mathcal{TF}(X) = \sum_{k=0}^{d} a_k \; X^k$ to the decision boundary of a target class-label y_{target} defined by the classifier F_θ. As the algorithms for both universal attack and instance-specific attack are similar and follow the same general steps, we present the universal attack algorithm of TSA-STAT in Algorithm 3. The instance-specific attack is a *special-case* of the universal attack: Since the universal algorithm generates a single polynomial transformation that is applicable for *any* time-series X, the instance-specific transformation is just applicable for a *single* time-series X. Algorithm 3 can degenerate to the case of instance-specific attack by changing the value of l (the number of time-series inputs) in Line 2 to the value of 1 and optimize over only one time-series X.

Black-box setting. Black-box attacks are adversarial examples that are created with no knowledge about the target deep model parameters θ. In the best scenario, the attacker has the ability to query the target model to get the predicted label for any input time-series X. This allows the creation of a proxy deep model to mimic the behavior of the target model. This technique can be more effective when a target scenario is well-defined (Tramer et al., 2020; Papernot, McDaniel, I. Goodfellow, et al., 2017). For the instantiation of TSA-STAT, we consider the general case where we do not query the black-box target DL. We create adversarial examples using optimized transformations as in white-box setting and prove through experimental results that the same transformations generalize to fool other black-box deep learning models.

Algorithm 3 Optimized universal adversarial transformation

Input: A set of l inputs $\{X_i\}_{i=1}^{l}$; d, maximum degree; y_{target}, target class; F_θ, target model; $\mathcal{S}^m$, statistical feature set; η, learning rate

Output: $\{a_k^y\}_{0\leq k\leq d,\ y\in Y}$

Random initialization of $\{a_k^y\}$.
for i=1 to l **do**
  if $F_\theta(X_i) \neq y_{target}$ **then**
    $\hat{y} \leftarrow F_\theta(X_i)$
    $\delta \leftarrow \nabla_{\{a_k^{\hat{y}}\}} \mathfrak{L}(\{a_k^{\hat{y}}\}, X_i, \mathcal{S}^m) \ \ \forall k$
    $\forall k : \{a_k^{\hat{y}}\} \leftarrow \{a_k^{\hat{y}}\} - \eta \times \delta$
  end if
end for
return $\{a_k^y\}_{0\leq k\leq d,\ y\in Y}$

4.3 Certified Bounds for Adversarial Robustness of TSA-STAT

In this section, we propose a novel certification approach for adversarial robustness of the TSA-STAT framework. Given a time-series input $X \in \mathbb{R}^{n\times T}$ and a classifier F_θ, our overall goal is to provide a certification bound δ on the $\|\cdot\|_\infty$ over the statistical features $\mathcal{S}^m(X)$ of the time-series signal X. Traditionally, the certification bound is a constant δ that constrains the distance between an input X and a perturbed version X_{adv}=$X+n_P$ (n_P is a multi-variate noise) as shown in Figure 4.2(a). Using TSA-STAT, our goal is to derive a certification bound δ that constraints the difference between the statistical features of a given time-series input X and a perturbed version $X + n_P$ as shown in Figure 4.2(b). This bound will guarantee the robustness of classifier F_θ in predicting $F_\theta(X_{adv}) = F_\theta(X)$ for any adversarial time-series X_{adv} such that $\sum_{S_i\in\mathcal{S}^m} \|S_i(X_{adv}) - S_i(X)\|_\infty \leq \delta$, where S_i is a statistical feature (e.g., a vector of mean values, one for each time-series channel) and ∞ norm takes the maximum of

the difference between statistical feature values for each channel separately (e.g., maximum of the difference between mean for each channel separately).

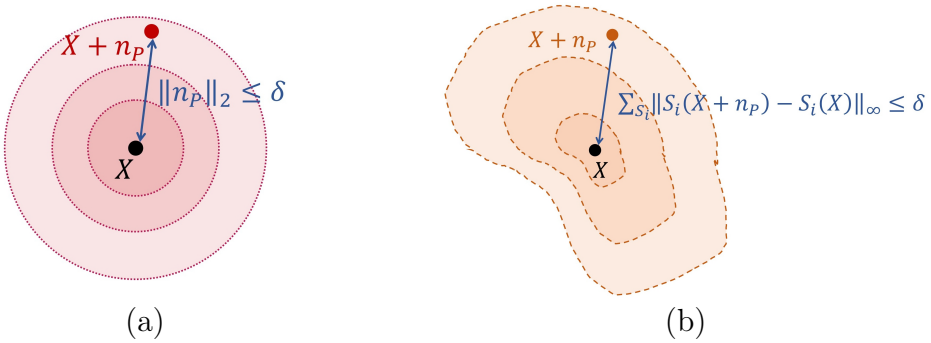


Figure 4.2 Conceptual illustration of the perturbation region of an input X with respect to noise n_P as considered by (a) Standard l_2 norm where δ is a constant representing the Euclidean distance between X and $X + n_P$; and (b) Statistical constraints as considered by TSA-STAT, where δ is a constant representing the cumulative sum of the maximum difference between statistical features computed over X and $X + n_P$ for each time-series channel separately. S_i represents one statistical feature (e.g., mean), and $S_i(X)$ and $S_i(X + n_P)$ represent the vector of values for a given statistical feature, one for each time-series channel (e.g., a vector of mean values for each channel).

As explained in Section 3, there are two families for certification methods, namely, exact and conservative. It has been shown that exact certification approaches do not scale well with the network in question (Cohen, Rosenfeld, and J Zico Kolter, 2019). Hence, we propose a certification algorithm that belongs to the conservative family. Our aim is to provide a bound that asserts that the prediction of $F_\theta(X)$ remains unchanged for any adversarial instance X_{adv} such that $\|S_i(X_{adv}) - S_i(X)\|_\infty$ is bounded by δ. State-of-the-art methods such as Gaussian smoothing (Cohen, Rosenfeld, and J Zico Kolter, 2019) rely on the Euclidean distance to measure the similarity between the original input and its adversarial example. Since TSA-STAT investigates statistical features of time-series for similarity purposes, the l_2 bounds derived by prior work are not sufficient to cover time-series adversarial examples. The certification provided in prior work cannot be extended to assess the robustness of DNNs for time-series domain as TSA-STAT relies on complex statistical features. To overcome this challenge, we propose a new robustness certification approach for TSA-STAT that is well-

suited for time-series domain by bounding the statistical features. Intuitively, we aim to

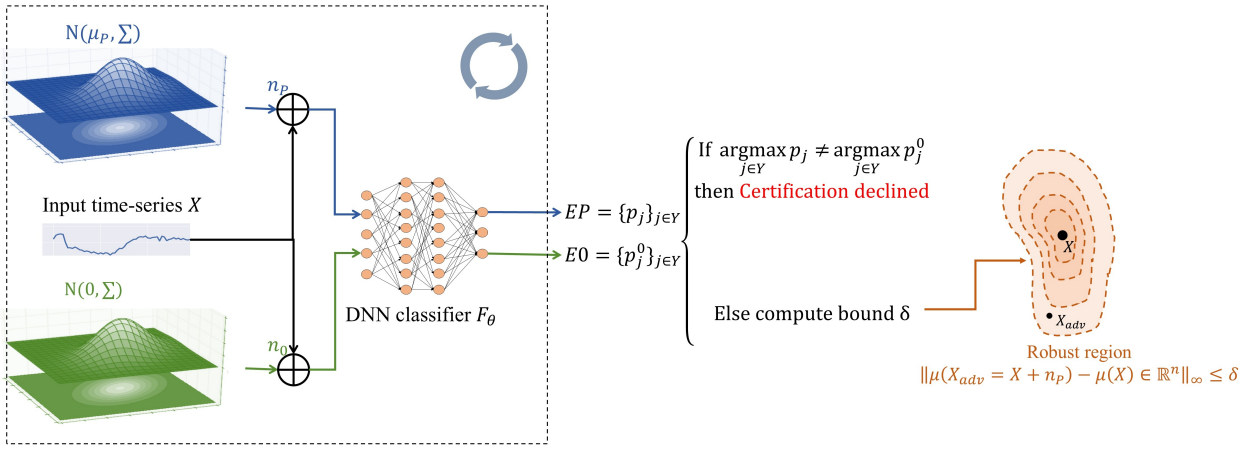


Figure 4.3 High-level illustration of the TSA-STAT certification approach to estimate the statistical perturbation space of a given time-series input X where the classifier F_θ is robust. This illustration is for mean only. For a given number of iterations, we repeatedly generate perturbations n_P and n_0 and add them to the time-series X to assess the robustness of classifier F_θ using the mean statistical feature. $n_P \sim \mathcal{N}(\mu_P, \cdot)$ is generated to mimic the perturbation that can affect the input time-series signal by producing EP (probability distribution over candidate class labels) that is used to characterize the robustness of classifier F_θ for predicting the same label for time-series X. $n_0 \sim \mathcal{N}(0, \cdot)$ is generated as an arbitrary noise that does not affect the mean vector (statistical feature for each channel) of X and produces $E0$ (probability distribution over candidate class labels) needed for the computation of the certification bound δ. Once δ is estimated, TSA-STAT guarantees the robustness of classifier F_θ for predicting $F_\theta(X_{adv})$=$F_\theta(X)$ for any X_{adv} such that $\|\mu(X_{adv}) - \mu(X)\|_\infty \leq \delta$, where $\mu(.)$ is the vector of mean values, one for each time-series channel separately and ∞ norm takes the maximum value of a given vector.

provide a certification for an input X that considers the statistical feature space of the time-series signal X. TSA-STAT's certification relies on adding random multi-variate perturbation n_P to quantify the robustness of the classifier on the surrounding region using statistical constraints as shown in Figure 4.3.

1. If the classifier F_θ predicts a class-label on the perturbation X_{adv} which differs from the prediction on the original time-series signal X, then the classifier is prone to adversarial attacks on the input.

2. If the classifier F_θ yields the correct classification in spite of all the perturbations, then it is easy to say that the classifier is robust against any perturbation (represented by n_P) on the time-series input X.

3. If the classifier F_θ yields the correct classification on most perturbation cases, then we develop an algorithmic approach to compute the conditions that n_P must satisfy in order to not affect the classifier's prediction. Therefore, the certification bound can be deduced.

Our certification study relies on Rényi Divergence (Van Erven and Harremos, 2014). Rényi divergence is a generalization of the well-known Kullback-Leibler (KL) divergence (Van Erven and Harremos, 2014). For a positive order $\alpha \neq 1$ and two probability distributions EP=$(p_1, \cdots, p_k)$ and $E0$=$(p_1^0, \cdots, p_k^0)$, which are estimated in our case, the Rényi divergence is defined as:

$$D_\alpha(EP \| E0) = \frac{1}{\alpha - 1}\ ln\left(\sum_{i=1}^{k} p_i^\alpha \cdot (p_i^0)^{1-\alpha}\right) \tag{4.6}$$

For the purpose of this work, we define the estimated probability distribution EP as the empirical probabilities p_i that the class i is predicted by F_θ on $X + n_P$ (n_P being a random perturbation). Our TSA-STAT framework is general to handle multivariate time-series data. Hence, we define a multi-variate Gaussian distribution $\mathcal{N}(\mu, \sum)$ characterized by a mean vector μ and a covariance matrix $\sum$ to generate n_P. To compute the divergence between multi-variate Gaussian distributions, the expression is provided by the following Lemma (Gil, Alajaji, and Linder, 2013).

Lemma 1. *For two multivariate Gaussian distributions* $\mathcal{N}(\mu_1, \sum_1)$ *and* $\mathcal{N}(\mu_2, \sum_2)$*:*

$$D_\alpha(\mathcal{N}(\mu_1, \sum_1) \| \mathcal{N}(\mu_2, \sum_2)) = \frac{\alpha}{2}(\mu_1 - \mu_2)^T \sum_\alpha (\mu_1 - \mu_2) - \frac{1}{2(\alpha - 1)}\ ln \frac{|\sum_\alpha|}{|\sum_1|^{1-\alpha} |\sum_2|^\alpha}$$

, where $\sum_\alpha = \alpha \sum_1 + (1 - \alpha) \sum_2$.

Algorithm 4 TSA-STAT Certification Algorithm

Input: A multivariate time-series signal X, F_θ, DNN classifier; Y, the set of class labels

Parameters: μ_P, multivariate mean; $\sum$, covariance matrix; n, the number of iterations

Output: $\hat{y}$, predicted class label; δ, certification bound

1: **for** i=1 to MAX **do**

2: Generate $n_P \sim \mathcal{N}(\mu_P, \sum)$ and $n_0 \sim \mathcal{N}(0, \sum)$

3: Compute $\hat{y}^p(i)$=$F_\theta(X + n_P)$ and $\hat{y}^0(i)$=$F_\theta(X + n_0)$

4: **end for**

5: Estimate EP=$\{p_j = \frac{\sum_{i=1}^{MAX} \mathbb{I}[[\hat{y}^p(i)==j]]}{MAX}\}_{j \in Y}$

6: Estimate E0=$\{p_j^0 = \frac{\sum_{i=1}^{MAX} \mathbb{I}[[\hat{y}^0(i)==j]]}{MAX}\}_{j \in Y}$

7: **if** $arg\max_{j \in Y}\ p_j \neq arg\max_{j \in Y}\ p_j^0$ **then**

8: **return** `Certification declined`

9: **else if** $\max_{j \in Y}\ p_j$ equals 1 **then**

10: **return** predicted label $\hat{y} = arg\max_{j \in Y}\ p_j$ and certification bound $\delta = \|\mu_P\|_\infty$

11: **else**

12: Compute the upper bound:

$$\delta^2 = \max_{\alpha \neq 1} \frac{2}{\alpha \cdot \sum^{(S)}} \cdot \left(- ln\left(1 - p_{(1)} - p_{(2)} + 2\left(\frac{1}{2}\left(p_{(1)}^{1-\alpha} + p_{(2)}^{1-\alpha}\right)\right)^{\frac{1}{1-\alpha}}\right)\right)$$

13: **return** predicted label $\hat{y}$=$arg\max_{j \in Y}\ p_j$ and certification bound δ

14: **end if**

Lemma 1 provides the expression of the divergence using the parameters of the multivariate Gaussian distributions. Consequently, we use it to produce the following theorem to provide certification bound over the mean of the time-series input space for adversarial robustness of TSA-STAT. For this purpose, we require a second multivariate Gaussian distribution $\mathcal{N}(\cdot, \cdot)$ to estimate $E0$ and to compute the divergence provided in Lemma 1. Therefore, we use an

arbitrary distribution $\mathcal{N}(0, \sum)$ with a zero-vector mean. This way, the mean feature of the input time-series signal will not be disturbed. For a computationally-efficient derivation of the certification bound, we use the same covariance matirx $\sum$ as the multi-variate Gaussian distribution that generated n_P.

Theorem 2. *Let $X \in \mathbb{R}^{n \times T}$ be an input time-series signal. Let $n_P \sim \mathcal{N}(\mu_P \in \mathbb{R}^n, \sum)$ and $n_0 \sim \mathcal{N}(0, \sum)$. Given a classifier $F_\theta : \mathbb{R}^{n \times T} \rightarrow Y$ that produces a probability distribution $(p_1, \cdots, p_k)$ over k labels for $F_\theta(X + n_P)$ and another probability distribution $(p_1^0, \cdots, p_k^0)$ for $F_\theta(X + n_0)$. To guarantee that $arg\max_{p_i} \; p_i = arg\max_{p_i^0} \; p_i^0$, the following condition must be satisfied:*

$$\|\mu_P\|_\infty^2 \leq \max_{\alpha \neq 1} \frac{2}{\alpha \cdot \sum^{(S)}} \cdot \left(-ln \left(1 - p_{(1)} - p_{(2)} + 2 \left(\frac{1}{2} \left(p_{(1)}^{1-\alpha} + p_{(2)}^{1-\alpha} \right) \right)^{\frac{1}{1-\alpha}} \right) \right)$$

where $\|\mu_P\|_\infty$ is the maximum perturbation over the mean of each time-series channel and $\sum^{(S)}$ is the sum of all elements of $\sum$.

This new certification formulation is suitable for the time-series domain, as it takes into account the different channels of the time-series signal input and adversarial attacks using TSA-STAT explore a larger space of valid adversarial examples using statistical constraints and polynomial transformations.

To derive the certification bound for a given time-series signal $X \in \mathbb{R}^{n \times T}$ and a classifier F_θ, we employ two different noise distributions to generate two different noise samples that we denote $n_P \in \mathbb{R}^{n \times T}$ and $n_0 \in \mathbb{R}^{n \times T}$, where n is the number of channels and T is the window size of the time-series signal. $n_P \sim \mathcal{N}(\mu_P, \cdot)$ is generated to mimic the perturbation that can affect the input time-series signal by producing EP (probability distribution over candidate class labels) that is used to characterize the robustness of classifier F_θ for predicting the same label for time-series X. $n_0 \sim \mathcal{N}(0, \cdot)$ is generated as an arbitrary noise that does not affect the mean vector (statistical feature for each channel) of X and produces

$E0$ (probability distribution over candidate class labels) needed for the computation of the certification bound. If both perturbations result to the same classifier prediction, we compute the tolerable perturbation's upper bound $\delta = \max \|\mu_P\|_\infty$. As $\|\mu_P\|_\infty$ is upper-bounded by the RHS term of Theorem 2, the maximum value for $\|\mu_P\|_\infty$ is the RHS term.

The upper bound δ guarantees that for any noise n_P with a mean feature for each channel constrained by δ, the classifier's prediction is robust on the perturbed input $X + n_P$. In other words, following the formulation used in Equation 4.4, if for an adversarial time-series signal X_{adv} such that $\|S_i(X_{adv}) - S_i(X)\|_\infty \leq \delta$ where S_i is the statistical feature mean (a vector of mean values, one for each channel) and ∞ norm takes the maximum of the difference between mean values for each channel separately, then $F_\theta(X_{adv}) = F_\theta(X)$. We provide Algorithm 4 to automatically assess the robustness of a classifier F_θ on a given multivariate time-series signal X as input. To generalize this result for other statistical features, we provide Lemma 2. Both proofs are present in **Appendix A**.

Lemma 2. *If a certified bound δ has been generated for the mean of input time-series signal $X \in \mathbb{R}^{n \times T}$ and classifier F_θ, then certified bounds for other statistical/temporal features can be derived consequently.*

4.4 Experiments and Results

In this section, we discuss the experimental evaluation of TSA-STAT along different dimensions and compare it with prior methods.

4.4.1 Experimental Setup

Datasets. To evaluate the proposed TSA-STAT framework, we employed diverse uni-variate and multi-variate time series benchmark datasets (Bagnall et al., 2020; Dua and Graff, 2017; Kwapisz, Weiss, and Moore, 2011). Complete details are provided in Table 4.1. We employ the standard training/validation/testing splits from these benchmarks. Table 4.1 describes

each dataset employed in our evaluation: acronym to represent the dataset, the number of classes, and the dimensions of each input time-series signal.

Table 4.1 Description of different benchmark time-series datasets.

NAME	ACRONYM	CLASSES	INPUT SIZE ($n \times T$)
Chlorine Concentration	CC	3	1×166
Synthetic Control	SC	6	1× 30
Cylinder-Bell-Funnel	CBF	3	1×128
CricketX	CX	12	1×300
CricketY	CY	12	1×300
CricketZ	CZ	12	1×300
Human Activities and Postural Transitions	HAPT	12	6×200
WISDM	WD	6	3×200
Character Trajectories	ChT	20	3×182

Algorithmic setup. We employ three different 1D-CNN architectures –A_0, A_1, and A_2– to create three deep models as target DNN classifiers: WB for white-box setting, and $BB1$ and $BB2$ for the black-box setting respectively. WB is a model using A_0 to evaluate the adversarial attack under a white-box setting, and trained using clean training examples. BB_1 and BB_2 use the architectures A_1 and A_2 respectively to evaluate the black-box setting. The architecture information of the deep learning models are presented in Table 4.2. To further evaluate the effectiveness of attacks, we create models that are trained using augmented data from baselines attacks that are not specific to the image domain: Fast Gradient Sign method (FGS) (Kurakin, I. Goodfellow, and Bengio, 2016) that was used by (H Ismail Fawaz et al., 2019), Carlini & Wagner (CW) (Carlini and D. Wagner, 2017), and Projected Gradient Descent (PGD) (Aleksander Madry et al., 2017). Finally, we evaluate the performance of

Table 4.2 Details of DNN architectures. C: Convolutional layers, K: kernel size, P: max-pooling kernel size, and R: rectified linear layer.

	C	K	C	K	P	R	R
A_0	x	x	66	12	12	1024	x
A_1	x	x	20	12	2	512	x
A_2	100	5	50	5	4	200	100

adversarial examples from TSA-STAT on two RNN models. To effectively test the transferability of adversarial transformations, we only use the knowledge of WB model. We assume that the framework is unaware of all other deep models. For FGS and PGD algorithms, we employed a minimal perturbation factor ($\epsilon < 0.4$) for two main reasons. First, larger perturbations significantly degrade the overall performance of adversarial training. We also want to avoid the risk of leaking label information (Aleksander Madry et al., 2017). Second, while analyzing the datasets, we found that there are time-series signals from different classes that are separated by a distance less than what an l_p-norm bounded perturbation engenders. Therefore, l_p-norm bounded attacks will create adversarial examples that are inconsistent (i.e., examples for a semantically different class label) for adversarial training. For example, in the case of CC dataset, there are time-series signals from different classes with l_2-distances ≤ 0.3, while FGS's average perturbation is around 0.3 for $\epsilon = 0.3$. If we employ l_∞-distance, CW causes several signal perturbations with l_∞-norm ≥ 1.5 on HAPT dataset, whereas many time-series signals with different class labels have l_∞-distances < 1.5. We observed similar findings in most of the other datasets.

For TSA-STAT, we use β_{label}=1 and β_{stat}=1 for the loss function in Equation ★ in all our experiments. TSA-STAT's attack algorithm and adversarial training have both shown good performance with this simple configuration. Therefore, we chose not to fine-tune the hyper-parameters β_{label} and β_{stat} to avoid additional complexity. We use constraints over statistical features including mean, standard deviation, kurtosis, skewness, and root mean

square (Brockwell and Davis, 2016) of an input time-series signal. We explain the methodology that was used to select these statistical features below.

4.4.2 Selection of Statistical Features and Polynomial Transformation

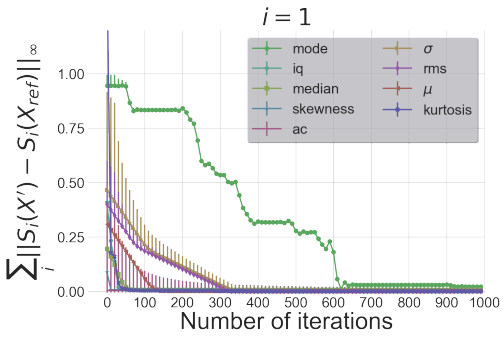


Figure 4.4 Convergence of different statistical constraints for $i \leq 1$

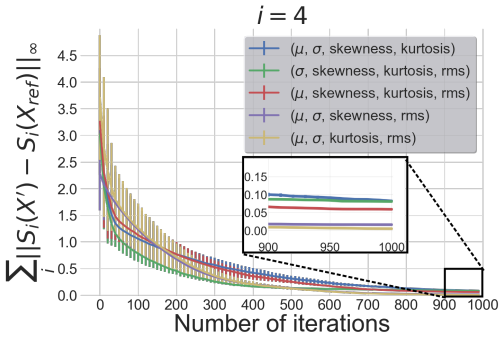


Figure 4.5 Convergence of different statistical constraints for $i \leq 4$

We initially started with the following statistical features of time-series signals: $\mathcal{S}^m$={Mean (μ), Standard deviation (σ), Median, Mode, Interquartile range (iq), Skewness, Kurtosis, Root mean square (rms), Auto-correlation (ac)}. To decide on the most appropriate subset to use for all our TSA-STAT experiments, we ran a convergence test on $\sum_i \|S_i(X') - S_i(X_{ref})\|_\infty$ using a subset of the data from WD. We note that the TSA-STAT framework can be used with both l_2 norm and l_∞ norm on the statistical features. For $X \in \mathbb{R}^{n \times T}$, we have one statistical feature for each channel, i.e., $S_i(X) \in \mathbb{R}^n$. Our goal from constraining $S_i(X)$ is to guarantee that for all the n channels, the value of the statistical feature is less than the given bound. Hence, the use of l_∞ norm is straightforward. However, any other norm can be used. To demonstrate the generality of TSA-STATE, we provide a comparison between l_2 norm and l_∞ norm on the statistical features in Figure 4.12. As Figure 4.4 illustrates, we ran the convergence test at first on each $S_i \in \mathcal{S}^m$ individually ($i \leq 1$). We eliminate the statistical feature S_i which does not converge properly in contrast to other statistical

constraints, and repeat the experiment each time by increasing i. Figure 4.5 illustrates the step at $i \leq 4$. We conclude from both Figures 4.4 and 4.5 that our approach empirically satisfies the ϵ_i bound presented in Equation 4.1. Hence, for all our TSA-STAT experiments, we choose $\mathcal{S}^m$={Mean μ, Standard deviation σ, Skewness, Root mean square} or $\mathcal{S}^m$={Mean μ, Standard deviation σ, Kurtosis, Root mean square}. We did not increase i further to avoid increasing the time-complexity of the proposed algorithm for negligible benefits. We have observed similar patterns for all other datasets. We note that it is not possible to use the basic PGD method to satisfy the constraint over $\sum_i \|S_i(X') - S_i(X_{ref})\|_\infty$: a projection function on the statistical feature space is not a straightforward projection as in the Euclidean space.

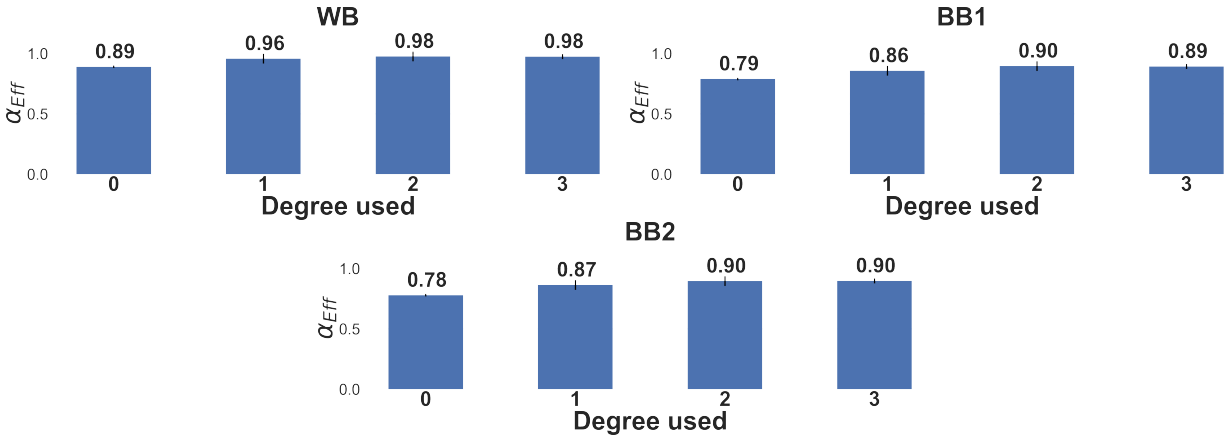


Figure 4.6 Performance of TSA-STAT based universal adversarial attacks using polynomial transformations with different degrees on multiple DNN models.

Regarding the degree of polynomial transformation for TSA-STAT, we employ d=1 and d=2 in all our experiments. Figure 4.6 shows the impact of different degrees used for the polynomial adversarial transformation when tested on the WD dataset noting that we observed similar patterns for all other datasets. Degree 0 corresponds to the standard constant δ additive perturbation. While the adversarial attack is still functional, degrees ≥ 1 showed improved effectiveness of adversarial attacks. Starting from degree 3, the attack's effectiveness did not increase significantly. To prevent increasing the time-complexity of the optimization method to find the coefficients of polynomial transformation, we chose degrees

d=1 and d=2 to evaluate our TSA-STAT framework.

4.4.3 Results and Discussion

Spatial distribution of TSA-STAT outputs. One of the claims of this work is that adversarial examples relying on l_p-norm bounds are not applicable for time-series domain. To evaluate this claim, we employ a t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008) technique to visualize the adversarial examples generated by TSA-STAT and PGD, an l_p-norm based attack. t-SNE provides a dimensionality reduction method that constructs a probability distribution for the high-dimensional samples to create a reduced feature space where similar instances are modeled by nearby points and dissimilar instances are modeled by distant points.

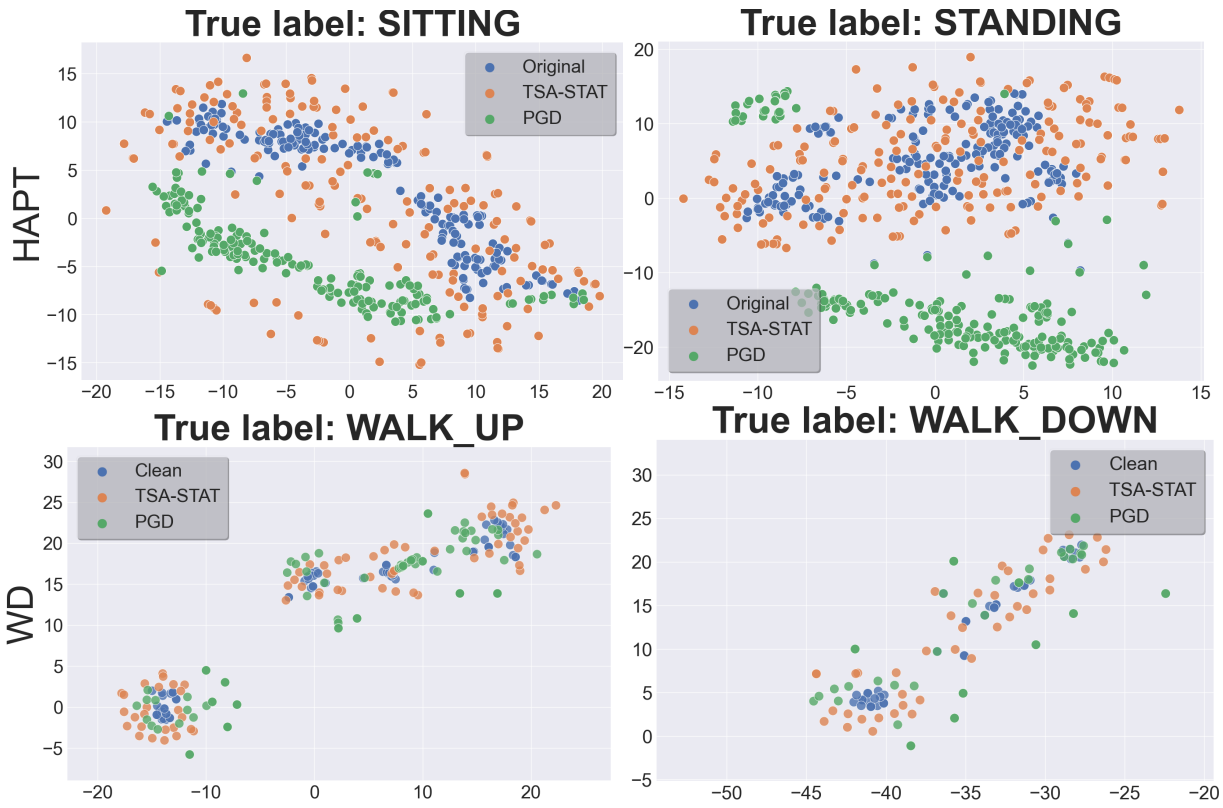


Figure 4.7 t-Distributed Stochastic Neighbor Embedding showing the distribution of natural and adversarial examples from TSA-STAT and PGD. Adversarial examples from TSA-STAT are more or equally similar to the original time-series input than PGD-based adversarial examples.

Figure 4.7 illustrates a representative example of the spatial distribution between same-class data of HAPT and WD, and their respective adversarial examples using TSA-STAT and PGD. We can clearly see that TSA-STAT succeeds in preserving the similarity between the original and adversarial example pairs, and in most cases, better than PGD.

Effectiveness of adversarial examples from TSA-STAT. All following experiments were repeated 10 times and we report the averaged results (variance was negligible). We have used the standard benchmark training, validation, and test split on the datasets. We implemented the TSA-STAT framework using TensorFlow (Abadi et al., 2016) and the baselines using the CleverHans library (Papernot, Faghri, et al., 2018). We employ ρ=-20 for $\mathfrak{L}^{label}$ in Equation 4.3. The choice was due to the observations made from Figures 4.8 and 4.9. A low value of ρ has worse performance on generalization to unseen data or black-box models. However, higher values of ρ slow down the convergence on each data point. Hence, we picked a confidence value of ρ at which the fooling rate performance did not increase significantly.

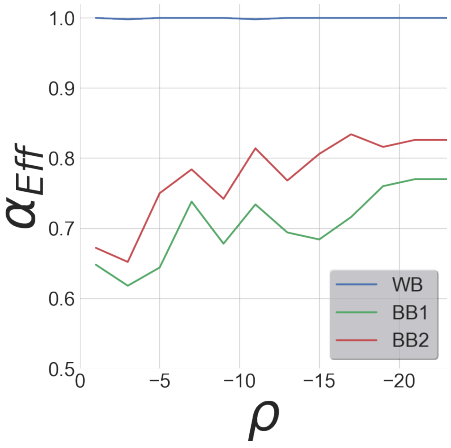


Figure 4.8 Performance of the fooling rate on a subset of WD dataset with a variable ρ for the instance-specific attack setting.

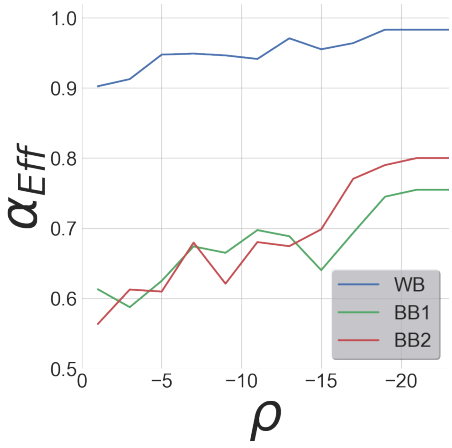


Figure 4.9 Performance of the fooling rate on a subset of WD dataset with a variable ρ for the universal attack setting.

Adversarial examples are generated for $\mathcal{L} < 0.1$ with a maximum of 5×10^3 iterations of gradient descent using the learning rate η=0.01. We construct a group of transformations

$\{\mathcal{PT}(X, y)\}_{y \in Y}$, one for each class y in Y. The transformation to be used depends on the initial output class-label predicted by the deep model for the given input X. Therefore, the universal transformation $\mathcal{PT}(X, y) = \sum_{k=0}^{d} a_k^y \, X^k$ will transform the inputs of the same class-label into adversarial outputs belonging to the target class-label. A targeted attack is a more sophisticated attack, which exposes the vulnerability of a DNN model better than an untargeted attack. From an attacker's perspective, having an attack model that allows choosing the target classification label of the adversarial example is better. Hence, we use targeted attacks for our experimental setup to show that TSA-STAT has the best opportunity for exploring time-series adversarial examples. We run the algorithm repeatedly on all the different class labels as targets. If the maximum iteration number is reached, we select the coefficients $\{a_k^y\}$ with the lowest corresponding loss.

We show the effectiveness of created adversarial examples for different settings (white-box, black-box etc.) to fool deep models for time-series domain. We evaluate TSA-STAT using the attack efficiency metric $\alpha_{Eff} \in [0, 1]$ over the created adversarial examples. α_{Eff} (higher means better attacks) measures the capability of targeted adversarial examples to fool a given DNN classifier F_θ to output the class-label y_{target} (i.e., targeted attacks). Figure 4.10 shows the results for instance-specific targeted attacks under white-box and black-box settings on different deep models. Figure 4.11 shows the results for universal attacks using TSA-STAT. Unlike instance-specific attacks, universal attacks are created by directly using the resulting polynomial transformation $\mathcal{PT}(\cdot)$. Recall that for black-box attacks, we do not query the target deep model at any phase. While comparing TSA-STAT based attack with the existing attacks using success rate provides an assessment about the performance of different attacks, it does not show which attack is stronger. To investigate the real performance of TSA-STAT, we show the effectiveness of the TSA-STAT based attack to fool deep models for time-series domain using both standard and adversarial training. If any baseline algorithm were better attacks than TSA-STAT, the adversarial training using that baseline will be robust towards TSA-STAT's attacks.

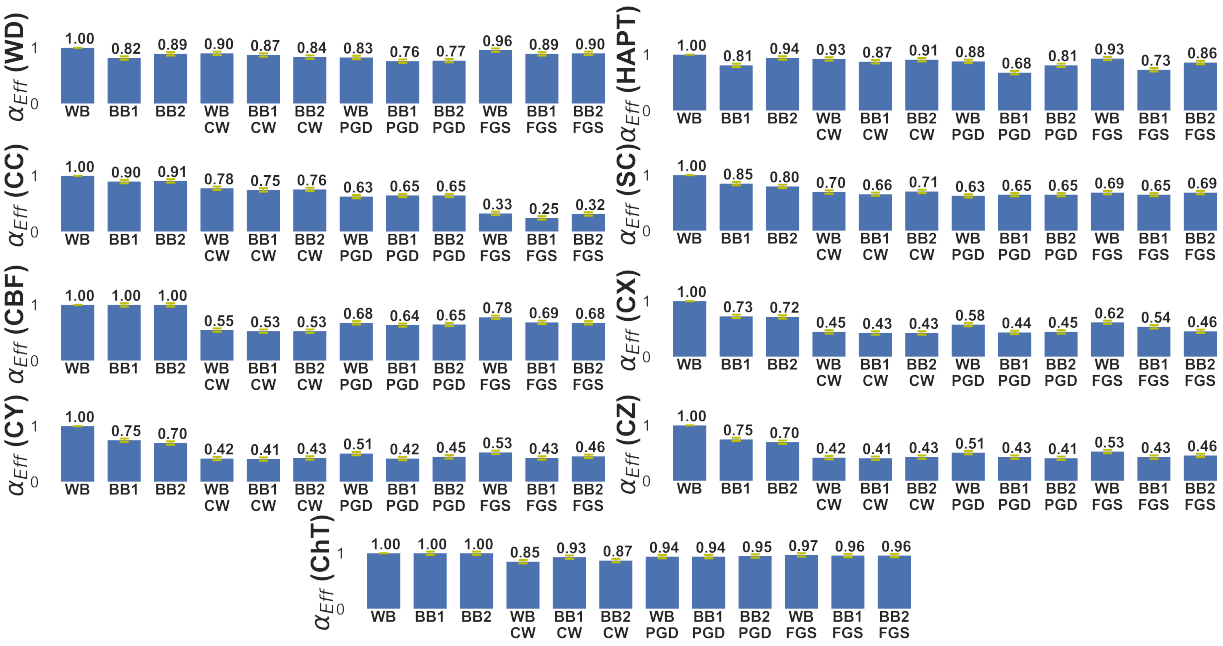


Figure 4.10 Results for TSA-STAT instance-specific adversarial examples on different deep models trained with clean data and adversarial training baselines.

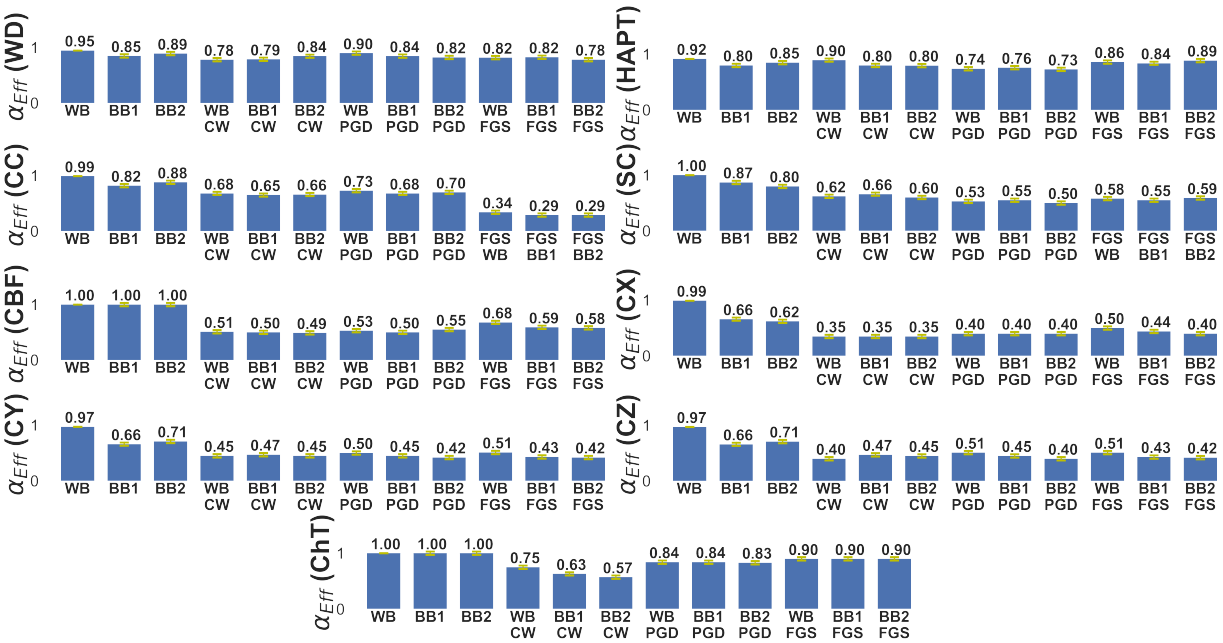


Figure 4.11 Results for TSA-STAT universal adversarial examples on different deep models trained with clean data and adversarial training baselines.

We can observe from both Figures 4.10 and 4.11 that on the multivariate WD and HAPT dataset, the fooling rate is good across all settings. Adversarial examples created

by optimized $\mathcal{PT}(\cdot)$ are highly effective as $\alpha_{Eff} \geq 0.7$ for most cases. For CC and SC datasets, we see a lower performance for TSA-STAT, essentially at the level of FGS on CC. We believe that this is due to the effect of l_p-bounded adversarial examples that mislead the deep models during adversarial training. Additionally, we show in Figure 4.12 that using l_2 norm or l_∞ norm on the statistical features has no difference in the general performance of TSA-STAT based attacks. Finally, Figure 4.13 shows the results of different deep models

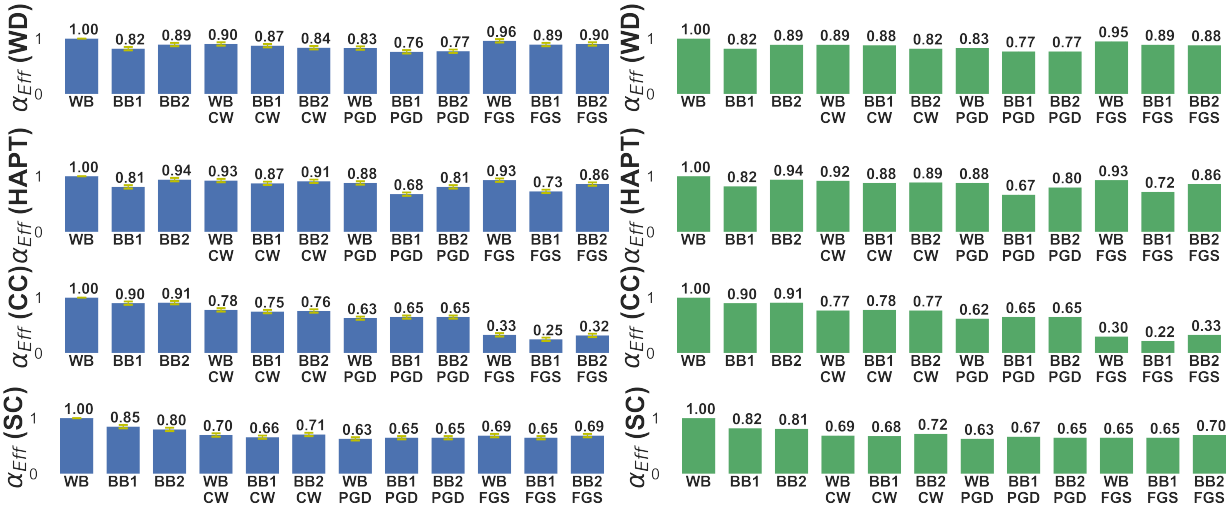


Figure 4.12 Results for TSA-STAT instance-specific adversarial examples on different deep models trained with clean data and adversarial training baselines using l_∞ (shown in Blue) and l_2 (shown in Green) norm on the statistical features.

after adversarial training using adversarial examples from different methods including TSA-STAT. This performance is relative to the clean testing set of the data. We can easily observe from the results of using FGS, CW, and PGD for adversarial training (degrades overall performance), the validity of our claim: l_p distance-based perturbation lacks true-label guarantees and can degrade the overall performance of deep models on real-world data. On the other hand, by using the adversarial examples from TSA-STAT, the overall performance did not decrease and has improved for some datasets: for SC, the accuracy increased from 90% to 97% for WB, and accuracy on CC improved from 83% to 96% for BB_2. We observe that FGS was the worst method in terms of preserving the performance of deep models.

We conclude from the experiments to test the effectiveness of adversarial examples from TSA-STAT that indeed using statistical features is well-justified for adversarial time-series data. If the standard L_p-norm-based methods from the image domain were to be very effective for the time-series domain:

- TSA-STAT based attacks will not be able to fool the models using baselines as a defense method as shown in Figures 4.10 and 4.11.
- Adversarial training on clean data using baseline methods would have outperformed TSA-STAT unlike the observations from Figure 4.13.

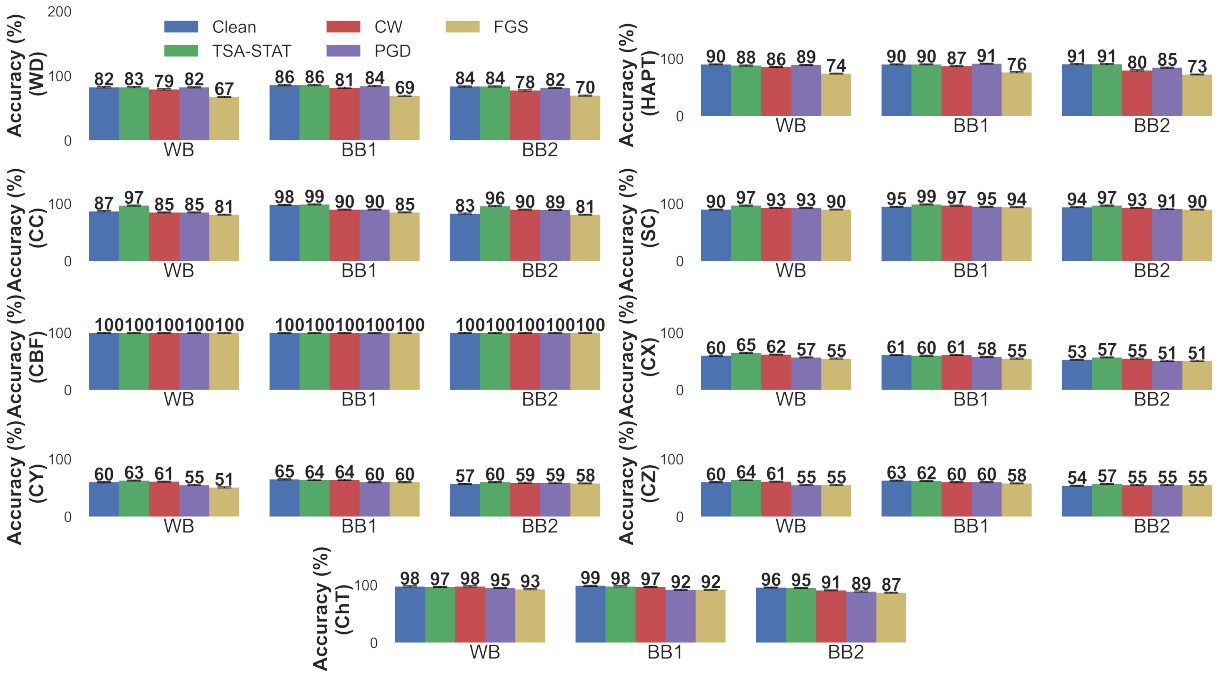


Figure 4.13 Results for adversarial training using adversarial examples from different methods including TSA-STAT, FGS, CW, PGD, and standard training (Clean) on clean testing data for different deep models.

Certified bounds. Using Algorithm 4, we can infer the robustness of an input X by calculating the upper bound δ that limits the tolerable adversarial perturbation over the $\|\cdot\|_\infty$. Hence, for any generated perturbation on X which employs $\hat{\delta} \leq \delta$, the classification result is guaranteed to remain the same. In other words, for a given time-series X and

its robustness bound δ, the perturbation $\hat{\delta}$ can take any value $\leq \delta$. As a consequence, the classification of a time-series input X with the perturbation $\hat{\delta}$ is stable/certified. For the following experiments, we employ $MAX = 5 \times 10^3$. For the generation of Σ, we use a random algorithm to generate a semi-definite positive matrix that has parameter σ as diagonal elements.

Figure 4.14 shows the classification accuracy on testing set under the attack of different possible $\hat{\delta}$ with various choices of $\sigma = \| \sum_{i,i} \|_\infty$ for the multivariate Gaussian n_p and n_q of the Algorithm 4 (noting that we observed similar findings on other datasets). σ refers to the diagonal element of the covariance matrix $\sum$. As an example, for WD dataset where $(\mu_P = 0.1, \sigma = 0.1)$: At $\hat{\delta} = 0$, we have a testing accuracy of 0.83, which translates to the fact that 83% of the testing set is robust to the given perturbation and 19% of the testing test is vulnerable to adversarial attacks. We also observe that the larger the value of σ is, the faster the curve declines. This shows that inputs are unstable with respect to robustness to noises with higher σ.

Figure 4.15 shows the robustness in accuracy of the deep model against perturbation $\hat{\delta}$ (In blue). It illustrates the classification accuracy on the testing set under attacks with different possible $\hat{\delta}$ values using (μ_P=0.01, σ=0.1, MAX=10^3) as parameters of the multivariate Gaussian for Algorithm 4. Consider the analysis for WD dataset as an example. At $\hat{\delta} = 0$, we have a test accuracy of 0.83, which translates to the fact that 83% of the test set inputs are robust to the given perturbation and 17% of the test is vulnerable to adversarial attacks. At $\hat{\delta}$=1, the plot shows that around 28% of the dataset has a certified bound $\delta \geq 1$.

The same figure shows the influence of adversarial training on the certification bound of the deep model via Algorithm 4. We also provide a comparison using Gaussian augmentation (Dodge and Karam, 2017) to show the substantial role of TSA-STAT in using statistical features vs. using a standard Gaussian noise. We can observe the effect of adversarial training with TSA-STAT on increasing the robustness of most inputs on the different datasets. For example, on HAPT dataset, the initial region of $\hat{\delta} \leq 1.5$, the robustness of several inputs

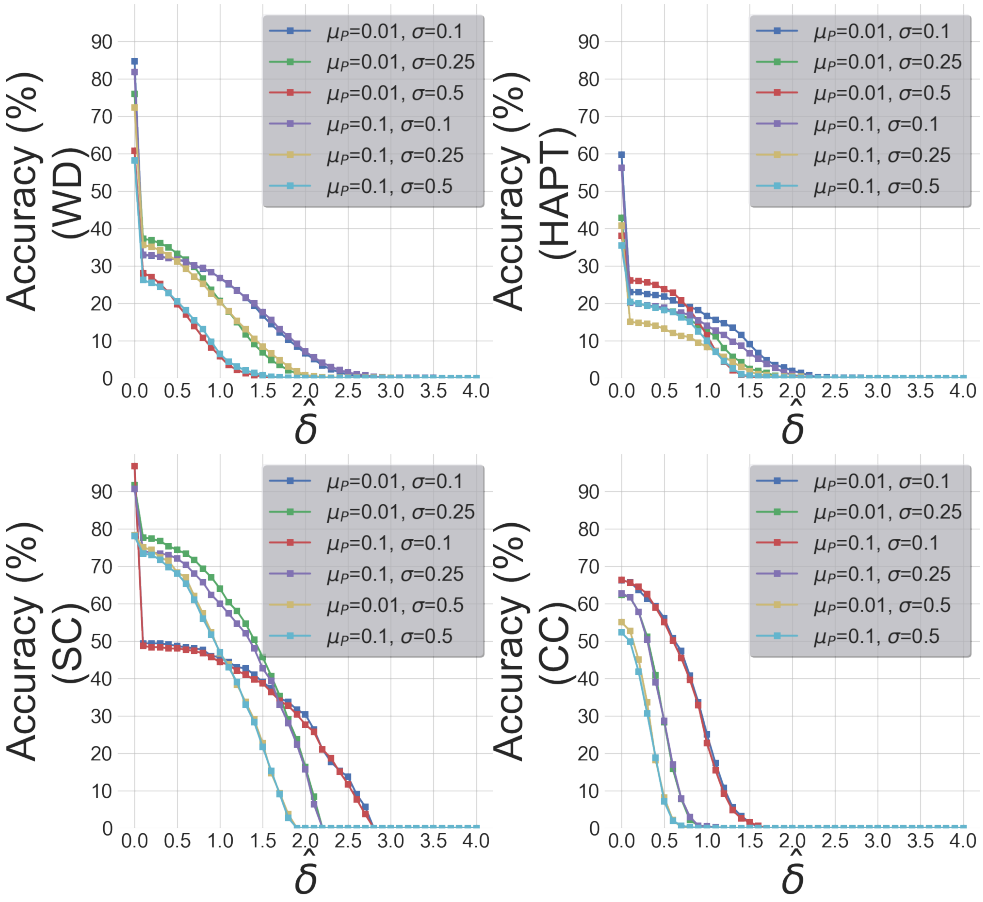


Figure 4.14 Certification lower bound accuracy on the testing data with varying (μ_P, σ) for Algorithm 4.

have increased (20% increase at $\hat{\delta}$=0 and 10% increase at $\hat{\delta}$=1.0).

Comparison with the work of (Karim, Majumdar, and Darabi, 2020). We mentioned in the related Work section that there is a recent work that proposed an approach for studying adversarial attacks for the time-series domain (Karim, Majumdar, and Darabi, 2020). This method employs network distillation to train a student model for creating adversarial attacks. We provide a comparison between TSA-STAT and the network distillation approach to show the effectiveness of our proposed framework. First, the method in (Karim, Majumdar, and Darabi, 2020) is severely limited: only a small number of target classes yield to a generation of adversarial examples and the method does not guarantee a generation of adversarial example for every input. (Karim, Majumdar, and Darabi, 2020) showed that

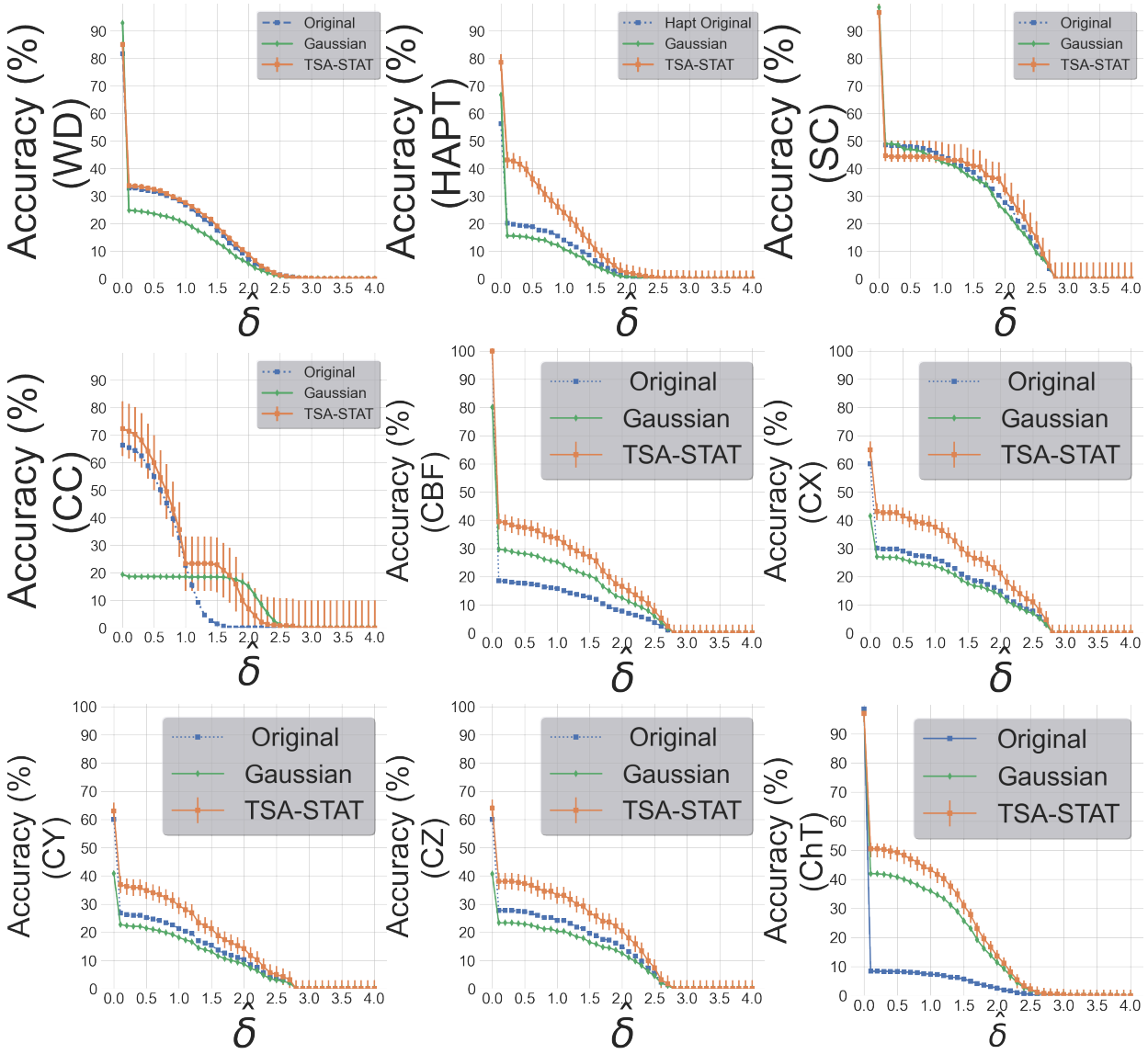


Figure 4.15 Robustness results with adversarial training. Comparison of the accuracy of Original model (standard training without adversarial examples) and adversarial training based on TSA-STAT and Gaussian augmentation. This figure illustrates that TSA-STAT is a better method to improve the robustness of deep model as it has the highest accuracy for a given $\hat{\delta}$ for most datasets.

for many datasets, this method creates a limited number of adversarial examples in the white-box setting. To test the effectiveness of this attack against TSA-STAT, we employ adversarial training using adversarial examples generated by the model proposed in (Karim, Majumdar, and Darabi, 2020) under the black-box setting. We use the code [1] provided by

[1]https://github.com/titu1994/Adversarial-Attacks-Time-Series.git

the authors to generate the adversarial examples using this baseline method.

Figure 4.16 shows the fooling rate of TSA-STAT generated attacks on different datasets. We can conclude that adversarial training using (Karim, Majumdar, and Darabi, 2020) does not improve the robustness of the models against our proposed attack. Additionally, we show a direct comparison between TSA-STAT and (Karim, Majumdar, and Darabi, 2020) using the attack performance in Figure 4.17. This figure shows the results comparing both attack performances under the white-box setting WB. We observe that the attack success rate (α_{ref}) of TSA-STAT outperforms the adversarial attacks created by (Karim, Majumdar, and Darabi, 2020) method. Figure 4.18 shows the effectiveness of adversarial examples generated from (Karim, Majumdar, and Darabi, 2020) on the deep models created via adversarial training using augmented data from TSA-STAT. We can see that using TSA-STAT for adversarial training results in a robust model against any attack generated by the method in (Karim, Majumdar, and Darabi, 2020).

4.4.4 Summary of Key Experimental Findings

Our comprehensive experimental evaluation demonstrated that TSA-STAT is an effective adversarial framework for time-series domain. We briefly summarize the main experimental findings below.

- The similarity measure based on statistical features of time-series used by TSA-STAT is more effective in capturing the unique characteristics of time-series data when compared to the standard algorithms which rely on l_p-norm distance (Figure 4.7).
- Figures 4.10 and 4.11 demonstrate that the instance-specific and universal adversarial attacks created by TSA-STAT are very effective in fooling DNNs for time-series classification tasks and evading adversarial training based on adversarial examples created by prior methods.
- Adversarial examples created by TSA-STAT provide better true-label guarantees (ex-

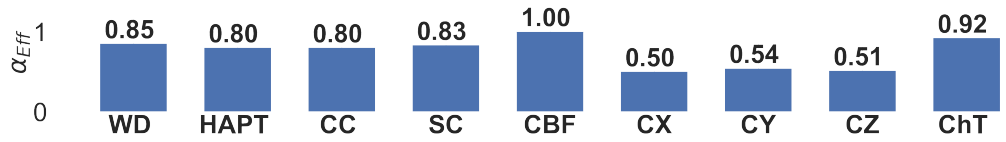


Figure 4.16 Results for effectiveness of TSA-STAT on deep models via adversarial training using the augmented data generated from (Karim, Majumdar, and Darabi, 2020).

Figure 4.17 Results for the effectiveness of TSA-STAT and (Karim, Majumdar, and Darabi, 2020) method under the white-box setting WB.

Figure 4.18 Results of TSA-STAT based adversarial training performance on predicting the true labels of adversarial attacks generated by (Karim, Majumdar, and Darabi, 2020).

amples belonging to the semantic space of true label) than those based on prior methods relying on l_p-norm distance. As a result, adversarial training based on TSA-STAT improves the robustness of deep models more than adversarial training with prior methods (Figure 4.13).

- Figure 4.15 demonstrates that adversarial training based on TSA-STAT provides better robustness certification for time-series classifiers than prior methods.time tun

4.5 Summary

We introduced in this chapter the TSA-STAT framework that studies adversarial robustness of deep models for time-series domain. TSA-STAT relies on two key ideas to create more effective adversarial examples for the time-series domain: 1) Constraints over statistical features of time-series signals to preserve similarities between original input and adversarial examples; and 2) Polynomial transformations to expand the space of valid adversarial examples compared to prior methods. TSA-STAT synergistically combines these two key ideas to overcome the drawbacks of prior methods from the image domain which rely on l_p-distance and are not suitable for the time-series domain. We provided theoretical and empirical analysis to explain the importance of these two key ideas in making TSA-STAT more suitable to create adversarial attacks for the time-series domain. We also provided certification guarantees for adversarial robustness of the TSA-STAT framework. We theoretically derived the computation of certification bound for TSA-STAT and provided a concrete algorithm that can be used with any deep model for the time-series domain. Finally, we empirically demonstrated the effectiveness of TSA-STAT on diverse real-world datasets and different deep models in terms of fooling rate and improved robustness with adversarial training. Our work concludes that time-series domain requires separate investigation for robustness analysis due to its unique characteristics and shows the effectiveness of the TSA-STAT framework towards this goal.

CHAPTER FIVE

DYNAMIC TIME WARPING BASED ADVERSARIAL FRAMEWORK FOR TIME-SERIES DOMAIN

T. Belkhouja, Y. Yan, and J. Doppa. "Dynamic Time Warping based Adversarial Framework for Time-Series Domain". *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 45(6): 7353-7366*, 2022.

Originally published in the *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Attributions:

T. Belkhouja has contributed to this work by formulating the problem setting, investigating the state of the art, formulating the theoretical contribution and the algorithmic solution, implementing the algorithm and running the required empirical analysis to highlight the performance improvement of the proposed solution compared to the state of the art.

Y. Yan has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the formulation of the theory of the proposed solution, and in writing the scientific manuscript.

J. Doppa has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

DYNAMIC TIME WARPING BASED ADVERSARIAL FRAMEWORK FOR TIME-SERIES DOMAIN

In this chapter, we propose a novel adversarial framework for time-series domain referred as *Dynamic Time Warping for Adversarial Robustness (DTW-AR)*. DTW-AR employs the dynamic time warping measure (Sakoe, 1971; Müller, 2007) as it can be used to measure a realistic distance between two time-series signals (e.g., invariance to shift and scaling operations) (Berndt and Clifford, 1994; Müller, 2007). For example, a signal that has its frequency changed due to Doppler effect would output a small DTW measure to the original signal. However, if Euclidean distance is used, both signals would look very dissimilar, unlike the reality. We theoretically analyze the suitability of DTW measure over the Euclidean distance. Specifically, the space of candidate adversarial examples in the DTW space is a superset of those in Euclidean space for the same distance bound. Therefore, DTW measure provides a more appropriate bias than the Euclidean space for the time series domain and our experiments demonstrate practical benefits of DTW-based adversarial examples.

To create targeted adversarial examples, we formulate an optimization problem with the DTW measure bound constraint and propose to solve it using an iterative gradient-based approach. However, this simple method has two drawbacks. First, this method allows us to only find one valid adversarial example out of multiple solution candidates from the search space because it operates on a single optimal alignment. Second, we need to compute DTW measure in each iteration as the optimal DTW alignment path changes over iterations. Since the number of iterations are typically large and DTW computation is expensive, the overall algorithm becomes prohibitively slow. To successfully overcome these two drawbacks, *the key insight is to employ stochastic alignments to create adversarial examples*. We theoretically and experimentally show that a simpler distance measure based on random alignment path upper-bounds the DTW measure measure and that this *bound is tight*. This algorithm allows us to efficiently create many diverse adversarial examples using different alignment paths to

improve the robustness of DNN models via adversarial training. Our experiments on real-world time-series datasets show that the DTW-AR creates more effective adversarial attacks to fool DNNs when compared to prior methods and enables improved robustness.

5.1 Background and problem setup

Let $X \in \mathbb{R}^{n \times T}$ be a multi-variate time-series signal, where n is the number of channels and T is the window-size of the signal. We consider a DNN classifier $F_\theta : \mathbb{R}^{n \times T} \to \mathcal{Y}$, where θ stands for parameters and $\mathcal{Y}$ is the set of classification labels.

X_{adv} is called an adversarial example of X if:

$$\left\{ X_{adv} \;\middle|\; DIST(X_{adv}, X) \leq \delta \text{ and } F_\theta(X) \neq F_\theta(X_{adv}) \right\}$$

where δ defines the neighborhood of highly-similar examples for input X using a distance metric $DIST$ to create worst-possible outcomes from the learning agent's perspective. Note that adversarial examples depend on the target concept because it defines the notion of invariance we care about.

DTW measure. The DTW measure between two uni-variate signals X and $Z \in \mathbb{R}^T$ is computed via a cost matrix $C \in \mathbb{R}^{T \times T}$ using a dynamic programming (DP) algorithm with time-complexity $\mathcal{O}(T^2)$. The cost matrix is computed recursively using the following equation:

$$C_{i,j} = d(X_i, Z_j) + \min \left\{ C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1} \right\} \tag{5.1}$$

where $d(\cdot, \cdot)$ is any given distance metric (e.g., $\| \cdot \|_p$ norm). The DTW measure between signals X and Z is $DTW(X, Z) = C_{T,T}$. The sequence of cells $P = \{c_{i,j} = (i, j)\}$ contributing to $C_{T,T}$ is the *optimal alignment path* between X and Z. Figure 5.1 provides illustration for an optimal alignment path. We note that the diagonal path corresponds to the Euclidean distance metric.

For the multi-variate case, where X and $Z \in \mathbb{R}^{n \times T}$, to measure the DTW measure using Equation 5.1, we have $d(X_i, Z_j)$ with $X_i, Z_j \in \mathbb{R}^n$ (Shokoohi-Yekta et al., 2017). We

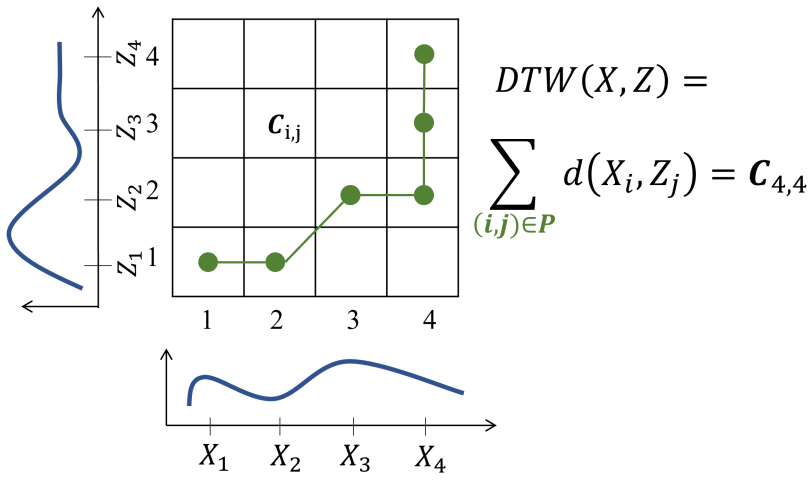


Figure 5.1 Illustration of DTW alignment between two uni-variate signals X and Z of length 4. The optimal alignment path (shown in green color) is $P = \{(1,1),(2,1),(3,2),(4,2),(4,3),(4,4)\}$.

define the distance function $dist_P(X, Z)$ between time-series inputs X and Z according to an alignment path P using the following equations:

$$dist_P(X, Z) = \sum_{(i,j)\in P} d(X_i, Z_j) \tag{5.2}$$

Hence, the DTW measure between X and Z is given by:

$$DTW(X, Z) = \min_P dist_P(X, Z) \tag{5.3}$$

Time-series pre-processing methods. A possible solution to overcome the Euclidean distance concerns is to introduce pre-processing steps that are likely to improve the existing frameworks. Simple pre-processing steps such as MinMax-normalization or z-normalization only solves problems such as scaling problem. However, they do not address any concern about signal-warping or time-shifts. Other approaches rely on learning feature-preserving representations. A well-known example is the GRAIL(Paparrizos and Franklin, 2019) framework. This framework aims to learn compact time-series representations that preserve the properties of a pre-defined comparison function such as DTW. The main concern about feature-preserving pre-processing steps is that the representation learnt is not reversible. In

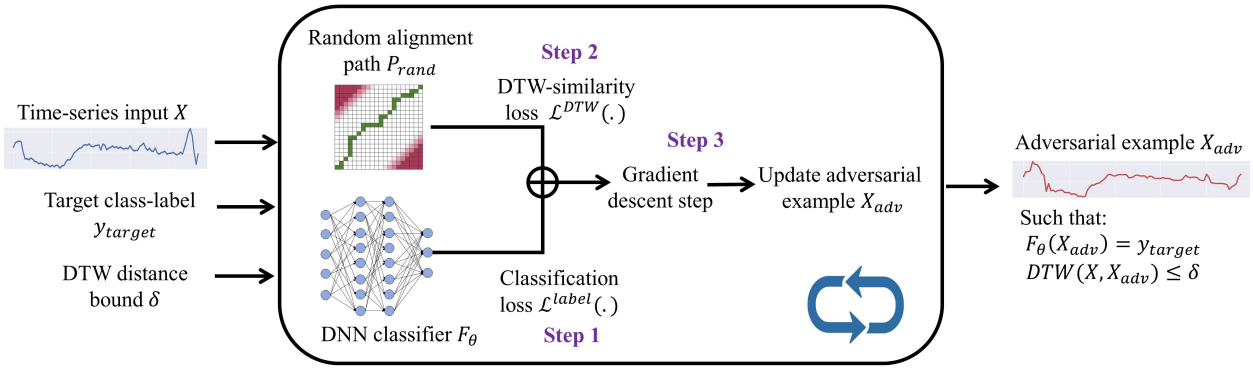


Figure 5.2 Overview of the DTW-AR framework to create targeted adversarial examples. Given an input X, a target class-label y_{target} and a distance bound δ, DTW-AR aims to identify an adversarial example X_{adv} using a random alignment path P_{rand}. DTW-AR solves an optimization problem involving a DTW-similarity loss and a classification loss using a random alignment path P_{rand} and a DNN classifier F_θ. Using different random alignment paths, DTW-AR will be able to create diverse adversarial examples which meet the DTW measure bound δ.

other words, a real-world time-series signal cannot be generated from the estimated representation. The goal of adversarial attacks is to create real-world time-series that can be used to fool any DNN. Such challenges would limit the usability and the generality of methods based on pre-processing steps to study the robustness of DNNs for time-series data. All proof of the proposed theorems are provided in Appendix B.

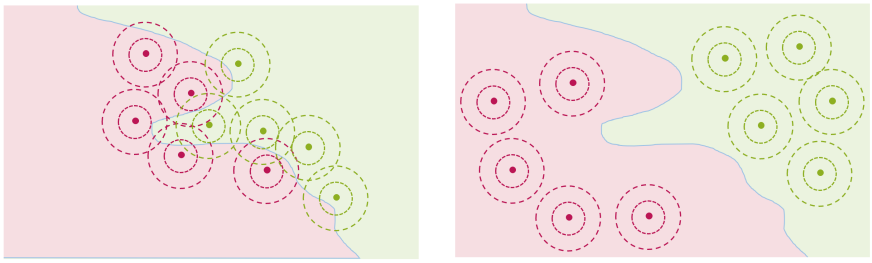
5.2 Dynamic Time Warping based Adversarial Robustness framework

The DTW-AR framework creates targeted adversarial examples for time-series domain using the DTW measure as illustrated in Figure 5.2. For any given time-series input X, DNN classifier F_θ, and distance bound δ, we solve an optimization problem to identify an adversarial example X_{adv} which is within DTW measure δ to the original time-series signal X. In what follows, we first provide empirical and theoretical results to demonstrate the suitability of DTW measure over Euclidean distance for adversarial robustness studies in the time-series domain (Section 3.1). Next, we introduce the optimization formulation based on the DTW measure to create adversarial examples and describe its main drawbacks

(Section 3.2). Finally, we explain our key insight of using stochastic alignment paths to successfully overcome those drawbacks to efficiently create diverse adversarial examples and provide theoretical justification (Section 3.3).

5.2.1 Effectiveness of DTW measure measure

Empirical justification. As we argued before, the standard l_2 distance is impractical for adversarial learning in time-series domain. Perturbations based on Euclidean distance can result in adversarial time-series signals which semantically belong to a different class-label. Based on the real-world data representation provided in Figure 5.4, we create and show in Figure 5.3 an intuitive illustration of suitability of DTW over l_2 distance to explain the advantages of DTW as a similarity measure. It shows the difference in the true data

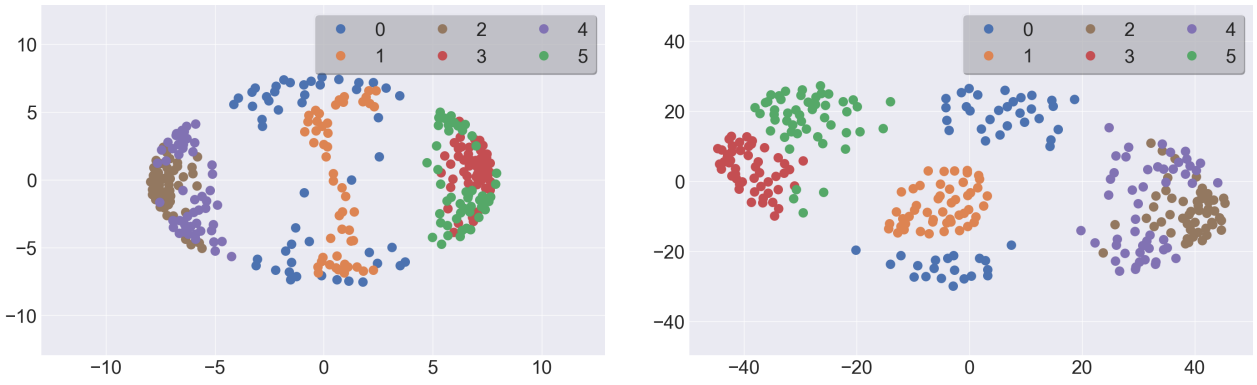


Euclidean Space DTW Space

Figure 5.3 Illustration of the suitability of DTW over Euclidean distance using the true data distribution from two classes shown in red and green colors. The concentric circles represent the close-similarity area of each input instance (i.e., center) using the corresponding distance measure.

distribution in Euclidean space (i.e., l_2 is used as the similarity measure) and in DTW space (i.e., DTW is used as the similarity measure) for two classes shown in red and green colors. The concentric circles represent the close-similarity area around each input instance (i.e., center) where adversarial examples are considered. We can observe that in the Euclidean space, adversarial example of an input instance can belong to another class label, which is not the case in the DTW space. This simple illustration shows how DTW-AR can generate effective adversarial examples due to the appropriate bias of DTW for time-series domain.

This abstraction is tightly based on the observations made on real-world data. We employ multi-dimensional scaling (MDS), a visual representation of dissimilarities between sets of data points (Buja et al., 2008), to compare DTW and Euclidean spaces. MDS is a dimensionality reduction method that preserves the distances between data points in the original space. Figure 5.4 shows MDS results of `SC` dataset. We can clearly see how the data from different class labels are better clustered in the DTW space compared to the Euclidean space, as provided in Figure 5.4. An adversarial example for an `SC` data point in the green-labeled class is more likely to semantically belong to the red-labeled distribution in the Euclidean space. However, in the DTW space, the adversarial example is more likely to remain in the green-labeled space, while only being misclassified by the DNN classifier due the adversarial problem.



Euclidean Space DTW Space

Figure 5.4 Multi-dimensional scaling results showing the labeled data distribution in Euclidean space (left column) and DTW space (right column) for the SC dataset. DTW space exhibits better clustering for same-class data than Euclidean space.

Theoretical justification. We prove that the DTW measure allows DTW-AR to explore a larger space of candidate adversarial examples when compared to perturbations based on the Euclidean distance, i.e., identifies blind spots of prior methods. This result is based on the fact that the point-to-point alignment (i.e., Euclidean distance) between two time-series signals is not always the optimal alignment. Hence, the existence of adversarial examples which are similar based on DTW and may not be similar based on the Euclidean distance.

To formalize this intuition, we provide Observation 1. We characterize the effectiveness of DTW-AR based attack as better for their ability to extend the space of attacks based on the Euclidean distance and their potential to fool DNN classifiers that rely on Euclidean distance for adversarial training. Our experimental results demonstrate that DTW-AR generates effective adversarial examples to fool the target DNN classifiers by leveraging the appropriate bias of DTW for time-series data.

Observation 1. *Let l_2 be the equivalent of Euclidean distance using the cost matrix in the DTW space. $\forall X \in \mathbb{R}^{n \times T}$ $(n > 1, T > 1)$, there exists $\epsilon \in \mathbb{R}^{n \times T}$ and an alignment path P such that $dist_P(X, X + \epsilon) \leq \delta$ and $l_2(X, X + \epsilon) > \delta$.*

Theorem 3. *For a given input space $\mathbb{R}^{n \times T}$, a constrained DTW space for adversarial examples is a strict superset of a constrained euclidean space for adversarial examples. If $X \in \mathbb{R}^{n \times T}$:*

$$\left\{ X_{adv} \middle| DTW(X, X_{adv}) \leq \delta \right\} \supset \left\{ X_{adv} \middle| \|X - X_{adv}\|_2^2 \leq \delta \right\} \tag{5.4}$$

As an extension of Observation 1, the above theorem states that in the space where adversarial examples are constrained using a DTW measure bound, there exists more adversarial examples that are not part of the space of adversarial examples based on the Euclidean distance for the same bound (i.e., blind spots). This result implies that DTW measure has an appropriate bias for the time-series domain. Hence, our DTW-AR framework is *potentially* capable of creating more effective adversarial examples than prior methods based on l_2 distance for the same distance bound constraint. These adversarial examples are potentially more effective as they are able to break deep models by leveraging the appropriate bias of DTW measure. We present the proofs of both Observation 1 and Theorem 3 in the Appendix B.

However, to convert this potential to reality, we need an algorithm that can efficiently search this larger space of attacks to identify most or all adversarial examples which meet the

DTW measure bound. Indeed, developing such an algorithm is one of the key contributions of this work.

5.2.2 Naive optimization based formulation and challenges to create adversarial examples

To create adversarial examples to fool the given DNN F_θ, we need to find an optimized perturbation of the input time-series X to get X_{adv}. Our approach is based on minimizing a *loss function* $\mathcal{L}$ using gradient descent that achieves two goals. 1) Misclassification goal: Adversarial example X_{adv} to be mis-classified by F_θ as a target class-label y_{target}; and 2) DTW similarity goal: close DTW-based similarity between time-series X and adversarial example X_{adv}.

To achieve the mis-classification goal, we employ the formulation of (Carlini and D. Wagner, 2017) to define a loss function:

$$\mathcal{L}^{label}(X_{adv}) = \max\Big[\max_{y \neq y_{target}} \left(\mathcal{S}_y\left(X_{adv}\right)\right) - \mathcal{S}_{y_{target}}\left(X_{adv}\right), \ \rho\Big] \tag{5.5}$$

where $\rho < 0$. It ensures that the adversarial example will be classified by the DNN as class-label y_{target} with a confidence $|\rho|$ using the output of the pre-softmax layer $\{\mathcal{S}_y\}_{y \in Y}$.

To achieve the DTW similarity goal, we need to create X_{adv} for a given time-series input X such that $DTW(X, X_{adv}) \leq \delta$. We start by a naive optimization over the DTW measure using the Soft-DTW measure $\text{SDTW}(X, X_{adv})$ (Marco Cuturi and Blondel, 2017). Hence, the DTW similarity loss function is:

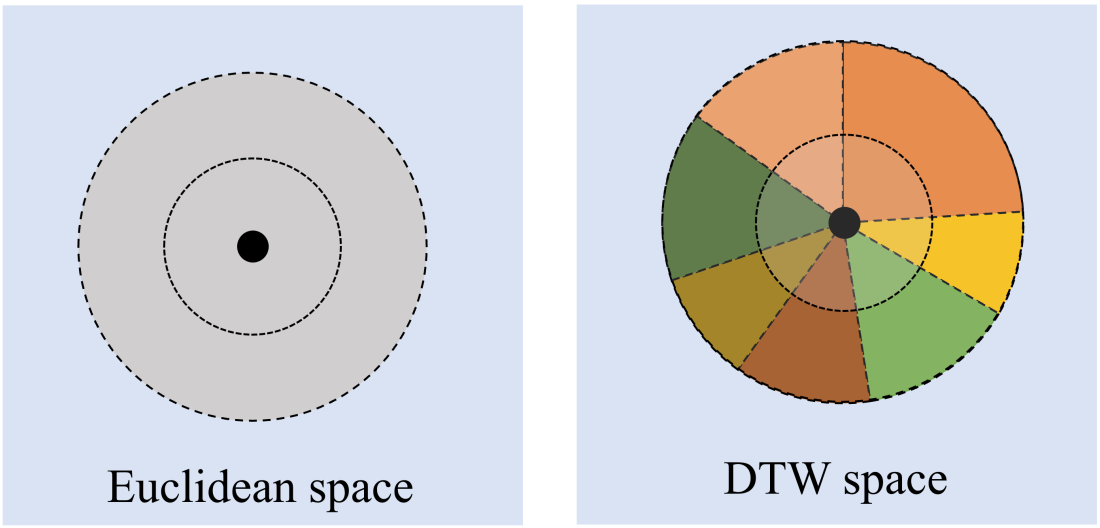
$$\mathcal{L}^{DTW}(X_{adv}) = \text{SDTW}(X, X_{adv}) \tag{5.6}$$

The final loss function $\mathcal{L}$ we want to minimize to create optimized adversarial example X_{adv} is:

$$\mathcal{L}(X_{adv}) = \mathcal{L}^{label}(X_{adv}) + \mathcal{L}^{DTW}(X_{adv}) \tag{$\star$}$$

We operate under white-box setting and can employ gradient descent to minimize the loss function in Equation $\star$ over X_{adv}. This approach works for black-box setting also. In this work, we consider the general case where we do not query the black-box target DNN classifier. We show through experiments that the created adversarial examples can generalize to fool other black-box DNNs.

Challenges of Naive approach. Recall that our overall goal is to identify most or all targeted adversarial time-series examples that meet the DTW measure bound. This will allow us to improve the robustness of DNN model using adversarial training. This naive approach has two main drawbacks.



(a) (b)

Figure 5.5 Illustration of the close-similarity space around a given time-series signal (black center) in the Euclidean and DTW space. Using l_2 norm is sufficient to explore the entire Euclidean space around the input. However, in the DTW space, each colored section corresponds to one adversarial example that meets the DTW measure bound constraint. Each of them can be found using only a subset of candidate alignment paths.

• *Single adversarial example.* The method allows us to only find one valid adversarial example out of multiple solution candidates from the search space because it operates on a single optimal alignment path. Using a single alignment path (whether the diagonal path

for Euclidean distance or the optimal alignment path generated by DTW), the algorithm will be limited to the adversarial examples which use that single alignment. In Figure 5.5, we provide a conceptual illustration of $S_{ADV}(X)$, the set of all adversarial examples X_{adv} which meet the distance bound constraint $DTW(X, X_{adv}) \leq \delta$. In the Euclidean space, using l_2 norm is sufficient to explore the entire search space around the original input to create adversarial examples. However, in the DTW space, each colored section in $S_{ADV}(X)$ can only be found using a subset of candidate alignment paths.

• *High computational cost.* DTW is non-differentiable and approximation methods are often used in practice. These methods require $\mathcal{O}(n.T^2)$ to fill the cost matrix and $\mathcal{O}(T)$ to backtrack the optimal alignment path. These steps are computationally-expensive. Gradient-based optimization iteratively updates the adversarial example X_{adv} to achieve the DTW similarity goal, i.e., $DTW(X, X_{adv}) \leq \delta$, and the mis-classification goal, i.e., $F_\theta(X_{adv})=y_{target}$. Standard algorithms such as projected gradient descent (PGD) (Aleksander Madry et al., 2017) and Carloni & Wagner (CW) (Carlini and D. Wagner, 2017) require a large number of iterations to generate valid adversarial examples. This is also true for the recent computer vision specific adversarial algorithms (Laidlaw and Feizi, 2019a; Shafahi et al., 2020). For time-series signals arising in many real-world applications, the required number of iterations to create successful attacks can grow even larger. We need to compute DTW measure in each iteration as the optimal DTW alignment path changes over iterations. Therefore, it is impractical to use the exact DTW computation algorithm to create adversarial examples. We also show that the existing optimized approaches to estimate the DTW measure remain computationally expensive for an adversarial framework. We provide results to quantify the runtime cost in our experimental evaluation.

Algorithm 5 DTW-AR based Adversarial Algorithm

Input: time-series X; DNN classifier F_θ; target class-label y_{target}; learning rate η; maximum iterations MAX

Output: adversarial example X_{adv}

1: $P_{rand} \leftarrow$ random alignment path

2: Initialization: $X_{adv} \leftarrow X$

3: **for** i=1 to MAX **do**

4: $\quad \mathcal{L}(X_{adv}) \leftarrow \mathcal{L}^{label}(X_{adv}) + \mathcal{L}^{DTW}(X_{adv}, P_{rand})$

5: $\quad$ Compute gradient $\nabla_{X_{adv}} \mathcal{L}(X_{adv})$

6: $\quad$ Perform gradient descent step:

$\quad X_{adv} \leftarrow X_{adv} - \eta \times \nabla_{X_{adv}} \mathcal{L}(X_{adv})$

7: **end for**

8: **return** optimized adversarial example X_{adv}

5.2.3 Theoretical justification for stochastic alignment

In this section, we describe the key insight of DTW-AR to overcome the above-mentioned two challenges and provide theoretical justification.

To overcome the above-mentioned two challenges of the naive approach, we propose the use of a random alignment path to create adversarial attacks on DNNs for time-series domain. The key idea is to select a random alignment path P and to execute our adversarial algorithm while constraining over $dist_P(X, X_{adv})$ instead of $DTW(X, X_{adv})$. This choice is justified from a theoretical point-of-view due to the special structure in the problem to create DTW based adversarial examples. Using the distance function $dist_P(X, X_{adv})$, we redefine Equation 5.6 as follows:

$$\begin{aligned} \mathcal{L}^{DTW}(X_{adv}, P) = \; & \alpha_1 \times dist_P(X, X_{adv}) \\ & - \alpha_2 \times dist_{P_{diag}}(X, X_{adv}) \end{aligned} \tag{5.7}$$

where $\alpha_1 > 0,\ \alpha_2 \geq 0,\ P_{diag}$ is the diagonal alignment path equivalent to the Euclidean

distance, and P is a given alignment path ($P \neq P_{diag}$). The first term of Equation 5.7 is defined to bound the DTW similarity of adversarial example X_{adv} to a threshold δ as stated in Observation 2. The second term represents a penalty term to account for adversarial example with close Euclidean distance to the original input X and pushes the algorithm to look beyond adversarial examples in the Euclidean space. The coefficients α_1 and α_2 contribute in defining the position of the adversarial output in the DTW and/or Euclidean space. If $\alpha_2 \to 0$, the adversarial example X_{adv} will be highly similarity to the original input X in the DTW space with no consideration to the Euclidean space. Hence, the adversarial example may be potentially adversarial in the Euclidean space also. However, if $\alpha_2 > 0$, the adversarial output will be highly similar to the original input in the DTW space but out of the scope of adversarial attacks in the Euclidean space (i.e., a blind spot). Recall from Theorem 1 that DTW space allows more candidate adversarial examples than Euclidean space. Hence, this setting allows us to find blind spots of Euclidean space based attacks.

The DTW-AR approach to create adversarial examples is shown in Algorithm 5. We note that the naive approach that uses Soft-DTW with the Carlini & Wagner loss function is a sub-case of DTW-AR as shown below:

$$\begin{aligned} &\text{SDTW}(X, X_{adv}) = \mathcal{L}^{DTW}(X_{adv}, P_{DTW}) = \\ &1 \times dist_{P_{DTW}}(X, X_{adv}) - 0 \times dist_{P_{diag}}(X, X_{adv}) \end{aligned} \tag{5.8}$$

where P_{DTW} is the optimal DTW alignment path.

Observation 2. *Given any alignment path P and two multivariate time-series signals $X, Z \in \mathbb{R}^{n \times T}$. If we have $dist_P(X, Z) \leq \delta$, then $DTW(X, Z) \leq \delta$.*

Observation 2 states that $dist_P(X, Z)$ defined with respect to a path P is always an upper bound for $DTW(X, Z)$, since DTW uses the optimal alignment path. Hence, when the alignment path is fixed, the time-complexity is reduced to a simpler similarity measure that requires only $\mathcal{O}(n.T)$, which results in significant computational savings due to repeated calls within the adversarial algorithm.

Our stochastic alignment method also improves the search strategy for finding multiple desired adversarial examples. Suppose $S_{ADV}(X)$ is the set of all adversarial examples X_{adv} which meet the distance bound constraint $DTW(X, X_{adv}) \leq \delta$. Each adversarial example in $S_{ADV}(X)$ can be found using only a subset of candidate alignment paths. By using a stochastic alignment path, we can leverage the large pool of different alignment paths to uncover more than one adversarial example from $S_{ADV}(X)$. On the other hand, if the exact DTW computation based algorithm was feasible, we would only find a *single* X_{adv}, as DTW based algorithm operates on a single optimal alignment path.

Theoretical tightness of bound. While Observation 2 provides an upper bound for the DTW measure, it does not provide any information about the tightness of the bound. To analyze this gap, we need to first define a similarity measure between two alignment paths to quantify their closeness. We define `PathSim` as a similarity measure between two alignment paths P_1 and P_2 in the DTW cost matrix of time-series signals $X, Z \in \mathbb{R}^{n \times T}$. Let $P_1 = \{c_1^1, ..., c_k^1\}$ and $P_2 = \{c_1^2, ..., c_l^2\}$ represent the sequence of cells for paths P_1 and P_2 respectively.

$$\texttt{PathSim}\ (P_1, P_2) = \\ \frac{1}{2T}\left(\sum_{c_i^1} \min_{c_j^2} \|c_i^1 - c_j^2\|_1 + \sum_{c_i^2} \min_{c_j^1} \|c_i^2 - c_j^1\|_1\right) \tag{5.9}$$

As $\texttt{PathSim}(P_1, P_2)$ approaches 0, P_1 and P_2 are very similar, and they will be the exact same path if $\texttt{PathSim}(P_1, P_2) = 0$. For $X, Z \in \mathbb{R}^{n \times T}$, two very similar alignment paths corresponds to a similar feature alignment between X and Z. Theorem 4 shows the tightness of the bound given in Observation 2 using the path similarity measure defined above.

Theorem 4. *For a given input $X \in \mathbb{R}^{n \times T}$ and a random alignment path P_{rand}, the resulting adversarial example X_{adv} from the minimization over $dist_{P_{rand}}(X, X_{adv})$ is equivalent to*

minimizing over $DTW(X, X_{adv})$*. For any* X_{adv} *generated by DTW-AR using* P_{rand}*, we have:*

$$\begin{cases} \texttt{PathSim}(P_{rand}, P_{DTW}) = 0 \\ \quad \& \\ dist_{P_{rand}}(X, X_{adv}) = DTW(X, X_{adv}) \end{cases} \tag{5.10}$$

where P_{DTW} *is the optimal alignment path found using DTW computation between* X *and* X_{adv}*.*

Similarity measure `PathSim` definition. For DTW-AR, we rely on a stochastic align-

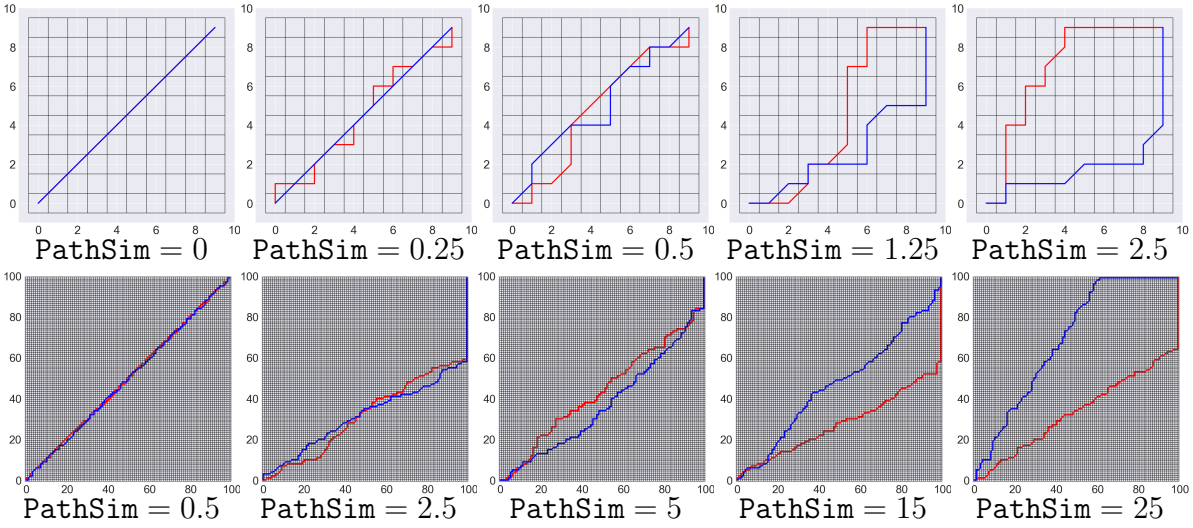


Figure 5.6 Visualization of `PathSim` values along different example alignment paths in $\mathbb{R}^{n\times 10}$ (First row) and $\mathbb{R}^{n\times 100}$ (Second row) spaces.

ment path to compute $dist_P$ defined in Equation 5.2. To improve our understanding of the behavior of DTW-AR framework based on stochastic alignment paths, we propose to define a similarity measure that we call `PathSim`. This measure quantifies the similarities between two alignment paths P_1 and P_2 in the DTW cost matrix for two time-series signals $X, Z \in \mathbb{R}^{n\times T}$. If we denote the alignment path sequence $P_1 = \{c_1^1, \cdots, c_k^1\}$ and $P_2 = \{c_1^2, \cdots, c_l^2\}$, then we can measure their similarity as defined in Equation 5.9.

This definition is a *valid* similarity measure as it satisfies all the distance axioms (Cullinane, 2011):

Non-negativity: By definition, $\texttt{PathSim}(P_1, P_2)$ is a sum of l_1 distances, which are all positives. Hence, $\texttt{PathSim}(P_1, P_2) \geq 0$.

Unicity: $\texttt{PathSim}(P_1, P_2) = 0$

$$\begin{aligned}
&\iff \frac{1}{2T}(\sum_{c_i^1} \min_{c_j^2} \|c_i^1 - c_j^2\|_1 \\
&\qquad + \sum_{c_i^2} \min_{c_j^1} \|c_i^2 - c_j^1\|_1) = 0 \\
&\iff \sum_{c_i^1} \min_{c_j^2} \|c_i^1 - c_j^2\|_1 + \sum_{c_i^2} \min_{c_j^1} \|c_i^2 - c_j^1\|_1 = 0
\end{aligned}$$

As we have a sum equal to 0 of all positive terms, we can conclude that each term $(\min \|\cdot\|_1)$ is equal to 0: $\texttt{PathSim}(P_1, P_2) = 0$

$$\begin{aligned}
&\iff \forall i : \|c_i^1 - c_i^2\|_1 = 0 \\
&\iff \forall i : \quad c_i^1 = c_i^2
\end{aligned}$$

As both paths have the same sequence of cells, we can safely conclude that $\texttt{PathSim}(P_1, P_2) = 0 \iff P_1 = P_2$:

Symmetric Property:

$$\begin{aligned}
&\texttt{PathSim}(P_1, P_2) = \\
&\frac{1}{2T}\left(\sum_{c_i^1} \min_{c_j^2} \|c_i^1 - c_j^2\|_1 + \sum_{c_i^2} \min_{c_j^1} \|c_i^2 - c_j^1\|_1\right) \\
&= \frac{1}{2T}\left(\sum_{c_i^2} \min_{c_j^1} \|c_i^2 - c_j^1\|_1 + \sum_{c_i^1} \min_{c_j^2} \|c_i^1 - c_j^2\|_1\right) \\
&= \texttt{PathSim}(P_2, P_1)
\end{aligned}$$

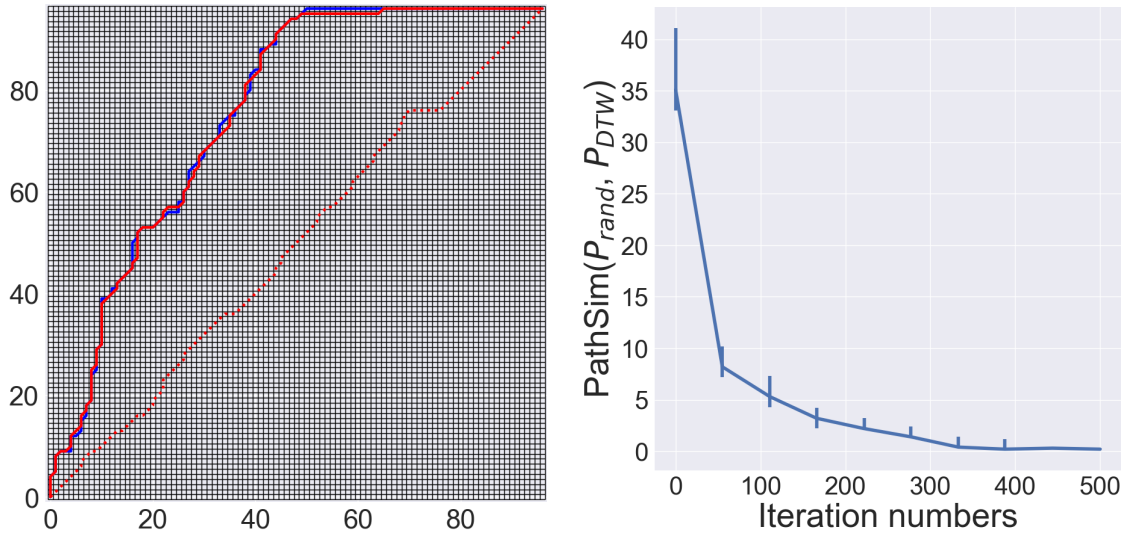
Note that the triangle inequality is not applicable as the alignment path spaces does not support additive operations.

This similarity measure quantifies the similarity between two alignment paths as it measures the l_1 distance between the different cells of each path. The multiplication factor $1/2T$

is introduced to prevent scaling of the measure for large T values for a given time-series input space $\mathbb{R}^{n \times T}$.

In Figure 5.6, we visually show the relation between `PathSim` measure and the alignment path for a given cost matrix. We observe that when $\texttt{PathSim}(P_1, P_2) \to 0$, P_1 and P_2 are very similar, and they will be the exact same path if $\texttt{PathSim}(P_1, P_2) = 0$. For $\texttt{PathSim}(P_1, P_2) \gg 0$, the alignment path will go through different cells which are far-placed from each other in the cost matrix.

Empirical tightness of bound. Figure 5.7 shows that over the iterations of the DTW-AR algorithm, the updated adversarial example yields to an optimal alignment path that is more similar to the input random path. This result strongly demonstrate that Theorem 2 holds empirically.



(a) (b)

Figure 5.7 (a) Example of the convergence of the optimal alignment path between the adversarial example and the original example at the start of the algorithm (dotted red path) and at the end (red path) to the given random alignment path (blue path). (b) `PathSim` score of the optimal alignment path between the adversarial example and the original example and the given random path for the ECG200 dataset averaged over multiple random alignment paths.

Corollary 1. *Let P_1 and P_2 be two alignment paths such that $\texttt{PathSim}(P_1, P_2) > 0$. If X^1_{adv}*

and X^2_{adv} are the adversarial examples generated using DTW-AR from any given time-series X using paths P_1 and P_2 respectively such that $DTW(X, X^1_{adv}) = \delta$ and $DTW(X, X^2_{adv}) = \delta$, then X^1_{adv} and X^2_{adv} are not necessarily the same.

Theorem 4 shows that the adversarial example generation using DTW-AR is equivalent to the ideal setting where it is possible to optimize $DTW(X, X_{adv})$. The above corollary extends Theorem 4 to show that if we employ different alignment paths within Algorithm 1, we will be able to find more adversarial examples which meet the distance bound in contrast to the naive approach.

5.3 Experiments and Results

In this section, we empirically evaluate the key elements of DTW-AR framework and discuss the results along different dimensions. Furthermore, we provide additional experiments and discussion on DTW-AR framework in Appendix D.

5.3.1 Experimental setup

Datasets. We employ the UCR datasets benchmark (Bagnall et al., 2020). We present the results on five representative datasets (`AtrialFibrillation`, `Epilepsy`, `ERing`, `Heartbeat`, `RacketSports`) from diverse domains noting that our findings are general as shown by the results on remaining UCR datasets in the Appendix D. We employ the standard training/validation/testing splits from these benchmarks.

Configuration of algorithms. We employ a 1D-CNN architecture for the target DNNs. We operate under a white-box (WB) setting for creating adversarial examples to fool WB model. To assess the effectiveness of attacks, we evaluate the attacks under the black-box (BB) setting and to fool BB model. The adversarial algorithm has no prior knowledge/querying ability of target DNN classifiers. Target DNNs include: 1) DNN model with a different architecture trained on clean data (BB); 2) DNNs trained using augmented data

from baselines attacks that are not specific to image domain: Fast Gradient Sign method (FGS) (Kurakin, I. Goodfellow, and Bengio, 2016), Carlini & Wagner (CW) attack (Carlini and D. Wagner, 2017), and Projected Gradient Descent (PGD) (Aleksander Madry et al., 2017); and 3) DNN models trained using stability training (Stephan Zheng et al., 2016) (STN) for learning robust classifiers.

DNN architectures. To evaluate the DTW-AR framework, we employ two different 1D-CNN architectures — A_0 and A_1 — to create two DNNs: WB uses A_0 to evaluate the adversarial attack under a white-box setting, and is trained using clean training examples. BB uses the architecture A_1 to evaluate the black-box setting for a model trained using clean examples. The architecture details of the deep learning models are presented in Table 5.1.

Table 5.1 Details of DNN architectures. C: Convolutional layers, K: kernel size, P: max-pooling kernel size, and R: rectified linear layer.

	C	K	C	K	P	R	R
A_0	x	x	66	12	12	1024	x
A_1	100	5	50	5	4	200	100

DTW-AR implementation. We implemented the DTW-AR framework using TensorFlow 2 (Martín Abadi and al., 2015). The parameter ρ that was introduced in Equation 5.5 plays an important role in the algorithm.

$$\mathcal{L}^{label}(X_{adv}) = \max[\max_{y \neq y_{target}} (\mathcal{S}_y(X_{adv})) - \mathcal{S}_{y_{target}}(X_{adv}), \ \rho] \tag{5.5}$$

ρ will push gradient descent to minimize mainly the second term ($\mathcal{L}^{DTW}$) when the first term plateaus at ρ. Otherwise, the gradient can minimize the general loss function by pushing $\mathcal{L}^{label}$ to $-\infty$, which is counter-productive for our goal. In all our experiments, we employ $\rho = -5$ for $\mathcal{L}^{label}$ in Equation 5.5 for a good confidence in the classification score. A good confidence score is important for the attack's effectiveness in a black-box setting. Black-box setting assumes that information about the target deep model including its parameters θ are

not accessible. In general, the attacker will create a proxy deep model to mimic the behavior of the target model using regular queries. This technique can be more effective when a target scenario is well-defined (Tramer et al., 2020; Papernot, McDaniel, I. Goodfellow, et al., 2017). However, in this work, we consider the general case where we do not query the black-box target DNN classifier for a better assessment of the proposed framework. Figure 5.8 shows the role of ρ value in enhancing DTW-AR attacks in a black-box setting on `ECG200` dataset noting that we see similar patterns for other datasets as well. Adversarial examples are

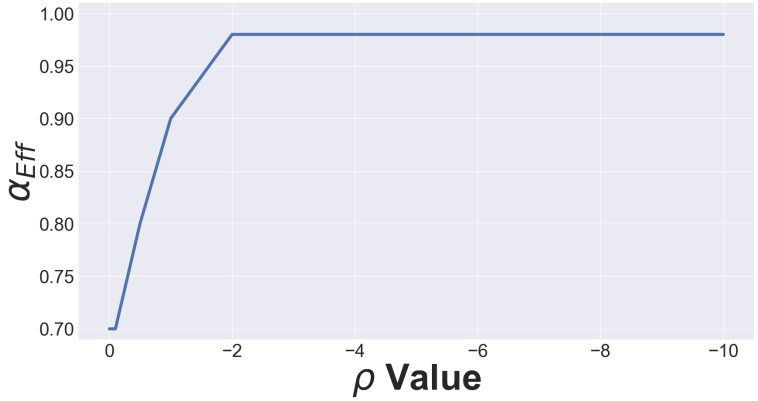


Figure 5.8 Results for the fooling rate on `ECG200` dataset w.r.t different ρ values for a black-box attack setting.

generated using a maximum of 5×10^3 iterations of gradient descent with the fixed learning rate η=0.01. After all the iterations, the final adversarial output is chosen from the iteration with the lowest DTW loss provided from Equation 5.7.

$$\begin{aligned} \mathcal{L}^{DTW}(X_{adv}, P) =& \alpha_1 \times dist_P(X, X_{adv}) \\ & - \alpha_2 \times dist_{P_{diag}}(X, X_{adv}) \end{aligned} \tag{5.7}$$

Experimentally, we notice that for $d(\cdot, \cdot)$ in Equation 1 of the main paper, there is no influence on the performance between choosing $p = 1, 2$ or ∞ for $d(\cdot, \cdot) = \| \cdot \|_p$. However, for datasets in $\mathbb{R}^{n \times T}$ with $n \geq 2$, we do not use $p = \infty$ as the dimensions are not normalized and data points will be compared only along the dimension with the greater magnitude.

Implementation of baselines. The baseline methods for CW, PGD ,and FGS were implemented using the CleverHans library (Papernot, Faghri, et al., 2018) with updates to TensorFlow 2. For FGS and PGD algorithms, we employed a minimal perturbation factors ($\epsilon < 1$) for two main reasons. First, larger perturbations significantly degrade the overall performance of the adversarial training and potentially creates adversarial signals that are semantically different than the original time-series input. Second, we want to avoid the risk of leaking label information (Aleksander Madry et al., 2017). STN was implemented using the code provided with the paper (Stephan Zheng et al., 2016).

Evaluation metrics. We evaluate attacks using the efficiency metric $\alpha_{Eff} \in [0, 1]$ over the created adversarial examples. α_{Eff} (higher means better attacks) measures the capability of adversarial examples to fool a given DNN F_θ to output the target class-label. α_{Eff} is calculated as the fraction of adversarial examples that are predicted correctly by the classifier: $\alpha_{Eff} = \frac{\#\text{ Adv. examples s.t.} F(X)==y_{target}}{\#\text{ Adv. examples}}$. We evaluate adversarial training by measuring the accuracy of the model to predict ground-truth labels of adversarial examples. A DNN classifier is robust if it is successful in predicting the true label of any given adversarial example.

5.3.2 Results and Discussion

Spatial data distribution with DTW. We have shown in Figure 5.4 how the data from different class labels are better clustered in the DTW space compared to the Euclidean space. These results demonstrate that DTW suits better the time-series domain as generated adversarial examples lack true-label guarantees. Moreover, Euclidean distance based attacks can potentially create adversarial examples that are *inconsistent* for adversarial training. Our analysis showed that for datasets such as `WISDM`, there are time-series signals from different classes with l_2-distances ≤ 2, while PGD or FGS require $\epsilon \geq 2$ to create successful adversarial examples for more than 70% time-series instances. We provide in the Appendix D an additional visualization of the adversarial examples using DTW.

Admissible alignment paths. The main property of DTW alignment is the one-to-many match between time-steps to identify similar warped pattern. Intuitively, if an alignment path matches few time-steps from the first signal with too many steps in the second signal, both signals are not considered similar. Consequently, the optimal path would be close to the corners of the cost matrix. Figure 5.9 provides a comparison between two adversarial signals generated using a green colored path closer to the diagonal vs. a red colored path that is close to the corners. We can see that the red path produces an adversarial example that is not similar to the original input. Hence, we limit the range of the random path P_{rand} used to a safe range omitting the cells at the top and bottom halves of the top-left and bottom-right corners.

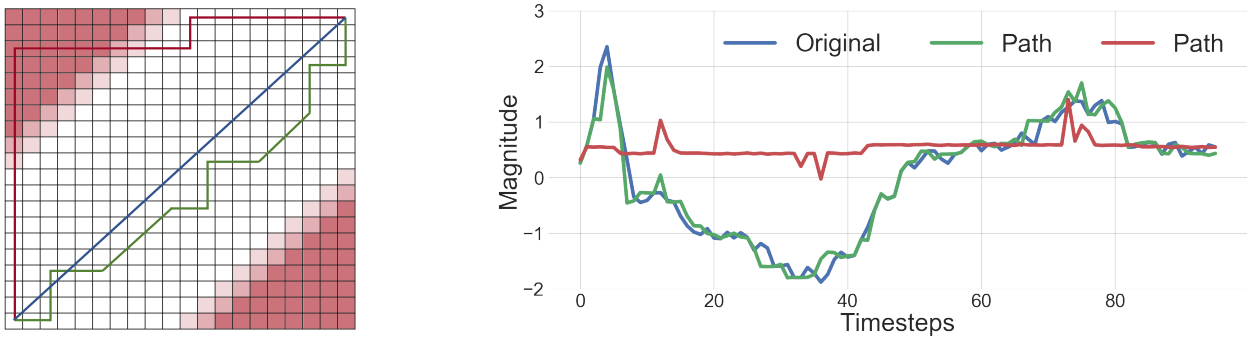


Figure 5.9 Effect of alignment path on adversarial example.

Multiple diverse adversarial examples using DTW-AR. In section 3.2, we argued that using stochastic alignment paths, we can create multiple diverse adversarial time-series examples within the same DTW measure bound. DTW-AR method leverages the large pool of candidate alignment paths to uncover more than one adversarial example as illustrated in Figure 5.5. To further test this hypothesis, we perform the following experiment. We sample a subset of different (using `PathSim`) alignment paths $\{P_{rand}\}_i$ and execute DTW-AR algorithm to create adversarial examples for the same time-series X. Let $X_{adv,i}$ be the adversarial example generated from X using $P_{rand,i}$. We measure the similarities between the generated $\{X_{adv}\}_i$ using DTW and l_2 distance. If the distance between two adversarial

Table 5.2 Average percentage of dissimilar adversarial examples created by DTW-AR using stochastic alignment paths for a given time-series. The threshold ϵ_{sim} determines whether two adversarial examples are dissimilar or not based on l_2 and DTW measures.

	ϵ_{sim} l_2 norm			ϵ_{sim} DTW		
	0.01	0.05	0.1	0.01	0.05	0.1
Atrial Fibrillation	98%	90%	87%	100%	100%	98%
Epilepsy	99%	96%	93%	100%	100%	97%
ERing	99%	95%	93%	100%	100%	98%
Heartbeat	99%	94%	92%	100%	99%	98%
RacketSports	99%	94%	92%	100%	100%	96%

examples is less than a threshold ϵ_{sim}, then they are considered the same adversarial example. Table 5.2 shows the percentage of adversarial examples generated using different alignment paths from a given time-series signal that are not similar to any other adversarial example. We conclude that DTW-AR algorithm indeed creates multiple different adversarial examples from a single time-series signal for the same DTW measure bound.

Empirical justification for Theorem 2. We provided a proof for the gap between creating an adversarial example using the proposed DTW-AR algorithm and an ideal DTW algorithm. In Figure 5.7(a), we provide an illustration of the optimal alignment path update using DTW-AR. This experiment was performed on the ECG200 dataset as an example (noting that we observed similar patterns for other datasets as well): the blue path represents the selected random path to be used by DTW-AR and the red path represents the optimal alignment path computed by DTW. At the beginning, the optimal alignment path (dotted path) and the random path are dissimilar. However, as the execution of DTW-AR progresses, the updated adversarial example yields to an optimal alignment path similar to the random path. In Figure 5.7(b), we show the progress of the `PathSim` score as a function of the iteration numbers of Algorithm 1. This figure shows the convergence of the `PathSim` score to

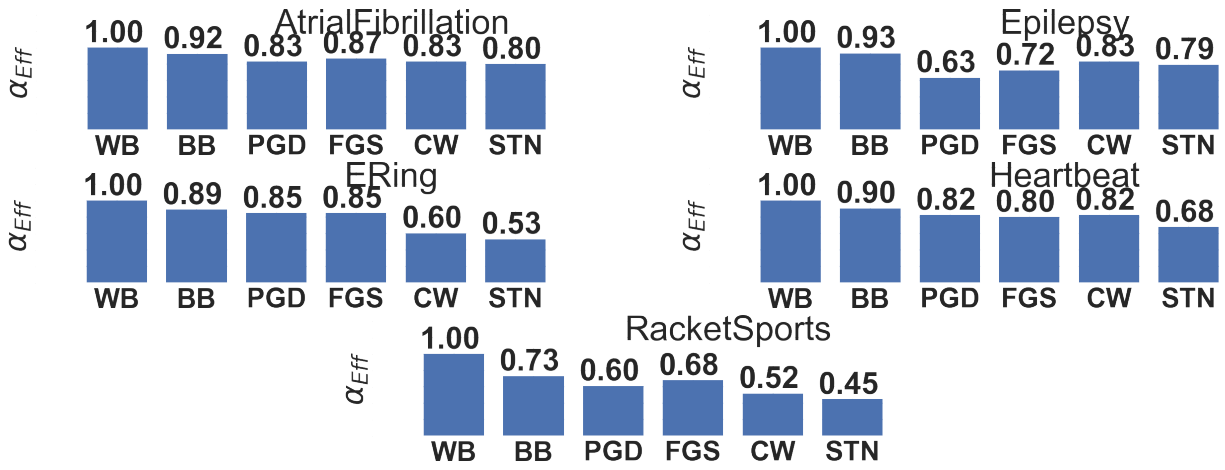


Figure 5.10 Results for the effectiveness of adversarial examples from DTW-AR on different DNNs under white-box (WB) and black-box (BB) settings, and using adversarial training baselines (PGD, FGS, CW and STN) on different datasets.

0. These strong results confirm the main claim of Theorem 2 that the resulting adversarial example X_{adv} from the minimization over $dist_{P_{rand}}(X, X_{adv})$ is equivalent to minimizing over $DTW(X, X_{adv})$!

Loss function scaling. As the final loss function is using two different terms to create adversarial attacks, the absence of a scaling parameter can affect the optimization process. In Figure 5.11, we demonstrate that empirically, the first term of the Equation $\star$ plateaus at ρ before minimizing $\mathcal{L}^{DTW}$. The figure shows the progress of both $\mathcal{L}^{label}$ and $\mathcal{L}^{DTW} = \alpha_1 \times dist_P(X, X_{adv})$ over the first 100 iterations of DTW-AR algorithm. We conclude that there is no need to scale the loss function noting that our findings were similar for other time-series datasets. In the general case, if a given application requires attention to scaling both terms ($\mathcal{L}^{DTW}$ and $\mathcal{L}^{label}$), the learning rate can be adjusted to two different values: Instead of having $\eta * \nabla L = \eta * \nabla L^{label} + \eta * \nabla L^{DTW}$, we can use a learning rate pair $\eta = (\eta_1, \eta_2)$ and gradient descent step becomes $\eta * \nabla L = \eta_1 * \nabla L^{label} + \eta_2 * \nabla L^{DTW}$.

Effectiveness of adversarial attacks. Results of the fooling rate of DTW-AR generated attacks for different models are shown in Figure 5.10. We observe that under the white-box setting (WB model), we have α_{Eff}=1. This shows that for any y_{target}, DTW-AR successfully

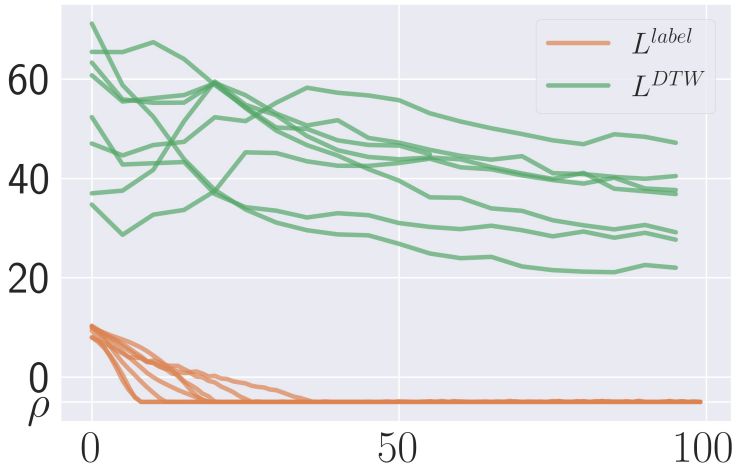
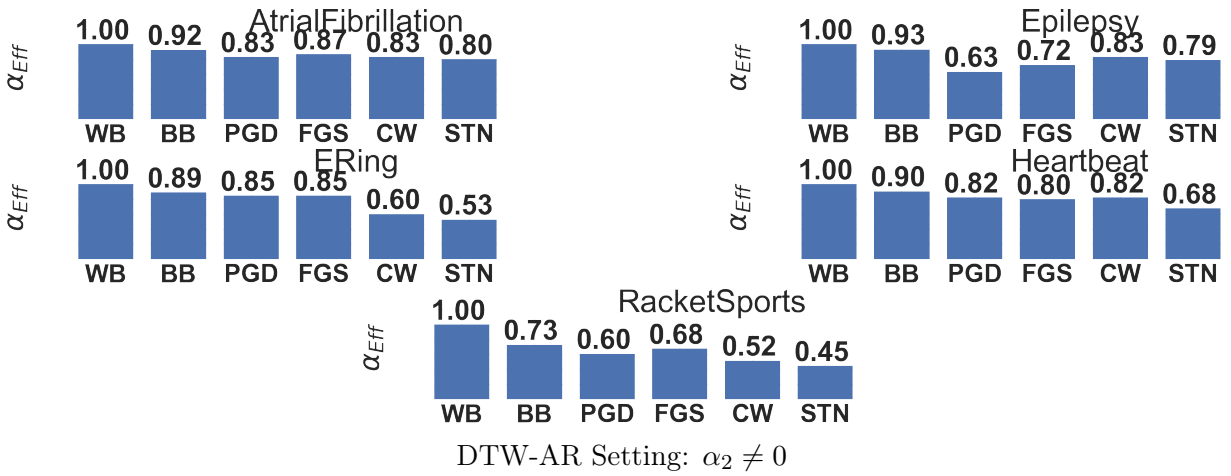


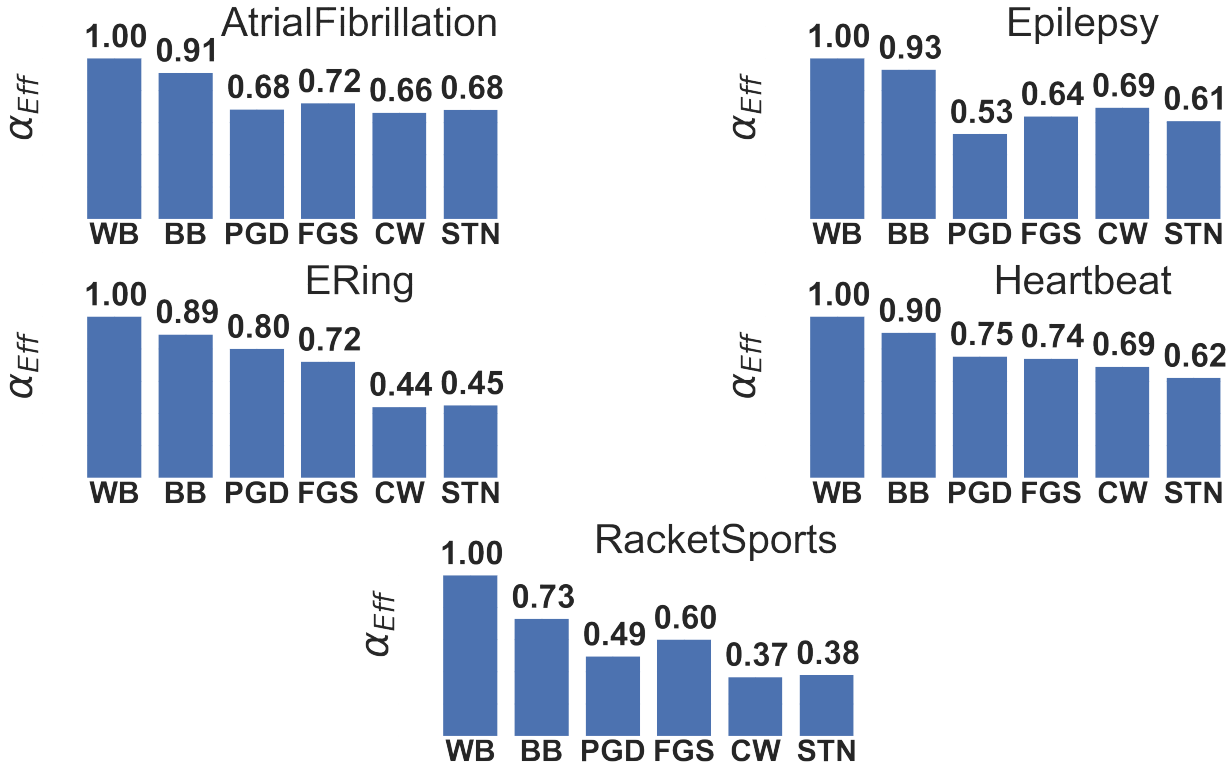
Figure 5.11 The progress of loss function values over the first 100 iterations of DTW-AR on different examples from AtrialFibrillation datasets. We observe that empirically, the Equation $\star$ plateaus at ρ before minimizing L^{DTW}.

generates an adversarial example for every input in the dataset. For black-box setting (BB model) and other models using baseline attacks for adversarial training, we see that DTW-AR attack is highly effective for most cases. We conclude that these results support the theoretical claim made in Theorem 1 by showing that standard l_2-norm based attacks have blind spots and the DTW bias is appropriate for time-series. The importance of α_2 in Equation 5.7 is shown to improve the fooling rate of adversarial examples. To implement DTW-AR adversarial attacks, we have fixed $\alpha_1 = 0.5$ and $\alpha_2 = 0.5$ for Equation 5.7. The importance of α_2 is to push the algorithm to create adversarial examples out of the scope of the Euclidean space as shown in Figure 5.12. The adversarial examples with $\alpha_2 \neq 0$ evade DNNs with adversarial training baselines better than the examples with $\alpha_2 = 0$.

DTW-AR based adversarial training. Our hypothesis is that l_2-based perturbations lack true-label guarantees and can degrade the overall performance of DNNs. Figure 5.13 shows the accuracy of different DNNs after adversarial training on clean data. This performance is relative to the clean testing set of each dataset. We observe that all l_2-based methods degrade the performance using adversarial training for at least one dataset. However, for many datasets, the performance is visibly improved using DTW-AR based adversarial training.



DTW-AR Setting: $\alpha_2 \neq 0$



DTW-AR Setting: $\alpha_2 = 0$

Figure 5.12 Results for the effectiveness of adversarial examples from DTW-AR on different DNNs under white-box (WB) and black-box (BB) settings, and using adversarial training baselines (PGD, FGS, CW and STN) on different datasets under two attack settings:$\alpha_2 \neq 0$ and $\alpha_2 = 0$.

Compared to standard training (i.e., no augmented adversarial examples), the performance on `AtrialFibrillation` improved using DTW-AR while it declined with other methods; and on `HeartBeat`, DTW-AR based training improves from 70% to 75%. To evaluate the

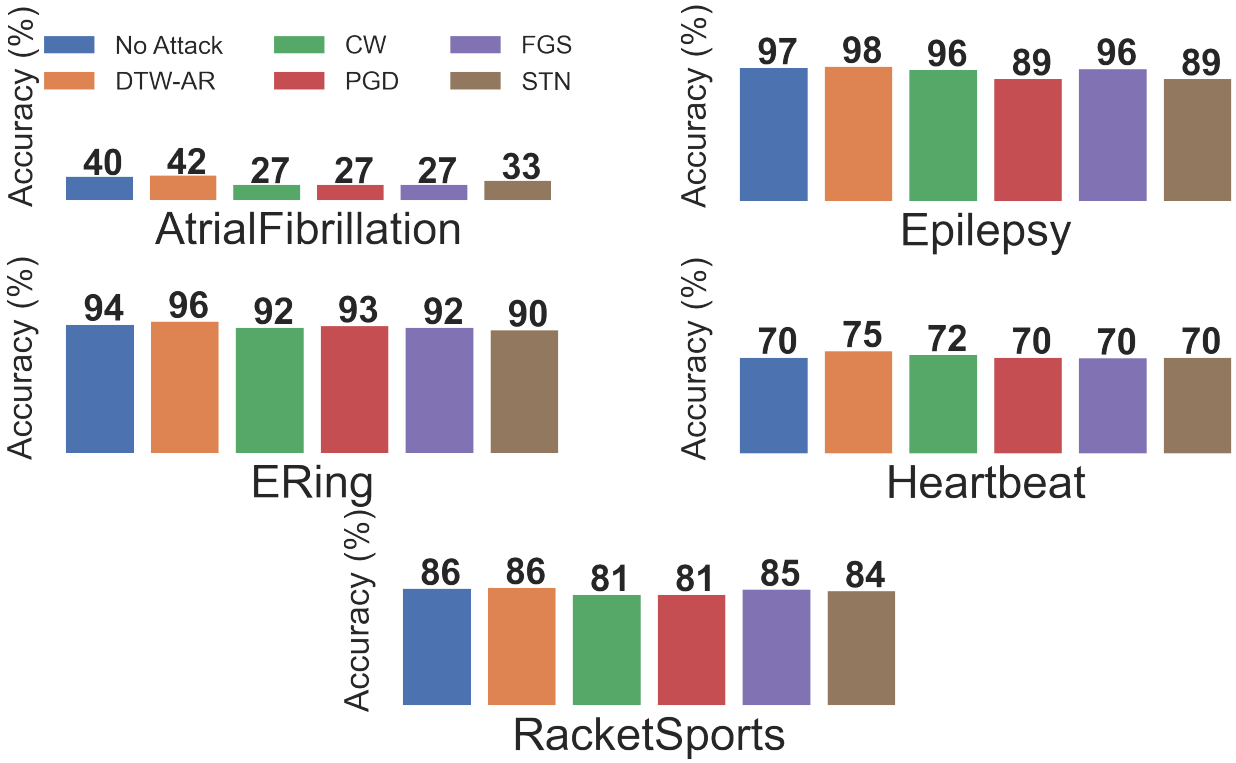


Figure 5.13 Results of adversarial training using baseline attacks and DTW-AR, and comparison with standard training without adversarial examples (No Attack) to classify clean data.

accuracy of DTW-AR in predicting the ground-truth label of adversarial examples, we create adversarial examples using a given attack algorithm and label each example with the true class-label of the corresponding clean time-series input. Figure 5.14 shows the results of DTW-AR based adversarial training using WB architecture. In this experiment, we consider the adversarial examples that have successfully fooled the original DNN (i.e., no adversarial training). We observe that DNNs using DTW-AR for adversarial training are able to predict the original label of adversarial examples with high accuracy. We can see how FGS and PGD attacks cannot evade the DTW-AR based trained deep model for almost any dataset. These results show that DTW-AR significantly improves the robustness of deep models for time-series data to evade attacks generated by DTW-AR and other baseline attacks. For the adversarial training, we employ several values to create adversarial examples to be used in the training phase. We have set $\alpha_1 \in [0.1, 1]$ and $\alpha_2 \in [0, 1]$. In Figure 5.15, we show the

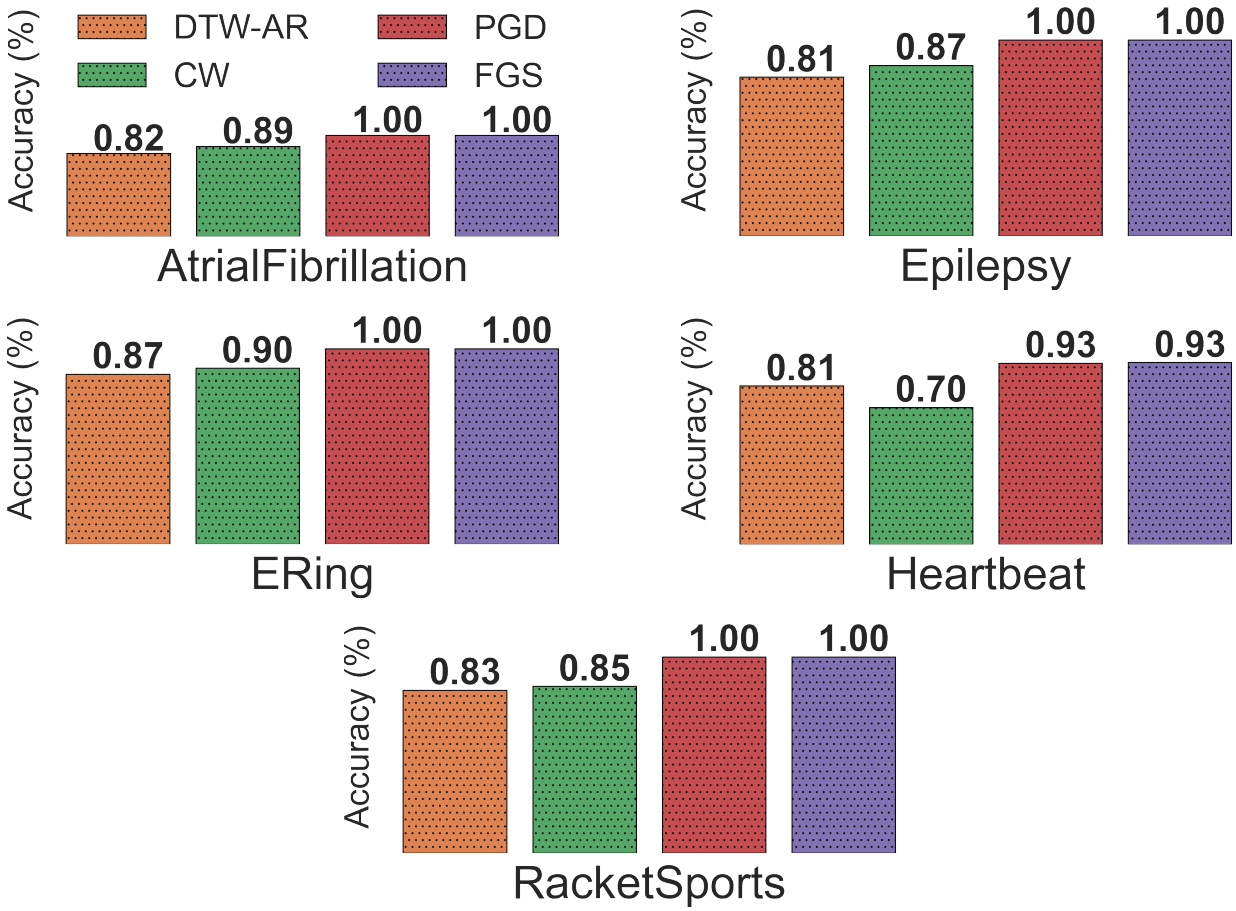
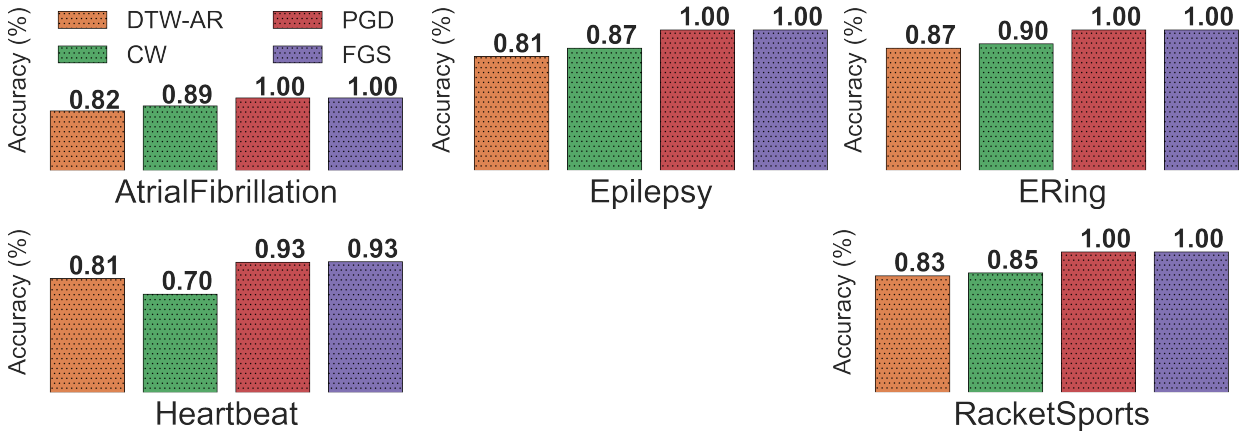
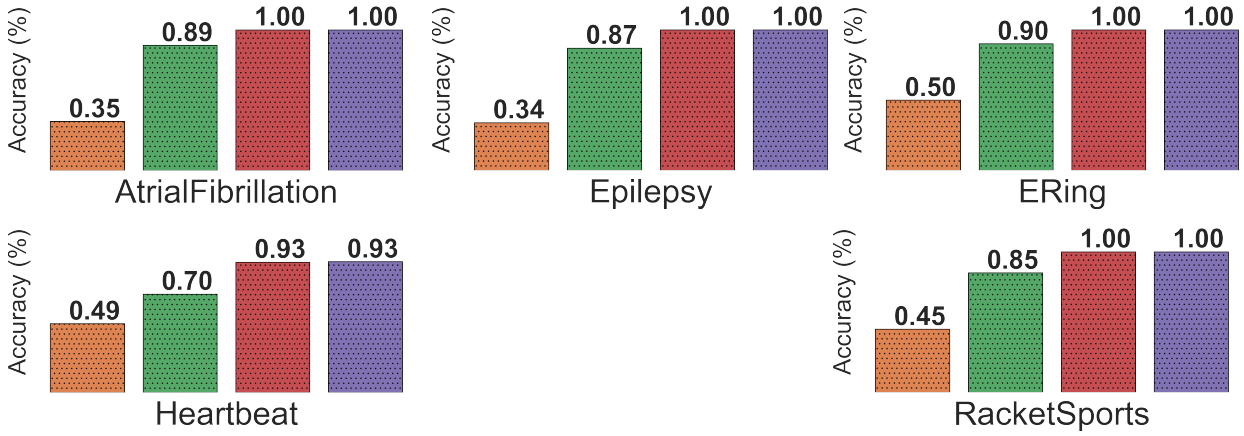


Figure 5.14 Results of DTW-AR based adversarial training to predict the true labels of adversarial examples generated by DTW-AR and the baseline attack methods. The adversarial examples considered are those which successfully fooled DNNs that do not use adversarial training.

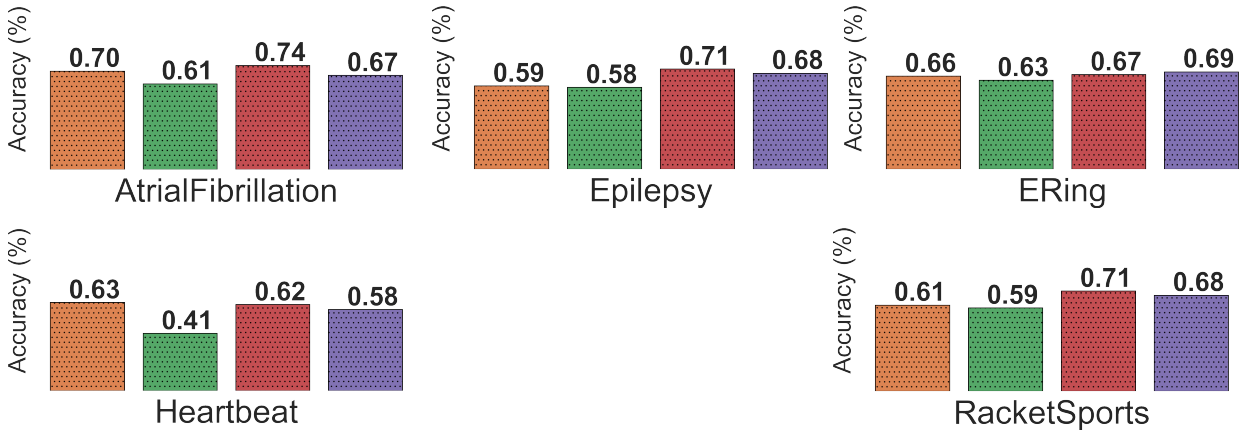
role of the term α_2 of Equation 5.7 in the robustness of the DNN. $\alpha_2 \in [0, 1]$ ensures diverse DTW-AR examples to increase the robustness of a given DNN. When set to 0, we see that there is no significant difference in the performance against baseline attacks. However, the DNN cannot defend against all DTW-AR attacks. We can also observe that the setting where α_2 is strictly different than 0 is the worst, as the DNN does not learn from the adversarial examples that are found in the Euclidean space by DTW-AR or the given baselines.



DTW-AR Setting: $\alpha_2 \in [0, 1]$



DTW-AR Setting: $\alpha_2 = 0$



DTW-AR Setting: $\alpha_2 \neq 0$

Figure 5.15 Results of DTW-AR based adversarial training to predict the true labels of adversarial examples generated by DTW-AR and the baseline attack methods. The adversarial examples considered are those that successfully fooled DNNs that do not use adversarial training.

Naive approach: Carlini & Wagner with Soft-DTW. Recall that naive approach uses DTW measure within the Carlini & Wagner loss function. SoftDTW (Marco Cuturi and Blondel, 2017) allows us to create a differentiable version of DTW measure. Hence, we provide results for this naive approach to verify if the the use of Soft-DTW with existing Euclidean distance based methods can solve the challenges for the time-series domain mentioned in this work. We compare the DTW-AR algorithm with the CW-SDTW that plugs Soft-DTW within the Carlini & Wagner algorithm instead of the standard l_2 distance. CW-SDTW has the following limitations when comapred against DTW-AR:

- The time-complexity of Soft-DTW is quadratic in the dimensionality of time-series input space, whereas the distance computation in DTW-AR is linear.
- The CW-SDTW attack method is a sub-case of the DTW-AR algorithm. If DTW-AR algorithm uses the optimal alignment path instead of a random path, the result will be equivalent to a CW-SDTW attack.
- For a given time-series signal, CW-SDTW will output one single adversarial example and cannot uncover multiple adversarial examples which meet the DTW measure bound. However, DTW-AR algorithm gives the user control over the alignment path and can create multiple diverse adversarial examples.

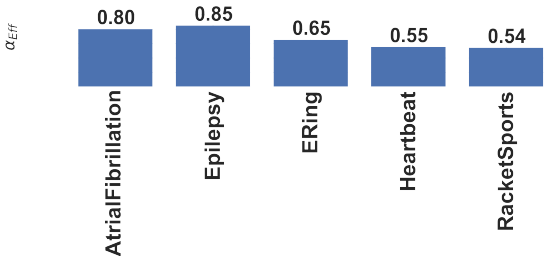


Figure 5.16 Results for the effectiveness of adversarial examples from DTW-AR against adversarial training using examples created by CW-SDTW on different datasets.

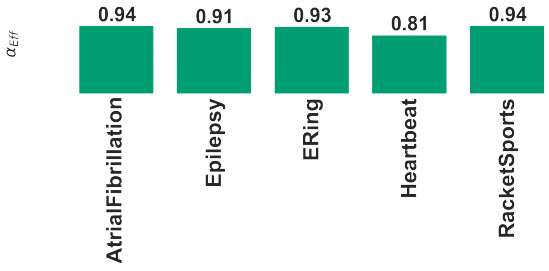


Figure 5.17 Results for the effectiveness of adversarial training using DTW-AR based examples against adversarial attacks from CW-SDTW on different datasets.

In conclusion, both challenges that were explained in the *Challenges of Naive approach* in Section 3.1 cannot be solved using CW-SDTW. As a consequence, the robustness goal aimed by this work cannot be achieved using solely CW-SDTW. Indeed, our experiments support this hypothesis. Figure 5.16 shows that DTW-AR is successful to fool a DNN that uses adversarial examples from CW-SDTW for adversarial training. This shows that our proposed framework is better than this naive baseline. Figure 5.17 shows that DTW-AR significantly improves the robustness of deep models for time-series as it is able to evade attacks generated by CW-SDTW. Both these experiments demonstrate that CW-SDTW is neither able to create stronger attacks nor a more robust deep model when compared to DTW-AR.

Comparison with Karim et al., (Karim, Majumdar, and Darabi, 2020). The approach from Karim et al., (Karim, Majumdar, and Darabi, 2020) employs network distillation to train a student model for creating adversarial attacks. However, this method is severely limited: only a small number of target classes yield adversarial examples and the method does not guarantee the generation of an adversarial example for every input. Karim et al., have shown that for many datasets, this method creates a limited number of adversarial examples in the white-box setting. To test the effectiveness of this attack against DTW-AR, Figure 5.18 shows the success rate of deep model from DTW-AR based adversarial training to predict the true labels of the attacks generated by the method from Karim et al., on different datasets.

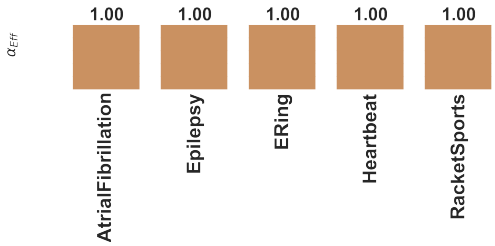


Figure 5.18 Results of the success rate of deep model from DTW-AR based adversarial training to predict the true label of adversarial attacks generated using method in (Karim, Majumdar, and Darabi, 2020).

DTW-AR outperforms (Karim, Majumdar, and Darabi, 2020) due to following reasons:

- DTW-AR generates at least one adversarial example for every input $X \in \mathbb{R}^{n \times T}$ as shown in our experiments.
- Adversarial examples created by DTW-AR are highly effective against deep models relying on (Karim, Majumdar, and Darabi, 2020) for adversarial training as this baseline fails to create adversarial examples for many inputs and target classes (shown in (Karim, Majumdar, and Darabi, 2020)).
- Adversarial examples created by the method from (Karim, Majumdar, and Darabi, 2020) does not evade deep models from DTW-AR based adversarial training.

Computational runtime of DTW-AR vs. DTW. As explained in the technical section, optimization based attack algorithm requires a large number of iterations to create a highly-similar adversarial example. For example, 10^3 iterations is the required default choice for CW to create successful attacks, especially, for large time-series in our experiments. The exact DTW method is non-differentiable, thus, it is not possible to perform experiments to compare DTW-AR method to the exact DTW method. Hence, we assume that each iteration will compute the optimal DTW path and use it instead of the random path. To assess the runtime of computing the DTW measure, we employ three different approaches: 1) The standard DTW algorithm, 2) The FastDTW (Salvador and Chan, 2007) that was introduced to overcome DTW computational challenges, and 3) cDTW (Dau, Silva, et al., 2018) that measures DTW in a constrained manner using warping windows. We note that FastDTW was proven to be inaccurate, and cDTW is faster and more accurate for computing DTW measure (R. Wu and Keogh, 2020). We show both baselines for the sake of completeness. We provide the runtime of performing each iteration using the different algorithms in Figure 5.19. We can clearly observe that DTW-AR is orders of magnitude faster than the standard DTW and the accelerated DTW algorithms. The overall computational cost will be significantly

reduced using DTW-AR compared to exact DTW or soft-DTW (Marco Cuturi and Blondel, 2017) for large-size time-series signals.

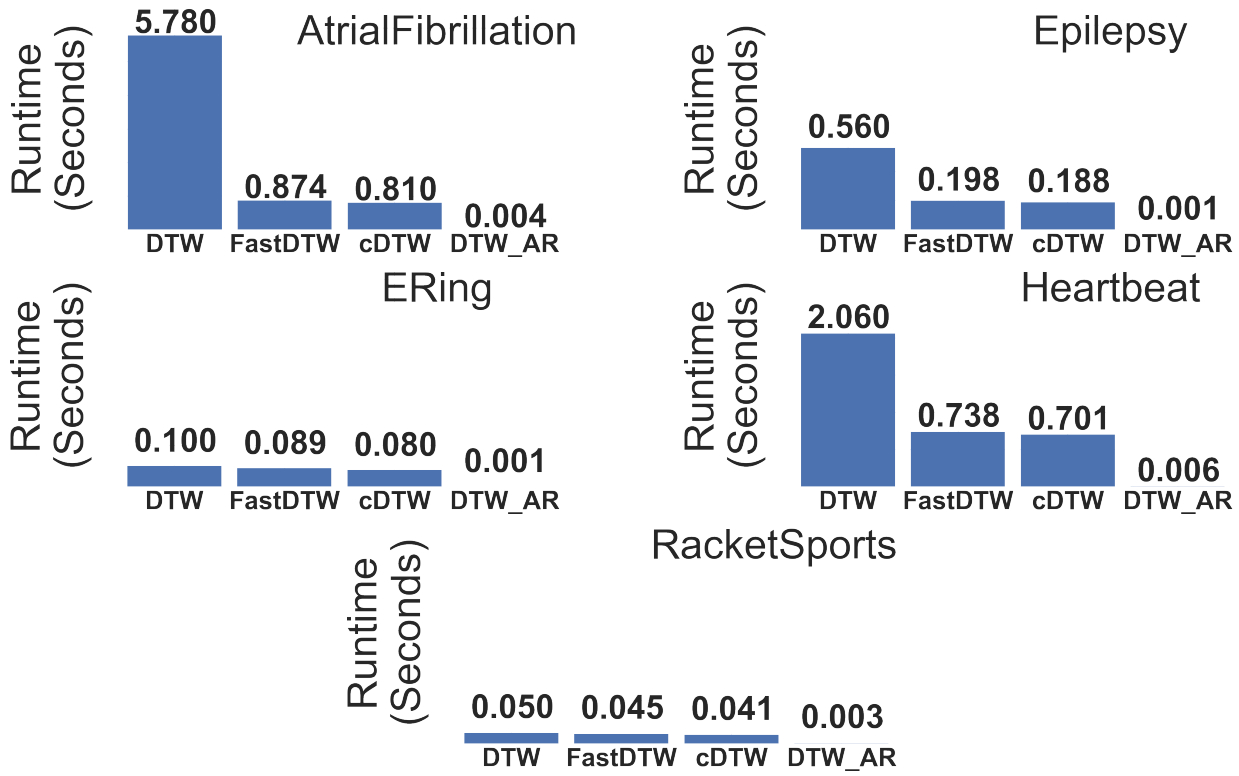


Figure 5.19 Average runtime **per iteration** for standard DTW, FastDTW, cDTW, and DTW-AR (on NVIDIA Titan Xp GPU).

DTW-AR extension to other multivariate DTW measures. The DTW-AR framework relies on the distance function $dist_P(X, Z) = \sum_{(i,j)\in P} d(X_i, Z_j)$ between two time-series signals X and Z according to an alignment path P to measure their similarity. Extending the DTW notion from univariate to multivariate is a known problem, where depending on the application, researchers' suggest to change the definition of $dist_P(X, Z)$ to better fit the characteristics of the application at hand. In all cases, DTW-AR relies on using the final cost matrix $DTW(X, Z) = \min_P dist_P(X, Z)$ using dynamic programming $C_{i,j} = d(X_i, Z_j) + \min\{C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}\}$. Therefore, the use of different variants of $dist_P(X, Z)$ (e.g., DTW_I or DTW_D (Shokoohi-Yekta et al., 2017)) will only affect the cost matrix values, but will not change the assumptions and applicability of DTW-AR. Therefore, DTW-AR is

general and can work with any variant of DTW. In Figure 5.20, we demonstrate that using a different family of DTW (DTW_I) does not have a major impact on DTW-AR's performance and effectiveness. The performance of DTW-AR framework using both alternative measures of multi-variate DTW does not affect the overall performance. Therefore, for a given specific application, the practitioner can configure DTW-AR appropriately.

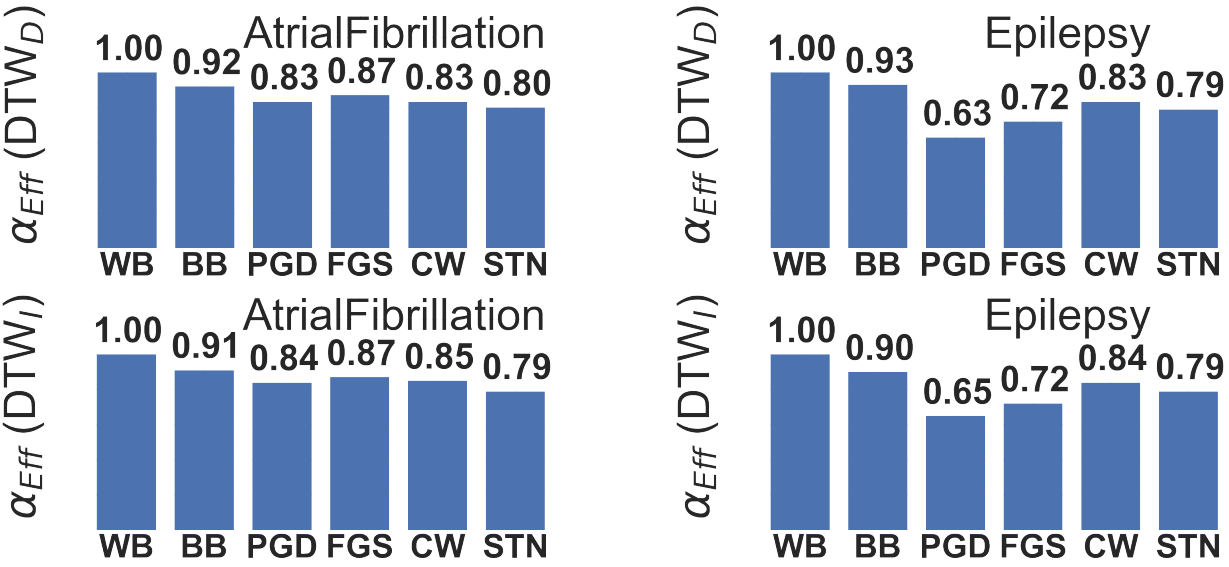


Figure 5.20 Results for the effectiveness of adversarial examples from DTW-AR using DTWAdaptive(Shokoohi-Yekta et al., 2017) (DTW_D top row, DTW_I bottom row) on different DNNs under different settings.

5.3.3 Summary of Experimental Results

Our experimental results supported all the claims made in Section 3. The summary list includes:

- Figure 5.4 showed that DTW space is more suitable for adversarial studies in the time-series domain than Euclidean distance to support Theorem 3.
- Using stochastic alignment paths, DTW-AR creates multiple diverse adversarial examples to support Corollary 1 (Table 5.2), which is impossible using the optimal alignment path.

- Figure 5.7 provides empirical justification for Theorem 4 showing that minimizing over a given alignment path is equivalent to minimizing using exact DTW method (bound is tight).
- Figure 5.10 shows that adversarial examples created by DTW-AR have higher potential to break time-series DNN classifiers.
- Figures 5.13 and 5.14 show that DTW-AR based adversarial training is able to improve the robustness of DNNs against baseline adversarial attacks.
- Figure 5.16 and 5.17 shows that DTW-AR outperforms the naive approach CW-SDTW that uses SoftDTW with the Carlini & Wagner loss function. We also demonstrated several limitations of CW-SDTW to achieve the robustness goal aimed by this work.
- Figure 5.19 clearly demonstrates that DTW-AR significantly reduces the computational cost compared to existing approaches of computing the DTW measure for creating adversarial examples.
- Figure 5.20 demonstrates that DTW-AR can generalize to any multivariate DTW measure (such as DTWAdaptive (Shokoohi-Yekta et al., 2017)) without impacting on its performance and effectiveness.

5.4 Summary

We introduced in this chapter the DTW-AR framework that studies adversarial robustness of deep models for the time-series domain using dynamic time warping measure. This framework creates effective adversarial examples by overcoming the limitations of prior methods based on Euclidean distance. We theoretically and empirically demonstrate the effectiveness of DTW-AR to fool deep models for time-series data and to improve their robustness. We conclude that the time-series domain needs focused investigation for studying robustness of deep models by shedding light on the unique challenges.

CHAPTER SIX

MIN-MAX OPTIMIZATION FOR TRAINING ROBUST DEEP MODELS FOR TIME-SERIES DOMAIN

T. Belkhouja, Y. Yan, and J. Doppa. "Training Robust Deep Models for Time-Series Domain: Novel Algorithms and Theoretical Analysis". *Proceedings of 36th AAAI Conference on Artificial Intelligence*, 2022.

Originally published in the *Proceedings of 36th AAAI Conference on Artificial Intelligence*.

Attributions:

T. Belkhouja has contributed to this work by formulating the problem setting, investigating the state of the art, formulating the theoretical contribution and the algorithmic solution, implementing the algorithm and running the required empirical analysis to highlight the performance improvement of the proposed solution compared to the state of the art.

Y. Yan has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the formulation of the theory of the proposed solution, and in writing the scientific manuscript.

J. Doppa has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

MIN-MAX OPTIMIZATION FOR TRAINING ROBUST DEEP MODELS FOR TIME-SERIES DOMAIN

In this chapter, we propose a novel and principled framework referred as **RO***bust Training for* **T***ime-***S***eries (RO-TS)* to create robust DNNs for time-series data. We employ additive noise variables to simulate perturbations within a small neighborhood around each training example. We incorporate these additive noise variables to formulate a *min-max* optimization problem to reason about the robustness criteria in terms of disturbances to time-series inputs by minimizing the worst-case risk. To capture the special characteristics of time-series signals, we employ the global alighnment kernel (GAK) based distance (M. Cuturi et al., 2007) to define neighborhood regions for training examples. We show the generality and advantage of our formulation using the summation structure over time-series alignments by relating both GAK and dynamic time warping (DTW) (Berndt and Clifford, 1994). To efficiently solve this family of optimization problems, we develop a principled *stochastic compositional alternating gradient descent ascent (SCAGDA)* algorithm by carefully leveraging the underlying structure of this problem. Another key computational challenge is that time-series distance measures including the GAK based distance involve going through all possible alignments between pairs of time-series inputs, which is expensive, e.g., $O(T^2)$ for GAK where T is the length of time-series signal. As a consequence, the computational cost grows significantly for iterative optimization algorithms where we need to repeatedly compute the distance between time-series signals.

6.1 Background and Problem Setup

Prior work on robustness of DNNs is mostly focused on image/text domains; and can be classified into two categories.

Adversarial training employs augmented data such as adversarial examples (Z. Kolter and Madry, 2018; W. Y. Wang, Singh, and J. Li, 2019) and input perturbations. Methods

to create adversarial examples include general attacks such as Carlini & Wagner attack (Carlini and D. Wagner, 2017), boundary attack (Brendel, Rauber, and Bethge, 2018), and universal attacks (Moosavi-Dezfooli, A. Fawzi, O. Fawzi, et al., 2017). Recent work regularizes adversarial example generation methods to obey intrinsic properties of images (Laidlaw and Feizi, 2019b; C. Xiao et al., 2018; Hosseini et al., 2017). There are also specific adversarial methods for NLP domain (Samanta and Mehta, 2017; J. Gao et al., 2018).

Training via explicit loss function employ an explicit loss function to capture the robustness criteria and optimize it. Stability training (S. Zheng et al., 2016; B. Li et al., 2019) for images is based on the criteria that similar inputs should produce similar DNN outputs. Adversarial training can be interpreted as min-max optimization, where a hand-designed optimizer such as projected gradient descent is employed to (approximately) solve inner maximization. (Xiong and Hsieh, 2020) train a neural network to guide the optimizer. Since characteristics of time-series (e.g., fast-pace oscillations, sharp peaks) are different from images/text, L_p distance based methods are not suitable for time-series domain.

In summary, there is no prior work to train robust DNNs for time-series domain in a principled manner. This work precisely fills this important gap in our scientific knowledge.

We consider the problem of learning *robust* DNN classifiers over time-series data. We are given a training set of n input-output pairs $\{(x_i, y_i)\}_{i=1}^{n}$. Each input $x_i \in \mathcal{X}$ is a time-series signal, where $\mathcal{X} \subseteq \mathbb{R}^{C \times T}$ with C denoting the number of channels and T being the window-size of the signal; and $y_i \in \mathcal{Y}$ is the associated ground-truth label, where $\mathcal{Y} \in \{1, \cdots, \mathcal{C}\}$ is a set of $\mathcal{C}$ discrete class labels. Traditional empirical risk minimization learns a DNN classifier $f : \mathcal{X} \times \Theta \to \mathcal{Y}$ with weights $w \in \Theta$ that maps time-series inputs to classification labels for a hypothesis space Θ and a loss function ℓ:

$$\min_{w \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i, w), y_i).$$

Training for robustness. We would like the learned classifier $f(x, w)$ to be robust to disturbances in time-series inputs due to noisy observations or adversarial attacks. For ex-

ample, a failure in the prediction task for the above health monitoring application due to such disturbances can cause injury to the patient without the system notifying the needed assistance. Therefore, we want the trained DNN classifier to be invariant to such disturbances. Mathematically, for an appropriate distance function $d(x, x')$ over time-series inputs x and x', we want the classifier f to predict the same classification label as x for all inputs x' such that $d(x, x') < \varepsilon$, where ε stands for the bound on allowed disturbance to input x. This goal can be achieved by reasoning about the worst-case empirical risk over possible perturbations $a_i \in \mathbb{R}^{C \times T}$ of x_i such that $d(x_i, x_i + a_i) \leq \varepsilon$. The resulting *min-max* optimization problem is given below.

$$\begin{aligned} \min_{w \in \Theta} \quad & \frac{1}{n} \sum_{i=1}^{n} \max_{a_i} \ell(f(x_i + a_i, w), y_i) \\ \text{s.t.} \quad & d(x_i, x_i + a_i) \leq \varepsilon \end{aligned} \tag{6.1}$$

In practice, instead of solving the above hard constrained problem, one can solve an equivalent soft constrained problem using regularization as follows

$$\min_{w \in \Theta} \max_{a_i} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i + a_i, w), y_i) - \lambda d(x_i, x_i + a_i) \tag{6.2}$$

There is a natural interpretation of this optimization problem. The *inner maximization* problem serves the role of an attacker whose goal is to find adversarial examples that achieves the largest loss. The *outer minimization* problem serves the role of a defender whose goal is to find the parameters of the deep model w by minimizing the adversarial loss from the inner attack problem. This formulation is applicable to all types of data by selecting an appropriate distance function d. For example, L_p-norm distance is usually used in the image domain (Z. Kolter and Madry, 2018).

Typical stochastic approaches to solving the above adversarial training problem include alternating stochastic optimization (Junchi Y., 2020) and stochastic gradient descent ascent (GDA) (T. Lin, Jin, and Jordan, 2020; Y. Yan et al., 2020). The alternating method first fixes w and solves the inner maximization approximately to get each a_i (e.g., using stochastic

gradient descent). Next, a_i is fixed and the outer minimization is solved over w. These two steps are performed alternatively until convergence. The GDA method computes the gradient of w and a_i simultaneously at each iteration, and then use these gradients to update w and a_i. Both methods require the ability to compute the unbiased estimation of the gradients w.r.t. w and a_i. When d is decomposable, e.g., L_p-norm, then its stochastic gradient can be easily computed. However, in time series domain, commonly used distance measures may not be decomposable, so its stochastic gradients are not accessible. Consequently, one has to calculate the exact gradient of $d(x_i, x_i + a_i)$ w.r.t, a_i. We will investigate this key challenge in Section 6.2.2.

6.2 RO-TS Algorithmic Framework

In this section, we describe the technical details of our proposed RO-TS framework to train robust DNN classifiers for time-series domain. First, we instantiate the min-max formulation with GAK based distance as it appropriately captures the similarity between time-series signals. Second, we provide an efficient algorithm to solve the GAK-based formulation to learn parameters of DNN classifers. We provide the proofs and the theoretical analysis of all proposed theorems in the Appendix C

6.2.1 Distance Measure for Time-Series

Unlike images and text, time-series data exhibits unique characteristics such as sparse peaks, fast oscillations, and frequency/time shifting which are important for pattern matching and data classification. Hence, measures such as Euclidean distance that do not account for these characteristics usually fail in recognizing the similarities between time-series signals. To address this challenge, elastic measures have been introduced for pattern-matching tasks for time-series domain (M. Cuturi et al., 2007), where one time-step of a signal can be associated with many time-steps of another signal to compute the similarity measure.

Time-series alignment. Given two time series x=$(x_1, \cdots, x_{T_1})$ and x'=$(x'_1, \cdots, x'_{T_2})$ for $T_1, T_2 \in \mathbb{N}_+$, the alignment $\pi = (\pi_1, \pi_2)$ is defined as a pair of increasing integral vectors of length $r \leq T_1 + T_2 - 1$ such that $1 = \pi_1(1) \leq \cdots \leq \pi_1(r) = T_1$ and $1 = \pi_2(1) \leq \cdots \leq \pi_2(r) = T_2$ with unitary increments and without simultaneous repetitions, which presents the coordinates of x and x'. This alignment defines the one-to-many alignment between x and x' to measure their similarity. Using a candidate alignment π, we can compute their similarity as follows:

$$d_\pi(x, x') = \sum_{i=1}^{|\pi|} \text{dist}(x_{\pi_1(i)}, x'_{\pi_2(i)}) \tag{6.3}$$

where $|\pi|$=r denotes the length of alignment and $\text{dist}(\cdot, \cdot)$ in the above equation is the Minkowski distance:

$$\text{dist}(x_{\pi_1(i)}, x'_{\pi_2(i)}) = \|x_{\pi_1(i)} - x'_{\pi_2(i)}\|_p, \ p \in \{1, \cdots, \infty\}$$

Global alignment kernel (GAK) based distance. The concept of alignment allows us to take into consideration the intrinsic properties of time-series signals, such as frequency shifts, to compute their similarity. There are some well-known approaches to define distance metrics using time-series alignment. For example, dynamic time warping (DTW) (Berndt and Clifford, 1994) selects the alignment with the minimum distance:

$$D_{\text{DTW}}(x, x') = \min_{\pi \in \mathcal{A}} d_\pi(x, x'),$$

where $\mathcal{A}$ denotes the set of all possible alignments.

While DTW only takes into account one candidate alignment, global alignment kernel (GAK) (M. Cuturi et al., 2007) takes all possible alignments into consideration:

$$k_{\text{GAK}}(x, x') = \sum_{\pi \in \mathcal{A}} \exp\Big(- \frac{d_\pi(x, x')}{\nu} \Big) \tag{6.4}$$

where ν is a hyper-parameter and $d_\pi(\cdot, \cdot)$ is defined in Equation (6.3). In practice, to handle the diagonally dominance issue (M. Cuturi et al., 2007; L. Wu et al., 2018; M. Cuturi, 2011),

$D_{\text{GAK}} := -\nu \log(k_{\text{GAK}})$ is typically used as a distance measure for a pair of time-series signals. GAK enjoys several advantages over DTW (M. Cuturi, 2011): (i) differentiable, (ii) positive definite, (iii) coherent measure over all possible alignments. Therefore, k_{GAK} (or D_{GAK}) is a better fit to train robust DNNs for the time-series domain.

On the other hand, GAK can also be a more *general* measure than DTW due to its summation structure, as $\lim_{\nu \to 0} D_{\text{GAK}}(x, x') = D_{\text{DTW}}(x, x')$, i.e., arbitrarily close to DTW by changing ν. The following proposition shows the tight approximation of the soft minimum of GAK to the hard minimum of DTW.

Proposition 1. *For a time-series pair (x, x'), we have:*

$$0 \leq D_{DTW}(x, x') - D_{\text{GAK}}(x, x') \leq \nu \log(|\mathcal{A}|).$$

As shown, D_{GAK} converges to D_{DTW} in $\nu \log(|\mathcal{A}|)$ as ν decreases. Due to the above advantages and the approximation ability of D_{GAK} to D_{DTW}, we consider the more *general* k_{GAK} and D_{GAK} in our RO-TS method.

6.2.2 SCAGDA Optimization Algorithm

By plugging k_{GAK} from Equation (6.4) to replace d into the min-max formulation in Equation (6.2), we reach the following objective function of our RO-TS framework:

$$\min_{w \in \Theta} \max_{a_i} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i + a_i, w), y_i) + \lambda \overbrace{\log\left(k_{\text{GAK}}(x_i, x_i + a_i)\right)}^{=d(x_i, x_i + a_i)} \tag{6.5}$$

where ν outside log in D_{GAK} can be merged into λ. The above problem is decomposable over individual training examples (i.e., index i), so we can compute stochastic gradients by randomly sampling a batch of data and employ stochastic gradient descent ascent (SGDA) (T. Lin, Jin, and Jordan, 2020; Y. Yan et al., 2020), a family of stochastic algorithms for solving min-max problems.

Key challenge. The second term $\log(k_{\text{GAK}}(x_i, x_i + a_i))$ has a *compositional* structure due to the outer log function. By chain rule, its gradient w.r.t. the dual variable a_i is

$$\nabla_{a_i} \log(k_{\text{GAK}}(x_i, x_i + a_i)) = \frac{\nabla_{a_i} k_{\text{GAK}}(x_i, x_i + a_i)}{k_{\text{GAK}}(x_i, x_i + a_i)}$$

where one has to go through all possible alignments to compute k_{GAK} and $\nabla_{a_i} k_{\text{GAK}}$ (see Equation (6.4)) and there is no unbiased estimation (i.e., stochastic gradients) for it.

Consequently, at each iteration, SGDA has to compute the exact value of $k_{\text{GAK}}(x_i, x_i + a_i)$ and $\nabla_{a_i} k_{\text{GAK}}(x_i, x_i + a_i)$ according to the chain rule, which leads to an additional time-complexity of $O(CT^2)$ per SGDA iteration, where C and T denote the number of channels and window-size respectively. This computational bottleneck will lead to extremely slow training algorithm when C and/or T is large, which is the case in many real-world applications. One candidate approach to alleviate the computational challenge due to $\log(k_{\text{GAK}})$ part of the objective is to make use of the inner summation structure of k_{GAK}. Since k_{GAK} involves a summation over all alignments, as shown in (6.4), we can use only a *subset* of alignments for estimating the full summation. This procedure will give an unbiased estimation of k_{GAK}, but the outer logarithmic function makes it a *biased* estimation for $\nabla_{a_i} \log(k_{\text{GAK}}(x_i, x_i + a_i))$. However, such biased estimation violates the assumption in SGDA studies, so their theoretical analysis cannot hold.

There is another line of research investigating stochastic compositional gradient methods for minimization problems with compositional structure (M. Wang, Fang, and H. Liu, 2017; T. Chen, Sun, and Yin, 2020). However, *min-max* optimization with *compositional* structure, including our case shown in Equation (6.5), is not studied yet. It is unclear whether these techniques and analysis hold for min-max problems.

SCAGDA algorithm. We propose a novel *stochastic compositional alternating gradient descent ascent (SCAGDA)* algorithm to solve a family of nonconvex-nonconcave min-max compositional problems, which include RO-TS (Equation (6.5)) as a special case. We summarize SCAGDA in Algorithm 6. Specifically, we consider solving the following family of

Algorithm 6 SCAGDA (Stochastic Compositional Alternating Gradient Descent Ascent)

1: Initialize w_0, a_i^0 for $i = 1, \cdots, n$ and $\omega_i^0 = 0$ for $i = 1, \cdots, n$, step sizes $\{\eta_k\}_{k=1}^K$ and $\{\gamma_k\}_{k=1}^K$.

2: **for** $k = 0, \cdots, K-1$ **do**

3: Randomly sample an index i_1 to compute stochastic gradient $\nabla_w f_{i_1}(w_k, a_{i_1}^k)$

4: Set: $w_{k+1} = w_k - \eta_k \nabla_w f_i(w_k, a_{i_1}^k)$

5: Randomly sample an index i_2 to compute stochastic gradient $\nabla_a f_{i_2}(w_{k+1}, a_{i_2}^k)$

6: Randomly sample two independent indices j_1, j_2 of h_{i_2} to compute $h_{i_2,j_1}(a_{i_2}^k)$ and $\nabla h_{i_2,j_2}(a_{i_2}^k)$

7: Set:

$$\omega_i^{k+1} = \begin{cases} \omega_i^k & \text{for } i \neq i_2 \\ (1-\beta)\omega_{i_2}^k + \beta h_{i_2,j_1}(a_{i_2}^k) & \text{for } i = i_2 \end{cases} \tag{6.6}$$

8: Set: $a_{i_2}^{k+1} = a_{i_2}^k + \gamma_k(\nabla_a f_{i_2}(w_{k+1}, a_{i_2}^k) - \nabla h_{j2}(a_{i_2}^k)^\top \nabla g(\omega_{i_2}^{k+1}))$

9: **end for**

10: **return** final solution w_K

problems:

$$\min_w \max_{a_i} \frac{1}{n} \sum_{i=1}^n \phi_i(w, a_i) \tag{6.7}$$

where $\phi_i(w, a_i) := f_i(w, a_i) - g(\frac{1}{m}\sum_{j=1}^m h_{i,j}(a_i))$.

Mapping Problem (6.7) to RO-TS (6.5). As mentioned above, RO-TS for time-series in Equation (6.5) is a special case of Problem (6.7) as shown below. The variables $\phi_i(w, a_i), f_i, g, h_{i,j}$ in Problem (6.7) can be instantiated by the following mappings:

- f_i in ϕ_i of (6.7) $\Rightarrow$ the loss ℓ on the i-th data in (6.5)
- $-g(\cdot)$ in ϕ_i of (6.7) $\Rightarrow$ $\lambda \log(\cdot)$ in (6.5)
- $\frac{1}{m}\sum_{j=1}^m h_{i,j}(a_i)$ in ϕ_i of (6.7) $\Rightarrow$ $k_{\text{GAK}}(x_i, x_i + a_i) = \sum_{\pi \in \mathcal{A}} \exp(-d_\pi(x_i, x_i + a_i)/\nu)$ in (6.5), where m and j corresponds to the total number of alignment paths $|\mathcal{A}|$ and the

index of alignment path, respectively. Note that the summation form of k_{GAK} can be easily converted to an average form due to $\log(x) = \log(x/m) + \log(m)$.

Algorithmic analysis of SCAGDA. To introduce and analyze Algorithm 6 for solving Problem (6.7), we first introduce some notations. Denote $P(w) := \max_{a_i} \frac{1}{n} \sum_{i=1}^{n} \phi_i(w, a_i)$ as the *primal function* of the above min-max optimization problem, where we are interested in analyzing the convergence of the *primal gap* after the K-th iteration:

$$P(w_K) - \min_{w} P(w).$$

Let $a := (a_1, a_2, ..., a_n) \in \mathbb{R}^{C \times T \times n}$ be the concatenation of a_i for $i = 1, \cdots, n$. We also use the following notations to improve the technical exposition and ease of readability.

$$\begin{aligned}
\phi(w, a) :=& \frac{1}{n} \sum_{i=1}^{n} \phi_i(w, a_i), \\
h_i(a_i) :=& \frac{1}{m} \sum_{j=1}^{m} h_{i,j}(a_i), \\
h(a) :=& \frac{1}{n} (h_1(a_1), \cdots, h_n(a_n)),
\end{aligned}$$

where the last term $h(a)$ is the concatenation of all h_i for $i = 1, \cdots, n$.

As mentioned above while discussing the key challenge of the compositional structure in RO-TS (6.5), conventional SGDA methods for Problem (6.7) require us to compute the full gradient of the compositional part $g(h_i(a_i))$, i.e., $\nabla h_i(a_i)^\top \nabla g(h_i(a_i))$, which involves *all* alignments in the case of RO-TS.

In contrast, SCAGDA only samples a constant number of $h_{i,j}(a_i)$ over j (i.e., over a subset of alignments for RO-TS) and $\nabla h_{i,j}(a_i)$ (**Line 6**). Subsequently, SCAGDA employs a simple iterative *moving average* (MA) approach to accumulate $h_{i,j}(a_i)$ into ω_i^{k+1} at iteration k for estimating $h_i(a_i)$ (**Line 7**). The key idea behind moving average method is to control the variance of the estimation for $h_i(a_i^{k+1})$ using a weighted average from the previous estimate $h_i(a_i^k)$. Even though $\nabla g(\omega_i^{k+1})$ is a biased estimation of $\nabla g(h_i(a_i^k))$, we can still use smoothness condition (introduced in Section 6.3 later) and bound the approximation error

$\mathbb{E}[\|\omega_i^k - h_i(a_i^{k-1})\|^2]$ where $\omega_k := (\omega_1^k, ..., \omega_n^k)$ is the concatenation of all $\{\omega_i^k\}_{i=1}^n$ at iteration k, as shown in Theorem 6.

Therefore, instantiation of SCAGDA for RO-TS does not require us to perform computation over *all* alignments contained in $h_i(a_i)$ for each time-series training sample, which leads to a more efficient algorithm with high scalability on large datasets. As shown in **Line 3** and **5**, SCAGDA updates the primal variable w and dual variable a in an *alternating* scheme, which means that w_{k+1} is updated based on a_k, while a_{k+1} is updated based on w_{k+1}. This is different from SGDA, which updates a_k based on w_k instead. We instantiate SCAGDA for the proposed RO-TS framework as shown in Algorithm 7. The primal variable update is provided in Line 6, and the dual variable update is provided in Line 11. In particular, Line 8 and 8 correspond to the moving average step for estimating k_{GAK} using randomly sampled alignment subset $\widehat{\mathcal{A}}_i^k$ for the i-th time-series training example.

In the next section, we show that our algorithm can converge to primal gap $P(w_K) - \min_w P(w) \leq \epsilon$ with iteration complexity $O(1/\epsilon^2)$, where ϵ is a pre-defined threshold. *To the best of our knowledge, this is the first optimization algorithm and convergence analysis for the famaily of compositional min-max optimization problems shown in (6.7).*

6.3 Theoretical Analysis

In this section, we present novel theoretical convergence analysis for SCAGDA algorithm. As mentioned in the previous section, for the problem (6.7), existing theoretical analysis of SGDA (T. Lin, Jin, and Jordan, 2020; Y. Yan et al., 2020), stochastic alternating gradient descent ascent (SAGDA) (Junchi Y., 2020) require us to compute exact gradient of $g(h_i(a_i))$ at each iteration. On the other hand, it is unclear if stochastic compositional alternating gradient algorithms for minimization problems (M. Wang, Fang, and H. Liu, 2017; T. Chen, Sun, and Yin, 2020) can handle the complex min-max case.

Summary of results. We answer the following question: *can we establish convergence*

Algorithm 7 RO-TS Instantiation of SCAGDA

Input: A training set $\{(x, y) \in \mathcal{X} \times \mathcal{Y}\}^{n_{\text{train}}}$; mini-batch size s, deep neural network $f(w, x, y)$; learning rates η_k and γ_k, loss function $l(\cdot)$, distance function $D(\cdot, \cdot)$.

Output: Classifier weights $w \in \Theta$

Randomly initialize weights of the DNN classifier: $w_0 \in \Theta$

// vector of worst-case perturbations, one for each time-series

Initialize $a_0 = 0$ and $\omega_0 = 0$

// Multiple iterations of SCAGDA

for $k = 0, \cdots, K-1$ **do**

  Randomly sample a mini-batch of data samples indexed by $\mathcal{I}_k$ s.t. $|\mathcal{I}_k| = s$

  Compute the stochastic gradient w.r.t. w_k

  $\mathcal{G}_{w,k} = \frac{1}{s} \sum_{i \in \mathcal{I}_k} \nabla_w l(f(w_k, x_i + a_i^k, y_i)$

  Perform stochastic gradient descent on w_k

  $w_{k+1} = w_k - \eta_k \mathcal{G}_{w,k}$

  Randomly sample a mini-batch of alignments indexed by $\widehat{\mathcal{A}}_i^k$ for each data index $i \in \mathcal{I}_k$

  Moving average for $i \in \mathcal{I}_k$

  $\omega_i^{k+1} = (1-\beta)\omega_i^k + \beta \sum_{\pi \in \widehat{\mathcal{A}}_i^k} \exp(-d_\pi(x_i, x_i + a_i^k)/\nu)$

  $\omega_i^{k+1} = \omega_i^k$ for $i \notin \mathcal{I}_k$.

  Compute $\mathcal{G}_{a,k,i} = \nabla_a \ell(f(w_{k+1}, x_i + a_i^k, y_i) - \sum_{\pi \in \widehat{\mathcal{A}}_i^k} \exp(-d_\pi(x_i, x_i + a_i^k)/\nu) \cdot \nabla_a d_\pi(x_i, x_i + a_i^k) \cdot \frac{\lambda}{\omega_i^{k+1} \nu}$ for $i \in \mathcal{I}_k$

  Perform stochastic gradient ascent over perturbations

  $a_i^{k+1} = a_i^k + \gamma_k \mathcal{G}_{a,k,i}$

end for

return weights of the learned DNN classifier, w_K

guarantee of our SCAGDA algorithm for nonconvex-nonconcave compositional min-max optimization problems?

Theorem 5 proves that SCAGDA shown in Algorithm 6 converges to an ϵ-primal gap in $O(\frac{1}{\epsilon^2})$ iterations. Theorem 6 demonstrates the efficacy of the moving average strategy to approximate GAK based distance: the approximation error $\|\omega_i^K - h_i(a_i^{K-1}))\|^2$ is also bounded by ϵ in expectation when the ϵ-primal gap is achieved.

6.3.1 Main Results

The following commonly used assumptions are used in our analysis.

Assumption 1. *Suppose* $\mu, L, C_g, C_h, L_g, L_h \geq 0$.

(i) $\phi(w, a)$ *satisfies two side* μ*-PL (Polyak-Lojasiewicz) condition:*

$$\|\nabla_w \phi(w, a)\|^2 \geq 2\mu(\phi(w, a) - \min_{w'} f(w, a)),$$

$$\|\nabla_a \phi(w, a)\|^2 \geq 2\mu(\max_{a'} f(w, a) - \phi(w, a)).$$

(ii) $\phi(w, a)$ *is* L*-smooth in* w *for fixed* a.

(iii) $\phi(w, a)$ *is* L*-smooth in* a_i *for fixed* w.

(iv) g *(resp.* h*) is* C_g *(resp.* C_h*)-Lipschitz continuous.*

(v) g *(resp.* h*) is* L_g *(resp.* L_h*)-smooth.*

(vi) $\exists\ \sigma > 0$ *s.t.* $\mathbb{E}[\|\nabla_w \phi_i(w, a_i) - \nabla_w \phi(w, a)\|^2] \leq \sigma^2$,
$\mathbb{E}[\|\nabla_a f_i(w, a_i) - \nabla_a f_i(w, a_i)\|^2] \leq \sigma^2$, $\mathbb{E}[\|h_{i,j}(a_i) - h(a)\|^2] \leq \sigma^2$, *and* $\mathbb{E}[\|\nabla h_{i,j}(a_i) - \nabla h(a)\|^2] \leq \sigma^2$

We present our main results for SCAGDA below.

Theorem 5. *Suppose Assumption 1 holds. In Algorithm 6, set* $\eta_k = \eta = O(1/\epsilon^2)$, $\gamma_k = \gamma = O(1/L^2)$ *and* $\beta = \sqrt{18\mu\eta}$. *After running Algorithm 6 for* K *iterations where* $K = \tilde{O}(1/\epsilon^2)$ *(*$\tilde{O}$ *hides logarithmic factor), we have*

$$\mathbb{E}[P(w_K) - P^*] + \frac{1}{8}\mathbb{E}[P(w_K) - \phi(w_K, a_K)]$$
$$+ \Big(\frac{4C_h^4 L_g^4 \eta_K}{\mu^5}\Big)^{1/2} \mathbb{E}[\|\omega_K - h(a_{K-1})\|^2] \leq \epsilon$$

Remark 1. The above theorem gives us two critical observations of the behavior of SCAGDA. **(1)** After running K iterations of SCAGDA, the primal gap $P(w_{K+1}) - P^*$ converges to ϵ in expectation, since all terms in the left hand side of the inequality are non-negative. This result shows that SCAGDA is able to effectively solve the compositional min-max optimization problem shown in Equation (6.7). **(2)** The required iteration complexity of SCAGDA is $O(1/\epsilon^2)$. To put this result in perspective, we compare it with related theoretical results. The rate for nonconvex-nonconcave min-max problem without compositional structure is shown to be $O(1/\epsilon)$ (Junchi Y., 2020). However, this improvement requires unbiased estimation (or exact value) of the gradient and computing the exact $g(h_i(a_i))$ at each iteration. Our iteration complexity is in the same order of that for (T. Chen, Sun, and Yin, 2020), whose convergence result is $O(1/\epsilon^2)$ for nonconvex compositional *minimization* problems instead of *min-max* ones. The difference is that their convergence metric is the average squared norm of gradients, while ours is for the primal gap. Importantly, *this is the first result on convergence rate for stochastic compositional min-max problems.*

Theorem 6. *After $K = \tilde{O}(1/\epsilon^2)$ iterations of Algorithm 6, we have:* $\mathbb{E}[\|\omega_i^K - h_i(a_i^{K-1})\|^2] \leq O(\epsilon)$.

Remark 2. The above result shows that as SCAGDA algorithm is executed, the approximation error of $\|\omega_i^K - h_i(a_i^{K-1})\|^2$ converges to ϵ in the expectation as it is achieving the ϵ-primal gap. For the condition numbers, we always have $L \geq \mu$. In practice, we usually set the accuracy level ϵ to a very small value, so the condition $\epsilon \leq O(L^3/\mu^2)$ will generally hold. This result provides strong theoretical support that if we apply SCAGDA to optimize our RO-TS problem in (6.5), it is able to approximate k_{GAK} on-the-fly, where we only need a constant number of alignments, rather than *all* possible alignments for computing k_{GAK} in each iteration of SCAGDA. When we have ϵ-primal gap, we also achieve ϵ-accurate estimation of k_{GAK} at the same time.

6.4 Experiments and Results

We present experimental evaluation of RO-TS on real-world time-series benchmarks and compare with prior methods.

6.4.1 Experimental Setup

Datasets. We employed diverse multi-variate time-series benchmark datasets from the UCR repository (Bagnall et al., 2020). Table 6.1 describes the details of representative datasets for which we show the results (due to space limits) noting that our overall findings were similar on other datasets from the UCR repo. We employ the standard training/validation/testing splits for these datasets.

Name	**Classes**	**Input Size** ($C \times T$)
ECG200	2	1×97
BME	3	1×129
ECG5000	5	1×141
MoteStrain	2	1×85
SyntheticControl	6	1×61
RacketSports	4	6×30
ArticularyWR	25	9×144
ERing	6	4×65
FingerMovements	2	28×50

Table 6.1 Description of different datasets.

Algorithmic setup and baselines. We employ a 1D-CNN architecture (Bai, J Z. Kolter, and Koltun, 2018) as the deep model for our evaluation. We ran RO-TS algorithm for a maximum of 500 iterations to train robust models. To estimate GAK distance within RO-TS, we employed 15 percent of the total alignments noting that larger sample sizes

didn't improve the optimization accuracy and increased the training time. We also employ adversarial training to create models using baseline attacks that are not specific to image domain for comparison: Fast Gradient Sign method (FGS) (Kurakin, I. Goodfellow, and Bengio, 2016) that was used by H Ismail Fawaz et al. (2019) and Projected Gradient Descent (PGD)(Aleksander Madry et al., 2017). We also compare RO-TS against stability training (STN) (Stephan Zheng et al., 2016).

Evaluation metrics. We evaluate the robustness of created models using different attack strategies on the testing data. The prediction accuracy of each model (via ground-truth labels of time-series) is used as the metric. To ensure robustness, DNN models should be least sensitive to different types of perturbations over original time-series signals. We measure the accuracy of each DNN model against: *1) Adversarial noise* is introduced by FGS and PGD baseline attacks; and *2) Gaussian noise* $\sim \mathcal{N}(0, \Sigma)$ that may naturally occur to perturb time-series. The covariance matrix Σ diagonal elements (i.e., variances) are all equal to σ. DNNs are considered robust if they are successful in maintaining their accuracy performance against such noises.

6.4.2 Results and Discussion

RO-TS vs. adversarial training. One of our key hypothesis is that Euclidean distance-based perturbations do not capture the appropriate notion of invariance for time-series domain to improve the robustness of the learned model. We show that using baseline attacks to create augmented data for adversarial training does not create robust models. From Figure 6.1, we can observe that models from our RO-TS algorithm achieve significantly higher accuracy than the baselines. For example, on MoteStrain dataset, RO-TS has a steady performance against both types of noises, unlike the baselines. On the other datasets, we can clearly observe that in most cases, RO-TS outperforms the baselines. We conclude that adversarial training using prior methods and attack strategies is not as effective as our RO-TS

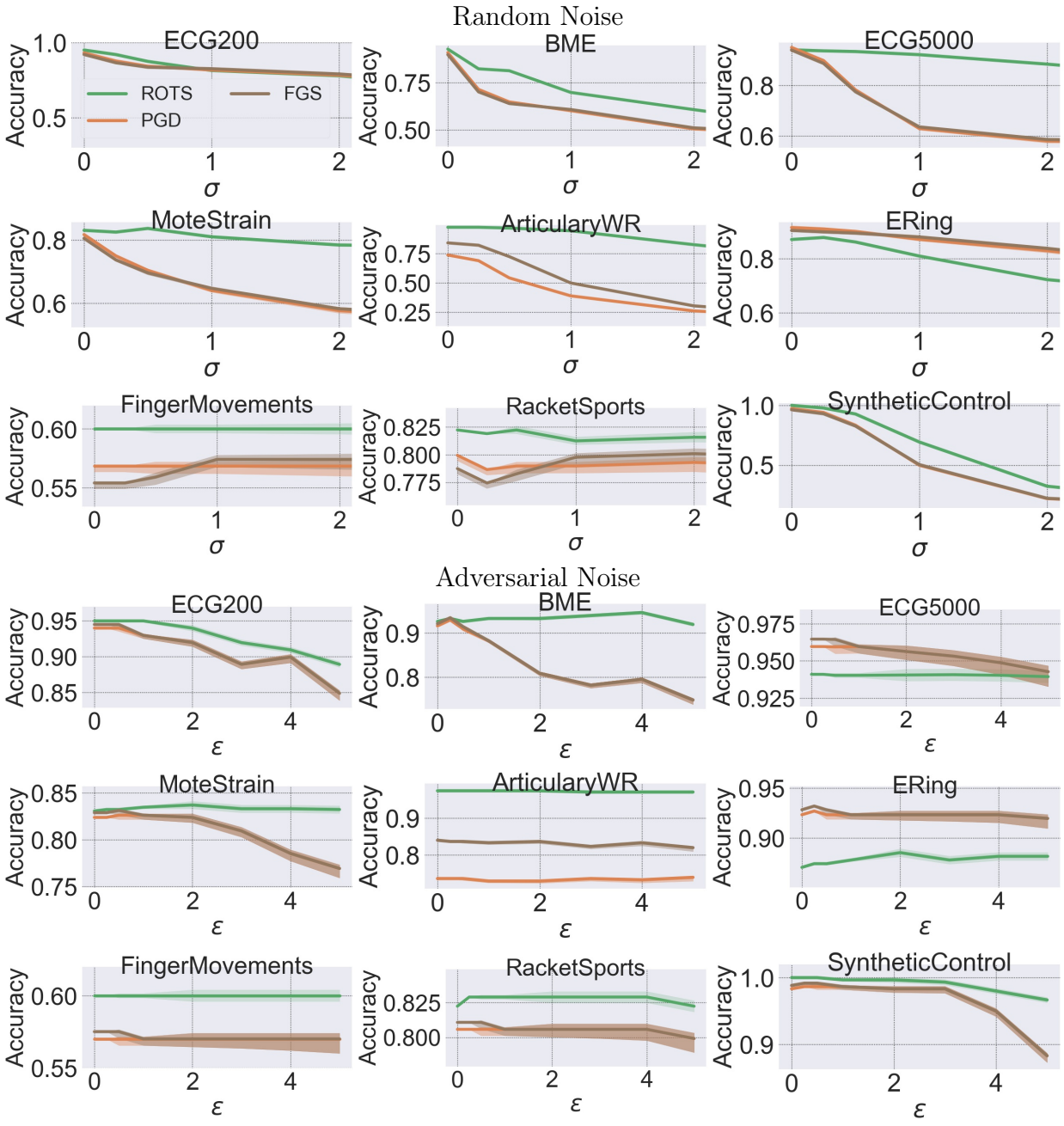


Figure 6.1 Comparison of RO-TS algorithm vs. adversarial training algorithm using baseline FGS and PGD attacks.

method, where we perform explicit primal-dual optimization to create robust models.

RO-TS vs. RO-TS with L2 distance. We want to demonstrate that choosing the right distance metric to compute similarity between time-series signals is critical to create robust models. Therefore, we compare models created by RO-TS by using two different distance

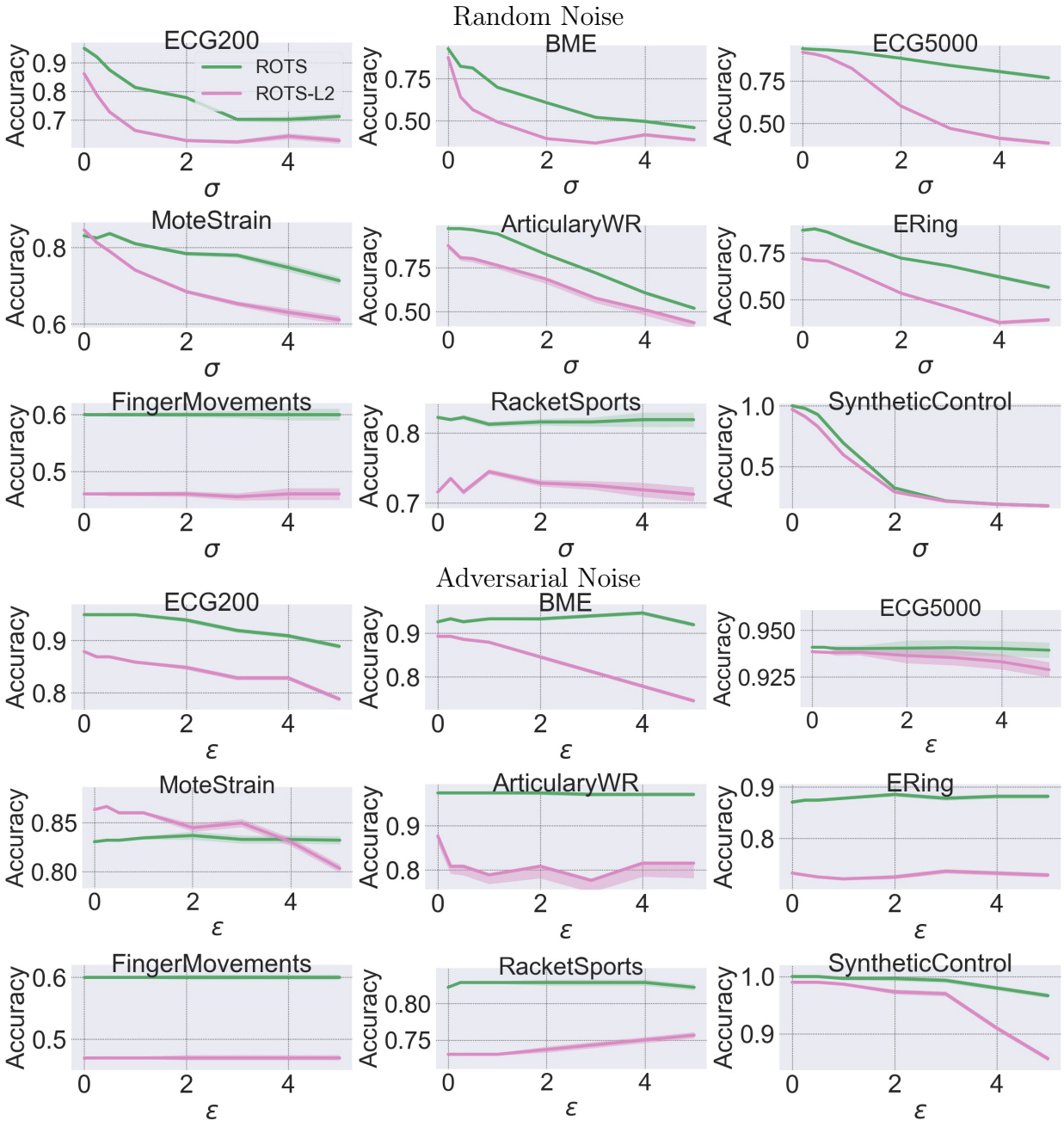


Figure 6.2 Comparison of RO-TS algorithm using GAK distance (k_{GAK}) vs. RO-TS using Euclidean distance (l_2).

metrics: 1) The standard Euclidean distance $\|\cdot\|_2$ used in image domains and prior work; and 2) Using the GAK distance D_{GAK}. From Figure 6.2, we can clearly see that the GAK distance is able to explore the time-seris input space better to improve robustness. The Euclidean distance either performs significantly worse than GAK (e.g., on ECG5000, ERing,

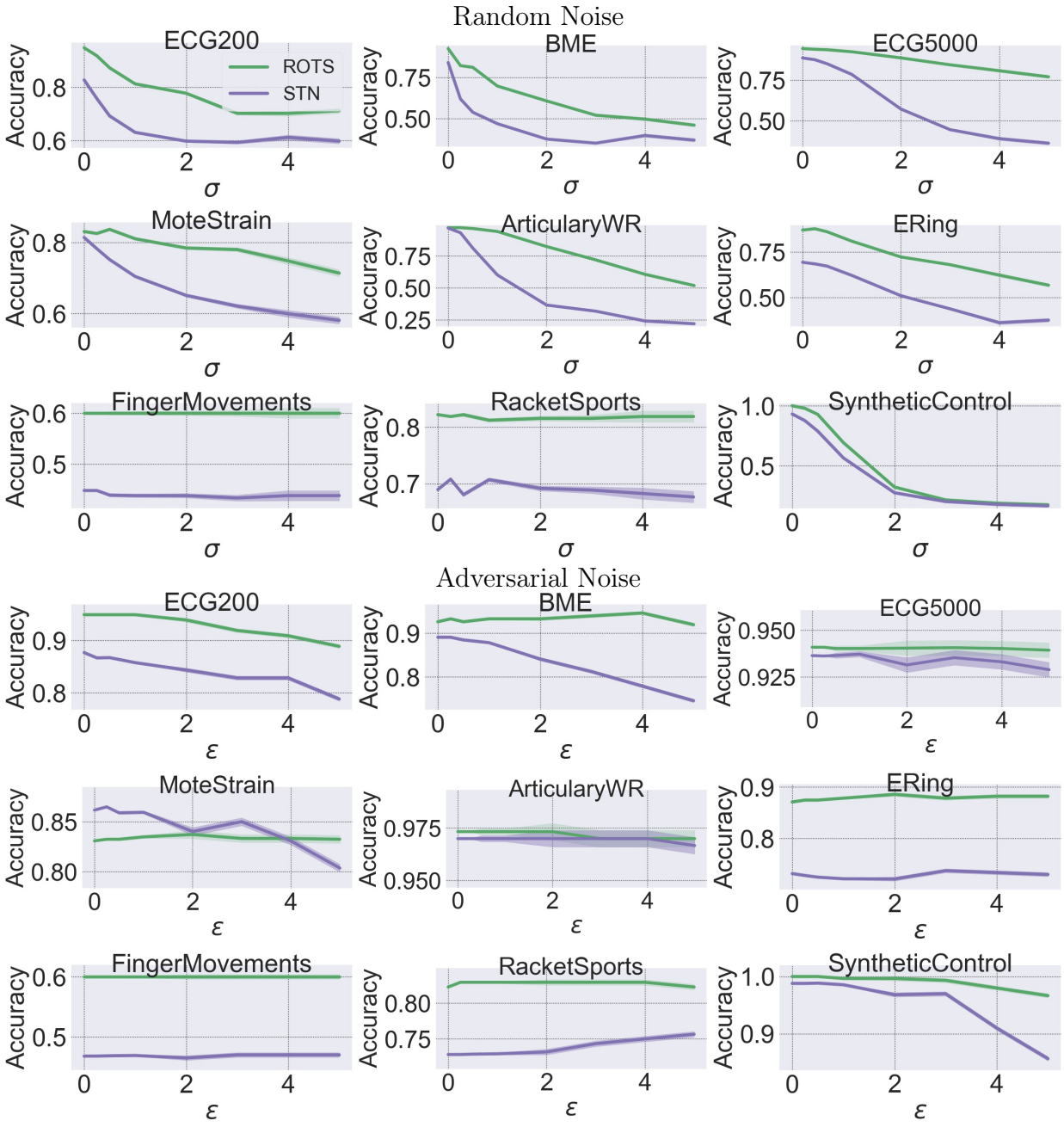


Figure 6.3 Comparison of RO-TS vs. stability training (STN).

and RacketSports datasets) or performs comparably to GAK (e.g., on SyntheticControl or ArticularyWR datasets). This experiment concludes that GAK is a suitable distance metric for time-series domain.

RO-TS vs. stability training. Unlike adversarial training, stability training (STN)

employs the below loss function (Stephan Zheng et al., 2016) to introduce stability to the deep model.

$$Loss_{STN} = L_0(x, \theta) + 0.01 \times L_{stability}(x, x', \theta) \tag{6.8}$$

where x is the original input, x' is a perturbed version of x using additive Gaussian noise $\sim \mathcal{N}(0, 0.04^2)$, L_0 is the cross-entropy loss, and $L_{stability}$ relies on KL-divergence. We experimentally demonstrate that RO-TS formulation is more suitable than STN for creating robust DNNs for time-serirs domain. Figure 6.3 shows a comparison between DNNs trained using STN and RO-TS. We observe that for most datasets, RO-TS creates significantly more robust DNNs when compared to STN for both types of perturbations. RO-TS algorithm is specifically designed for time-series domain by making appropriate design choices, whereas STN is designed for image domain. Hence, RO-TS allows us to create more robust DNNs for time-series domain.

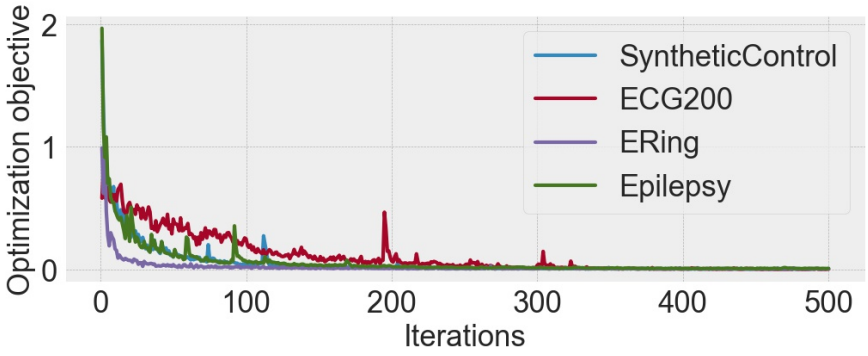


Figure 6.4 Empirical convergence of RO-TS algorithm.

Empirical convergence. We demonstrate the efficiency of RO-TS algorithm by observing the empirical rate of convergence. Figure 6.4 shows the optimization objective over iterations on some representative datasets noting that we observe similar patterns on other datasets. We can observe that RO-TS converges roughly before 150 iterations for most datasets.

Figure 6.5 shows the accuracy gap results and the computational runtime when comparing RO-TS with sampled alignments and original GAK (i.e., all alignment paths). The results clearly match with our theoretical analysis that accuracy gap decreases over training

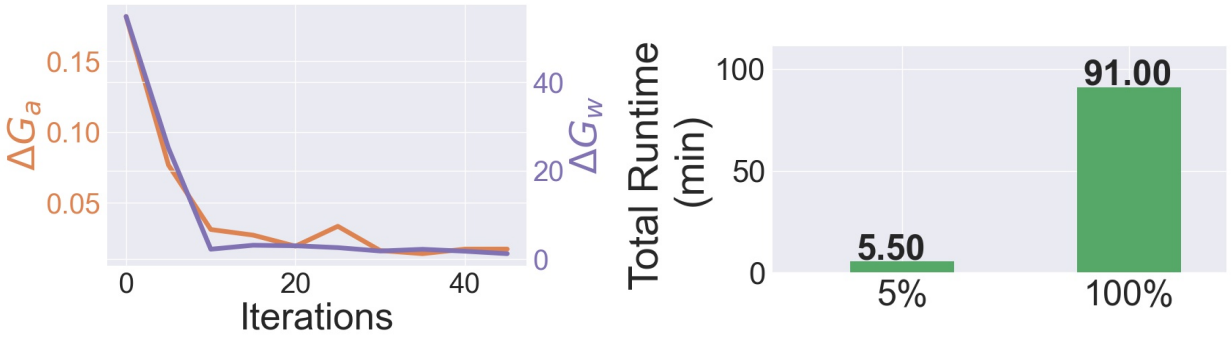


Figure 6.5 The accuracy gap in the gradients over weights ΔG_W and over perturbations ΔG_a using 5% of alignments and GAK using all alignments for RO-TS training on ERing (Left) and the comparison of the computational runtime between both settings of RO-TS (Right).

iterations leading to convergence. We conclude from these results that RO-TS converges quickly in practice and supports our theoretical analysis.

6.5 Summary

In this chapter, we introduced the RO-TS algorithm to train robust deep neural networks (DNNs) for time-series domain. The training problem was formulated as a min-max optimization problem to reason about the worst-case risk in a small neighborhood defined by the global alignment kernel (GAK) based distance. Our proposed stochastic compositional alternating gradient descent and ascent (SCAGDA) algorithm carefully leverages the structure of the optimization problem to solve it efficiently. Our theoretical and empirical analysis showed that RO-TS and SCAGDA are effective in creating more robust DNNs over prior methods and GAK based distance is better suited for time-series over the Euclidean distance.

CHAPTER SEVEN

OUT-OF-DISTRIBUTION DETECTION IN TIME-SERIES DOMAIN: A SEASONAL RATIO SCORING APPROACH

T. Belkhouja, Y. Yan, and J. Doppa. "Out-of-Distribution Detection in Time-Series Domain: A Novel Seasonal Ratio Scoring Approach. *ACM Transactions on Intelligent Systems and Technology (TIST), 15(1): 9:1-9:24*, 2023.

Originally published in the *ACM Transactions on Intelligent Systems and Technology.*

Attributions:

T. Belkhouja has contributed to this work by formulating the problem setting, investigating the state of the art, formulating the theoretical contribution and the algorithmic solution, implementing the algorithm and running the required empirical analysis to highlight the performance improvement of the proposed solution compared to the state of the art.

Y. Yan has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the formulation of the theory of the proposed solution, and in writing the scientific manuscript.

J. Doppa has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

OUT-OF-DISTRIBUTION DETECTION IN TIME-SERIES DOMAIN: A SEASONAL RATIO SCORING APPROACH

In this chapter, we proposes a novel OOD detection algorithm for the time-series domain referred to as Seasonal Ratio Scoring (SRS). *To the best of our knowledge, this is the first work on OOD detection over time-series data.* SRS employs the Seasonal and Trend decomposition using Loess (STL) (Cleveland et al., 1990) on time-series signals from the ID data to create a class-wise semantic pattern and a remainder component for each signal. For example, in a human activity recognition system, SRS would extract a pattern "running" that describes semantically all the recorded "running" windows. If the person trips and falls, SRS would detect that this event does not belong to the pre-defined activity classes and flag it as OOD. For this purpose, we train two separate DGMs to estimate the class-wise conditional likelihood of a given time-series signal and its STL-based remainder component. The Seasonal Ratio (SR) score for each time-series signal from ID is computed from these two estimates. A threshold interval is estimated from the statistics of all these scores over ID data. Given a new time-series input and a classifier at the testing time, the SRS approach computes the SR score for the predicted output and flags the time-series signal as OOD example if the score lies outside the threshold interval. Figure 7.2 illustrates the SRS algorithm. The effectiveness of SRS critically depends on the extraction of accurate class-wise semantic components. Since time-series data is prone to warping and time-shifts, we also propose a new alignment approach based on Dynamic Time Warping (DTW) (Müller, 2007) to improve the output accuracy of STL decomposition.

7.1 Background and Problem Setup

Suppose $\mathcal{D}_{in}$ is an in-distribution (ID) time-series dataset with d examples $\{(x_i, y_i)\}$ sampled from the distribution P^* defined on the joint space of input-output pairs $(\mathcal{X}, \mathcal{Y})$. Each $x_i \in \mathbb{R}^{n \times T}$ from $\mathcal{X}$ is a multi-variate time-series input, where n is the number of channels

and T is the window-size of the signal. $y_i \in \mathcal{Y} = \{1, \cdots, C\}$ represents the class label for time-series signal x_i. We consider a time-series classifier $F : \mathbb{R}^{n \times T} \rightarrow \{1, \cdots, C\}$ learned using $\mathcal{D}_{in}$. For example, in a health monitoring application using physiological sensors for patients diagnosed with cardiac arrhythmia, we use the measurements from wearable devices to predict the likelihood of cardiac failure.

OOD samples (x, y) are typically generated from a distribution other than P^*. Specifically, we consider a sample (x, y) to be OOD if the class label y is different from the set of in-distribution class labels, i.e., $y \notin \mathcal{Y}$. The classifier $F_\theta(x)$ learned using $\mathcal{D}_{in}$ will assign one of the C class labels from $\mathcal{Y}$ when encountering an OOD sample (x, y). Our goal is to detect such OOD examples for safe and reliable real-world deployment of time-series classifiers.

Challenges of time-series data. The unique characteristics of time-series data (e.g., temporal relation across time-steps, fast oscillations, continuous distribution of variables) pose unique challenges not seen in the image domain. Real-world time-series datasets are typically small (relative to image datasets) and exhibit high class-imbalance (Dau, Bagnall, et al., 2019). Therefore, estimating a good approximation of in-distribution P^* is hard, which results in the failure of prior OOD methods. Indeed, our experiments demonstrate that prior OOD methods are not suited for the time-series domain. As a prototypical example, Figure 7.7 shows the limitation of Likelihood Regret score (Z. Xiao, Q. Yan, and Amit, 2020) to identify OOD examples: ID and OOD scores of real-world time-series examples overlap.

Conditional VAE. Variational Auto-Encoders (VAEs) are a class of likelihood-based generative models with many real-world applications (Doersch, 2016). They rely on the encoding of a raw input x as a latent Gaussian variable z to estimate the likelihood of x. The latent variable z is used to compute the likelihood of the training data: $p_\theta(x) = \int p_\theta(x|z)p(z)dz$. Since the direct computation of this likelihood is impractical, the principle of evidence lower bound (ELBO) (Z. Xiao, Q. Yan, and Amit, 2020) is employed. In this work, we consider the ID data $\mathcal{D}_{in}$ as d input-output samples of the form (x_i, y_i). We want to estimate the ID using both x_i and y_i. Therefore, we propose to use conditional VAE (CVAE) for this

purpose. CVAEs are a class of likelihood-based generative models (Doersch, 2016). They rely on the encoding of raw input (x, y) as a latent Gaussian variable z to estimate the conditional likelihood of x over the class label y. CVAE is similar to VAE with the key difference being the use of conditional probability over both x_i and y_i. The ELBO objective of CVAE is:

$$\mathcal{L}_{\text{ELBO}} \triangleq \mathbb{E}_{\phi}\big[\log p_{\theta}(x|z,y)\big] - D_{\text{KL}}\big[q_{\phi}(z|x,y)||p(z|y)\big]$$

where $q_{\phi}(z|x,y)$ is the variational approximation of the true posterior distribution $p_{\theta}(x|z,y)$. As CVAE only computes the lower bound of the log-likelihood of a given input, the exact log-likelihood is estimated using Monte-Carlo sampling as shown below:

$$\mathcal{L}_{M} = \mathbb{E}_{z^m \sim q_{\phi}(z|x,y)}\left[\log \frac{1}{M}\sum_{m=1}^{M} \frac{p_{\theta}(x|z^m,y)p(z^m)}{q_{\phi}(z^m|x,y)}\right] \tag{7.1}$$

The intuitive expectation from a DGM learned using training data is to assign a high likelihood to ID samples and a low likelihood to OOD samples. However, recent research showed that DGMs tend to assign highly unreliable likelihood to OOD samples regardless of the different semantics of both ID and OOD data (Z. Xiao, Q. Yan, and Amit, 2020). Indeed, our experimental results shown in Table 7.3 demonstrate that this observation is also true for the time-series domain.

STL decomposition. STL (Jiang, Changbiao Huang, and B. Liu, 2011) is a statistical method for decomposing a given time-series signal x into three different components: 1) The seasonality x_s is a fixed regular pattern that recurs in the data; 2) The trend x_t is the increment or the decrement of the seasonality over time; and 3) The residual x_r represents random additive noise. STL employs Loess (LOcal regrESSion) smoothing in an iterative process to estimate the seasonality component x_s (McKinney, Perktold, and Seabold, 2011). The remainder is the additive residual from the input x after summing both x_s and x_t. For the proposed SRS algorithm, we assume that there is a fixed semantic pattern S_y for every class label $y \in \mathcal{Y}$, and this pattern recurs in all examples (x_i, y_i) from $\mathcal{D}_{in}$ with the same class label, i.e., $y_i{=}y$. We will elaborate more on this assumption and a reformulation of the

problem that can be used when the assumption is violated in the next section. Hence, every time-series example has the following two elements: $x_i = S_{y_i} + r_i$, where S_{y_i} is the pattern for the class label $y{=}y_i$ and r_i is the remainder noise w.r.t S_{y_i}. For time-series classification tasks, ID samples are assumed to be stationary. Therefore, we propose to average the trend x_t component observed during training and include it in the semantic pattern S_y. Figure 7.1 illustrates the above-mentioned decomposition for two different classes from the ERing dataset.

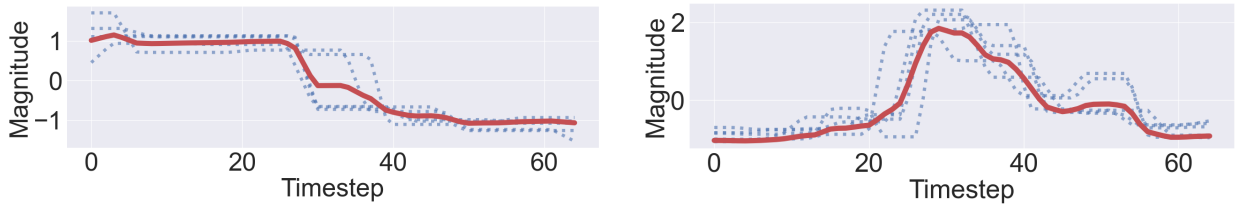


Figure 7.1 Illustration of STL method for two different classes from the ERing dataset. Dotted signals are natural time-series signals x and the red signal is the semantic pattern S_y.

7.2 Seasonal Ratio Scoring Approach for OOD Detection

Overview of SRS algorithm. The training stage proceeds as follows. We employ STL decomposition to get the semantic component S_y for each class label $y \in \mathcal{Y}{=}\{1, \cdots, C\}$ from the given in-distribution (ID) data $\mathcal{D}_{in}$. Details on the STL decomposition steps within the SRS framework are provided in Section 7.2.2. To improve the accuracy of STL decomposition, we apply a time-series alignment method based on dynamic time warping to address scaling, warping, and time-shifts as detailed in Section 7.2.3. We train two CVAE models $\mathcal{M}_x$ and $\mathcal{M}_r$ to estimate the class-wise conditional likelihood of each time-series signal x_i and its remainder component r_i w.r.t the semantic component S_{y_i}. The seasonal ratio score for each ID example (x_i, y_i) from $\mathcal{D}_{in}$ is computed as the ratio of the class-wise conditional likelihood estimates for x_i and its remainder r_i: $SR_i(x_i, y_i) \triangleq \frac{p(x_i|y=y_i)}{p(r_i|y=y_i)}$. We compute the SR scores for all in-distribution examples from $\mathcal{D}_{in}$ to estimate the threshold interval $[\tau_l, \tau_u]$

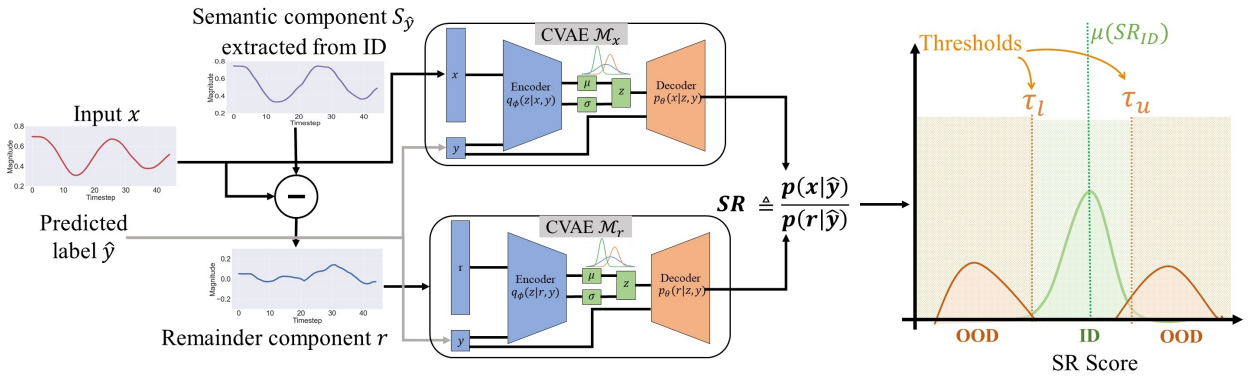


Figure 7.2 Overview of the seasonal ratio (SR) scoring algorithm. The semantic component $S_{\hat{y}}$ for the predicted output $\hat{y}$ is obtained from the training stage via Seasonal and Trend decomposition using Loess (STL). The semantic component $S_{\hat{y}}$ is subtracted from the time-series x to obtain the remainder r. The trained CVAE models $\mathcal{M}_x$ and $\mathcal{M}_r$ are used to compute the SR score. If the SR score is within the threshold interval $[\tau_l, \tau_u]$ identified during training, then x is classified as ID. Otherwise, it is flagged as OOD.

for OOD detection. During the inference stage, given a time-series signal x and a trained classifier $F(x)$, we compute the SR score of x with the predicted output $\hat{y}{=}F(x) \in \mathcal{Y}$ and identify it as an OOD example if the SR score lies outside the threshold interval $[\tau_l, \tau_u]$. Figure 7.2 provides a high-level illustration of the SRS algorithm.

Below we first provide an intuitive explanation to motivate the SR score. Next, we describe the complete details of the SRS algorithm including both training and inference stages. Finally, we motivate and describe a time-series alignment approach based on dynamic time warping to improve the effectiveness of SRS.

7.2.1 Intuition for Seasonal Ratio Score

We explain the intuition behind the proposed SRS algorithm using STL decomposition of time-series signals and CVAE models for likelihood estimation. Current research shows that DGMs alone can fail to identify OOD samples (Z. Xiao, Q. Yan, and Amit, 2020). They not only assign high likelihood to OOD samples, but they also exhibit good reconstruction quality. In fact, we show in Table 7.3 that CVAEs trained on a given ID data generally

exhibit a low reconstruction error on most of the OOD samples. Furthermore, we show in Table 7.7 that using a trained CVAE likelihood output for OOD detection fails to perform well. These results motivate the new for a new OOD scoring method for the time-series domain.

Class-wise seasonality via STL decomposition. The proposed SRS algorithm relies on the following assumption to analyze the time-series space for OOD detection.

Assumption 2. *Each time-series example (x_i, y_i) from the in-distribution data $\mathcal{D}_{in}$ consists of two components. 1) A class-wise semantic pattern S_y for each class label $y \in \mathcal{Y}$ representing the meaningful semantics of the class label y. 2) A remainder noise r_i representing an additive perturbation to the semantic portion. Hence, $\forall (x_i, y_i) \in \mathcal{D}_{in} : \; x_i = S_{y_i} + r_i$*

We propose to employ STL decompostion to estimate semantic pattern S_y (as illustrated in Figure 7.1) and deduce the remainder noise that can be due to several factors including errors in sensor measurements and noise in communication channels. These two components are analogous to the foreground and the background of an image, where the foreground is the interesting segment of the input that describes it, and the background may not be necessarily related to the foreground. In spite of this analogy, prior methods for the image domain are not suitable for time-series as explained in the related work. In this decomposition, we cannot assume that S_y and r are independent for a given time-series example (x, y), as S_y is class-dependant and r is the remainder of the input x using S_y. Hence, we present their conditional likelihoods in the following observation.

Observation 3. *Let $x \in \mathbb{R}^{n \times T}$ is a time-series signal and $y_i \in \mathcal{Y}$=$\{1, \cdots, C\}$ be the corresponding class label. As $x = S_{y_i} + r$, we have:*

$$p(x|y_i) = p(r|y_i)p(S_{y_i}|y_i) \tag{7.2}$$

Proof of Observation 1 As $X = S_{y_i} + r$, it is intuitive to think that $p(X|y = y_i) = p(S_{y_i}) \times p(r)$. However, we cannot assume that S_{y_i} and r are independent, as S_{y_i} is class-dependant and r is the remainder of the input X given S_{y_i}.

Therefore, we make use of the conditional probabilities of the components. The likelihood $p(X)$ can be decomposed as follows:

$$\begin{aligned} p(X|y = y_i) &= p(S_{y_i}, r|y = y_i) \\ &= \frac{p(S_{y_i}, r, y = y_i)}{p(y = y_i)} \\ &= \frac{p(r, y = y_i)p(S_{y_i}|r, y = y_i)}{p(y = y_i)} \end{aligned}$$

For the conditional probability $p(S_{y_i}|r, y = y_i)$, as only the pattern S_{y_i} depends on the class label, and that we have defined r as a non-meaningful noise to the input, we can assume that S_{y_i} **and** r **are conditionally independent given the class** y_i. Therefore, we have the following.

$$\begin{aligned} p(X|y = y_i) &= \frac{p(r, y = y_i)p(S_{y_i}|r, y = y_i)}{p(y = y_i)} \\ &= \frac{p(r, y = y_i)p(S_{y_i}|y = y_i)}{p(y = y_i)} \\ &= \frac{p(r|y = y_i)p(y = y_i)p(S_{y_i}|y = y_i)}{p(y = y_i)} \\ &= p(r|y = y_i)p(S_{y_i}|y = y_i) \end{aligned}$$

Discussion on Observation 1. S_y is a fixed class-wise semantic pattern that characterizes a class $y \in \mathcal{Y}$. By definition, S_y is a deterministic pattern extracted using STL decomposition during training and is not a random variable. At the inference time, we do not estimate S_y of each test input x, but we use S_y computed during the training stage to estimate the remainder component r. Hence, $P(S_y|y)$ is defined as a deterministic variable and not as a density that SRS is aiming to estimate.

OOD detection using CVAEs. Observation 3 shows the relationship between the conditional likelihood of the input x and its remainder r. We propose to employ CVAEs to

estimate both likelihoods since they are conditional likelihoods. Recall that OOD examples come from an unknown distribution which is different from the in-distribution P^* and do not belong to any pre-defined class label from $\mathcal{Y}$. Therefore, we propose to use the following observation for OOD detection in the time-series domain.

Observation 4. *Let $x \in \mathbb{R}^{n \times T}$ is a time-series signal and $y \in \mathcal{Y}$=$\{1, \cdots, C\}$ be the corresponding class label. As $x = S_y + r$, x is an OOD example if $p(x|y) \neq p(r|y)$ and an in-distribution example if $p(x|y) = p(r|y)$.*

Observation 4 shows how we can exploit the relationship between the estimated conditional likelihood of the time-series signal x and its remainder r to predict whether x is an OOD example or not. This observation relies on the assumption that $p(S_y|y) = 1$ for in-distribution data. For ID data, the semantic pattern S_y is a class-dependant signal that defines the class label y. Since the semantic component is guaranteed to be S_y for any time-series example with class label y, we have $p(S_y|y)$=1. On the other hand, OOD examples do not belong to any class label from $\mathcal{Y}$, i.e., $p(S_y|y) \neq 1$ for any $y \in \mathcal{Y}$. To estimate $p(x|y \in \mathcal{Y})$ and $p(r|y \in \mathcal{Y})$ in observation 4, we train two separate CVAE models using the in-distribution data $\mathcal{D}_{in}$. While estimating two separate distributions can cause instability, we note that

1. During hyper-parameter tuning and the definition of the ID score range $[\tau_l, \tau_u]$, any outlier that may cause estimation instability will be omitted.

2. In case of drastic estimation instability, both CVAEs can be tuned during training time to overcome the problem.

3. If this instability is seen during inference time, then the SRS algorithm automatically indicates that the test example is an OOD example.

Discussion on Assumption 1. This work acknowledges that this assumption may fail to hold in some real-world scenarios. However, surprisingly, our experimental results shown in

Table 7.6 strongly corroborate this key assumption: the distance between each time-series signal x_i and its semantic pattern S_{y_i} is very small. The strong OOD performance of SRS algorithm in our diverse experiments demonstrates the effectiveness of a simple approach based on this assumption.

Suppose the assumption does not hold and some class label y can possess $K > 1$ different semantics $\{S_y^k\}_{k \leq K}$. If we take a human activity recognition example, it is safe to think of a certain activity (e.g., running or walking) will have $K > 1$ different patterns (e.g., athletic runners vs. young runners). Therefore, the decomposition in Assumption 2 for a given time-series example (x_i, y_i) will result into a semantic pattern describing the patterns of the different sub-categories (e.g., a pattern that describes both athletic runners and young runners). By using Lowess smoothing, the STL season extracted over a multiple-pattern class is a pattern S_y that is a linear combination of $\{S_y^k\}_{k \leq K}$ (for our example, it describes the combination of both athletic runs and young runs). While for an in-distribution example, $p(S_y|y)$=1 of Observation 2 will not hold, $p(S_y|y)$ is likely to be well-defined from $p(S_y^k|y)$ as $\{S_y^k\}_{k \leq K}$ are fixed and natural for the class label y. Hence, we can still rely on the CVAEs to estimate this distribution and to perform successful OOD detection. Alternatively, we can use a simple reformulation of the problem by clustering time-series signals of a class label y (for which the assumption is not satisfied) to identify sub-classes and apply the SRS algorithm on transformed data. Since we found the assumption to be true in all our experimental scenarios (see Table 7.6), we didn't find the need to apply this reformulation.

7.2.2 OOD Detection Approach

One key advantage of SRS method is that it can be directly executed at inference stage and does not require additional training similar to prior VAE-based methods such as Likelihood Regret scoring.

Training stage. Our overall training procedure for time-series OOD detection is as follows:

1. Train a CVAE $\mathcal{M}_x$ using in-distribution data $\mathcal{D}_{in}$ to estimate the conditional likelihood $p(x|y \in \mathcal{Y})$ of time-series signal x.

2. Execute STL decomposition as follows:

   (a) From the training data $\{(x_i, y_i)\}$, we create a group $\mathcal{D}_y = \{x_i | y_i = y\}$.

   (b) We concatenate all the examples $x_i \in \mathcal{D}_y$ in a single stream of data according to the T dimension. If $\mathcal{D}_y$ has k examples, the output is a single stream $X_{stream} \in \mathbb{R}^{n \times (k \cdot T)}$

   (c) We apply STL decomposition on the stream X_{stream} by defining the pattern dimensions $S_y \in \mathbb{R}^{n \times T}$.

   (d) We store the semantic component S_y to be used later in estimating the remainder component for any given training example (x, y): $r = x - S_y$.

3. Create the remainder for each training example $(x_i, y_i) \in \mathcal{D}_{in}$ using the patterns S_y for each class label y : $r_i = x_i - S_{y_i}$. We train another CVAE $\mathcal{M}_r$ using all these remainders to estimate the conditional likelihood $p(r|y \in \mathcal{Y})$.

4. Compute seasonal ratio score for each $(x_i, y_i) \in D_{in}$ using the trained CVAEs $\mathcal{M}_x$ and $\mathcal{M}_r$.

$$SR_i(x_i, y_i) \triangleq \frac{p(x_i|y = y_i)}{p(r_i|y = y_i)} \tag{7.3}$$

5. Compute the mean μ_{SR} and variance σ_{SR} over SR scores of all in-distribution examples seen during training. Set the OOD detection threshold interval as $[\tau_l, \tau_u]$ such that $\tau_l = \mu_{SR} - \lambda \times \sigma_{SR}$ and $\tau_u = \mu_{SR} + \lambda \times \sigma_{SR}$, where λ is a hyper-parameter.

6. Tune the hyper-parameter λ on the validation data to maximize OOD detection accuracy.

The choice of $[\tau_l, \tau_u]$ for OOD detection is motivated by the fractional nature of the seasonal ratio scores. SRS algorithm assumes that in-distribution examples satisfy $p(x|y) =$

$p(r|y)$. Hence, we characterize in-distribution examples with an SR score close to 1, whether from left (τ_l) or right (τ_u) side. To identify in-distribution examples, we rely on SR scores that are close to the mean score recorded during training, whether from left (τ_l) or right (τ_u) side. This design choice is based on the fact that SR score is a quotient ideally centered around the value 1. Indeed, we observe in Figure 7.3 that the SR score for OOD examples can go on either the left or the right side of the SR scores for in-distribution examples. Ideally, the SR score for in-distribution examples is closest to μ_{SR} than SR scores for OOD examples, as illustrated in Figure 7.2. λ is tuned to define the valid range of SR scores for in-distribution examples from $\mathcal{D}_{in}$. We note that the score can be changed easily to consider the quantiles of estimated ratios during training stage and use it to separate the region of OOD and ID score. Therefore, we can redefine $\tau_l = (0.5 - \lambda)$ as the τ_l-quantile for the lower-limit of ID score and $\tau_u = (0.5 + \lambda)$ as the τ_u-quantile for the upper-limit of ID score. Given this definition, we need to tune the hyper-parameter $0 < \lambda \leq 0.5$ on the validation data to maximize OOD detection accuracy. Furthermore, the ID score range is not required to be symmetric. In the general case, we can define $\tau_l = (0.5 - \lambda_l)$ as the τ_l-quantile and $\tau_u = (0.5 + \lambda_u)$ as the τ_u-quantile, where $\lambda_l \neq \lambda_u$. We have observed in our experiments that both these settings give similar performance. Therefore, we only consider $\tau_{u,l} = \mu_{SR} \pm \lambda \times \sigma_{SR}$ for simplicity for our experimental evaluation.

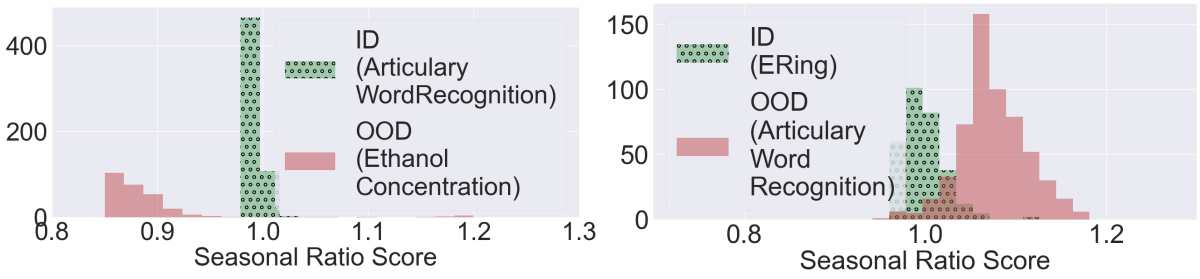


Figure 7.3 Histogram showing the ID and OOD scores along the seasonal ratio score axis. The seasonal ratio scores for OOD examples can be either greater or less than the seasonal ratio scores for ID examples.

Inference stage. Given a time-series signal x, our OOD detection approach works as

follows.

1. Compute the predicted class label $\hat{y}$ using the classifier $F(x)$.

2. Create the remainder component of x with the predicted label $\hat{y}$: $r = x - S_{\hat{y}}$.

3. Compute conditional likelihoods $p(x|\hat{y})$ and $p(r|\hat{y})$ from trained CVAE models $\mathcal{M}_x$ and $\mathcal{M}_r$.

4. Compute the seasonal ratio score using conditional likelihoods.

$$SR(x, \hat{y}) = \frac{p(x|y = \hat{y})}{p(r|y = \hat{y})}$$

5. If the seasonal ratio score $SR(x, \hat{y})$ does not lie within the threshold interval $[\tau_l, \tau_u]$, then classify x as OOD example. Otherwise, classify x as in-distribution example.

7.2.3 Alignment method for improving the accuracy of SRS algorithm

In this section, we first motivate the need for pre-processing raw time-series signals to improve the accuracy of SRS algorithm. Subsequently, we describe a novel time-series alignment method based on dynamic time warping to achieve this goal.

Motivation. The effectiveness of the SRS algorithm depends critically on the accuracy of the STL decomposition. STL method employs fixed-length window over the serialized data to estimate the recurring pattern. This is a challenge for real-world time-series signals as they are prone to scaling, warping, and time-shifts. We illustrate in Figure 7.4 the challenge of scaling, warping, and time-shift occurrences in time-series data. The top-left figure depicts a set of time-series signals with a clear ECG pattern. Due to their misalignment, if we subtract one fixed ECG pattern from every time-series signal, the remainder will be inaccurate. The figures in the left and right column show the difference in the remainder components between the *natural* data (Left) and the *aligned* version of the time-series data (Right). We can clearly observe that the remainder components from the aligned data are more accurate. If input

time-series data is not aligned, it can significantly affect the estimation of $p(r_i|y = y_i)$ and the effectiveness of SRS for OOD detection. Hence, we propose a novel alignment method using the class-wise semantic for the in-distribution data $\mathcal{D}_{in}$ during both training and inference stages.

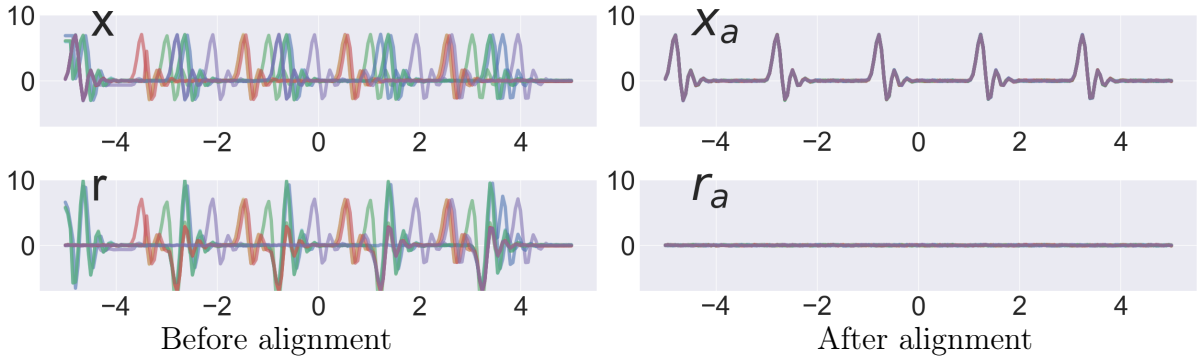


Figure 7.4 Illustration of the challenges in time-series data for STL decomposition: semantic component and remainder. (Left column) Set of natural time-series signals with an ECG wave as semantic component S_y and the corresponding remainders w.r.t S_y. (Right column) Time-series signals and remainders from STL decomposition after applying the alignment procedure.

Time-series alignment algorithm. The overall goal of our approach is to produce a class-wise aligned time-series signals using the ID data $\mathcal{D}_{in}$ so that STL algorithm will produce accurate semantic components S_y for each $y \in \mathcal{Y}$. We propose to employ dynamic time warping (DTW) (Müller, 2007) based optimal alignment to achieve this goal. The optimal DTW alignment describes the warping between two time-series signals to make them aligned in time. It overcomes warping and time-shifts issue by developing a one-to-many match over time steps. There are two key steps in our alignment algorithm. First, we compute the semantic components S_y for each $y \in \mathcal{Y}$ from $\mathcal{D}_{in}$ using STL decomposition. For each in-distribution example $(x_i, y_i) \in \mathcal{D}_{in}$, we compute the optimal DTW alignment between S_{y_i} and x_i. Second, we use an appropriate time-series transformation for each in-distribution example (x_i, y_i) to improve the DTW alignment from the first step. Specifically, we use the time-steps of the longest one-to-many or many-to-one or sequential one-to-one sequence match to select the Expand, Reduce, and Translate transformation as illustrated in Figure

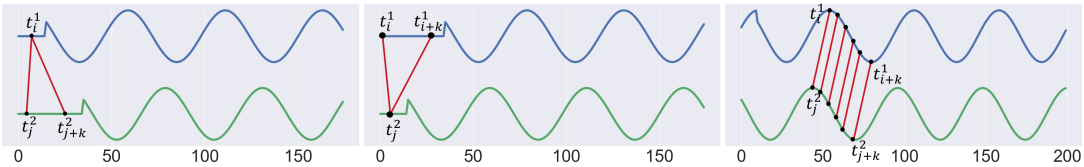
7.5. We define these three time-series transformations below.

Let $X^1 = (t_1^1, t_2^1, \cdots, t_T^1)$ and $X^2 = (t_1^2, t_2^2, \cdots, t_T^2)$ be two time-series signals of length T.

- **Expand**(X^1, X^2): We employ this transformation for a one-to-many time-step matching (t_i^1 is matched with $[t_j^2, \cdots, t_{j+k}^2]$ as shown in Figure 7.5(a)). It duplicates the t_i^1 time-step for k times.

- **Reduce**(X^1, X^2): We employ this transformation in the case of a many-to-one time-step matching ($[t_i^1, \cdots, t_{i+k}^1]$ is matched with t_j^2 as shown in Figure 7.5(b)). It replaces the time-steps $[t_i^1, \cdots, t_{i+k}^1]$ by a single averaged value.

- **Translate**(X^1, X^2): We employ this transformation in the case of a sequential one-to-one time-step matching ($[t_i^1, \cdots, t_{i+k}^1]$ is matched one-to-one with $[t_j^2, \cdots, t_{j+k}^2]$ as shown in Figure 7.5(c)). It translates X_1 to ensure that $t_i^1 = t_j^2$.



(a) One-to-Many (b) Many-to-One (c) One-to-One

Figure 7.5 Illustration of the use of appropriate transformation to adjust the alignment between two time-series signals X^1 (blue signal) and X^2 (green signal).

We illustrate in Figure 7.6 two examples of transformation choices for time-series signal x when aligned with a pattern S. The alignment on the left exhibits that the longest consecutive matching sequence is a one-to-many (x_4 is matched with $[S_2, \cdots, S_7]$) while the alignment on the right exhibits that the longest consecutive matching sequence is a sequential one-to-one ($[x_4, \cdots, x_8]$ is matched with $[S_3, \cdots, S_7]$).

7.3 Experiments and Results

In this section, we present experimental results comparing the proposed SRS algorithm and prior methods on diverse real-world time-series datasets.

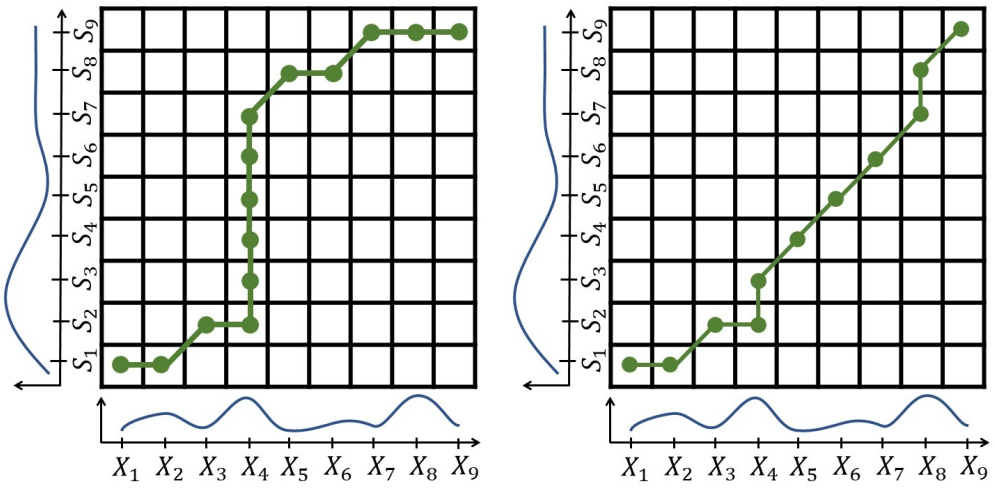


Figure 7.6 Illustration of two transformation choices for a time-series x aligned with a pattern S. (Left) One-to-many as the longest match, calling to use *Expand* transformation. (Right) sequential one-to-one as the longest match, calling to use the *Translate*.

7.3.1 Experimental Setup

Datasets. We employ the multivariate benchmarks from the UCR time-series repository (Dau, Bagnall, et al., 2019). Due to space constraints, we present the results on representative datasets from six different pre-defined domains *Motion*, *ECG*, *HAR*, *EEG*, *Audio* and *Other*. The list of datasets includes Articulary Word Recognition (AWR), Stand Walk Jump (SWJ), Cricket (Ckt), Hand Movement Direction (HMD), Heartbeat (Hbt), and ERing (ERg). We employ the standard training/validation/testing splits from these benchmarks.

OOD experimental setting. Prior work formalized the OOD experimental setting for different domains such as computer vision (Dan Hendrycks, 2017). However, there is no OOD setting for the time-series domain. In what follows, we explain the challenges for the time-series domain and propose a concrete OOD experimental setting for it.

The first challenge with the time-series domain is the dimensionality of signals. Let the ID space be $\mathbb{R}^{n_i \times T_i}$ and the OOD space be $\mathbb{R}^{n_o \times T_o}$. Since we train CVAEs on the ID space, $n_o \times T_o$ needs to match $n_i \times T_i$. Hence, if $n_o > n_i$ or $T_o > T_i$, we window-clip the respective

OOD dimension in order to have $n'_o = n_i$ or $T'_o = T_i$. If $n_o < n_i$ or $T_o < T_i$, we zero-pad the respective OOD dimension in order to have $n'_o = n_i$ or $T'_o = T_i$. Zero-padding is based on the assumption that the additional dimension exists but takes *null* values. The second challenge is in defining OOD examples. Since the number of datasets in UCR repository is large, conducting experiments on all combinations of datasets as ID and OOD is impractical and repetitive (600 distinct configurations for the 25 different datasets considered in this work).

Hence, we propose two settings using the notion of domains.

- **In-domain OOD**: Both ID and OOD datasets belong to the same domain. This setting helps in understanding the behavior of OOD detectors when real-world OOD examples come from the same application domain. For example, a detector of *Epileptic* time-series signals should consider signals resulting from sports activity (*Cricket*) as OOD.

- **Cross-domain OOD:** Both ID and OOD datasets come from two different domains. This configuration is more intuitive for OOD detectors, where time-series signals from different application domains should not confuse the ML model (e.g., *Motion* and *HAR* data).

Our intuition is that the in-domain OOD setting is more likely to occur during real-world deployment. Hence, we propose to do separate experiments by treating every dataset from the same domain as OOD. For the cross-domain OOD setting, we believe that a single representative dataset from the domain can be used as OOD. In this work, we focus on real-world OOD detection for the time-series domain. Since random noise does not inherit the characteristics of time-series data, methods from the computer vision literature have a good potential in detecting random noise.

For improved readability and ease of understanding, we provide Table 7.1 and Table 7.2 to explain the domain labels and dataset labels used in the experimental section of this

chapter along with the corresponding UCR domain name and dataset name.

- Table 7.1 shows the label used to represent a given domain for **Cross-Domain OOD** setting.

  Table 7.1 List of domain labels used in the experimental section and the corresponding UCR domain name.

Domain label	**Domain name**
D1	Motion
D2	ECG
D3	HAR
D4	EEG
D5	Audio
D6	Other

- Table 7.2 shows the label used to represent the dataset used as an OOD source against a given ID dataset for the **In-Domain OOD** setting. For example, while reading Table 7.7, when AWR is the ID distribution, according to Table 7.2, DS1 represents CharacterT. dataset. On the other hand, if HMD is the ID distribution, DS1 represents FingerM. dataset.

Evaluation metrics. We employ the following two standard metrics in our experimental evaluation. **1) AUROC score:** The area under the receiver operating characteristic curve is a threshold-independent metric. This metric (higher is better) is equal to 1.0 for a perfect detector and 0.5 for a random detector. **2) F1 score:** It is the harmonic mean of precision and recall. Due the threshold dependence of F1 score, we propose to use the highest F1 score obtained with a variable threshold. This score has a maximum of 1.0 in the case of a perfect precision and recall.

Table 7.2 Reference table for the In-Domain dataset labels used in the experimental section and the corresponding UCR dataset name. The second column shows the average CVAE normalized reconstruction Mean Absolute Error (MAE) with a negligible variance ≤ 0.001 on the in-distribution data.

In-distriution Dataset name	MAE	**OOD Dataset label**						
		DS1	DS2	DS3	DS4	DS5	DS6	DS7
ArticW. (Motion)	0.025	CharacterT.	EigenW.	PenD.	∅	∅	∅	∅
EigenW. (Motion)	0.000	ArticW.	CharacterT.	PenD.	∅	∅	∅	∅
PenD. (Motion)	0.001	ArticW.	CharacterT.	EigenW.	∅	∅	∅	∅
AtrialF. (ECG)	0.005	StandW.	∅	∅	∅	∅	∅	∅
StandW. (ECG)	0.012	AtrialF.	∅	∅	∅	∅	∅	∅
BasicM. (HAR)	0.024	Cricket	Epilepsy	Handw.	Libras	NATOPS	RacketS.	UWaveG.
Cricket (HAR)	0.010	BasicM.	Epilepsy	Handw.	Libras	NATOPS	RacketS.	UWaveG.
Epilepsy (HAR)	0.030	BasicM.	Cricket	Handw.	Libras	NATOPS	RacketS.	UWaveG.
Handw. (HAR)	0.006	BasicM.	Cricket	Epilepsy	Libras	NATOPS	RacketS.	UWaveG.
Libras (HAR)	0.003	BasicM.	Cricket	Epilepsy	Handw.	NATOPS	RacketS.	UWaveG.
NATOPS (HAR)	0.046	BasicM.	Cricket	Epilepsy	Handw.	Libras	RacketS.	UWaveG.
RacketS. (HAR)	0.026	BasicM.	Cricket	Epilepsy	Handw.	Libras	NATOPS	UWaveG.
UWaveG. (HAR)	0.015	BasicM.	Cricket	Epilepsy	Handw.	Libras	NATOPS	RacketS.
EthanolC. (Other)	0.001	ER.	LSST	PEMS-SF	∅	∅	∅	∅
ER. (Other)	0.044	EthanolC.	LSST	PEMS-SF	∅	∅	∅	∅
LSST (Other)	0.008	EthanolC.	ER.	PEMS-SF	∅	∅	∅	∅
PEMS-SF (Other)	0.525	EthanolC.	ER.	LSST	∅	∅	∅	∅
FingerM. (EEG)	0.048	HandM.	MotorI.	SelfR1.	SelfR2.	∅	∅	∅
HandM. (EEG)	0.006	FingerM.	MotorI.	SelfR1.	SelfR2.	∅	∅	∅
MotorI. (EEG)	0.543	FingerM.	HandM.	SelfR1.	SelfR2.	∅	∅	∅
SelfR1. (EEG)	0.009	FingerM.	HandM.	MotorI.	SelfR2.	∅	∅	∅
SelfR2. (EEG)	0.012	FingerM.	HandM.	MotorI.	SelfR1.	∅	∅	∅
Heartbeat (Audio)	0.011	JapaneseV.	SpokenA.	∅	∅	∅	∅	∅

Configuration of algorithms. We employ a 1D-CNN architecture for the CVAE models required for seasonal ratio scoring (SR) method. We consider a naive baseline where the CVAE is trained on the ID data and the likelihood (LL) is used to detect OOD samples.

We consider a variant of SR scoring (SR_a) that works on the aligned time-series data using the method explained in Section 4.3. We evaluate both SR and SR_a against state-of-the-art baselines and employ their publicly available code: Out-of-Distribution Images in Neural networks (ODIN) (S. Liang, Y. Li, and Srikant, 2018) and Gram Matrices (GM) (Sastry and Oore, 2020) that have been shown to outperform most of the existing baselines; recently proposed Likelihood Regret (LR) score (Z. Xiao, Q. Yan, and Amit, 2020); adaptation of a very recent time-series AD method referred to as Deep generative model with hierarchical latent (HL) (Challu et al., 2022) that does not require labeled anomalies for training purposes. We chose HL as the main baseline to represent time-series AD under OOD setting as it is the state-of-the-art time-series AD algorithm. HL for time-series was shown (Challu et al., 2022) to outperform nearest-neighbor based methods, LSTM-based methods, and other methods (Blázquez-García et al., 2021; Braei and S. Wagner, 2020) in various AD settings.

- **Choice of architecture:** We have experimented with 3 different types of CVAE architecture to decide on the most suitable one for our OOD experiments. We evaluated 1) fully connected, 2) convolutional, and 3) LSTM based architectures using the reconstruction error as the performance metric. We have observed that fully connected networks generally suffer from poor reconstruction performance especially on high-dimensional data. We have also observed that LSTM's runtime during training and inference is relatively longer than the other architectures. However, CNN-based CVAEs delivered both a good reconstruction performance and fast runtime.

- **1D-CNN CVAE details:** To evaluate the the effectiveness of the proposed seasonal ratio (SR) score, we employed a CVAE that is based on 1D-CNN layers. The encoder of the CVAE is composed of 1) A minmax normalization layer, 2) A series of 1D-CNN layers, and 3) A fully-connected layer. At the end of the encoder, the parameters μ_{CVAE} and σ_{CVAE} are computed to estimate the posterior distribution. A random sample is then generated from this distribution and passed on to the CVAE decoder along with

the class label. The decoder of the CVAE is composed of 1) A fully-connected layer, 2) A series of transposed convolutional layer, and 3) A denormalization layer.

- **CVAE Training:** We use the standard training, validation, and testing split on the benchmark datasets to train both CVAEs $\mathcal{M}_x$ and $\mathcal{M}_r$. Both CVAEs are trained to maximize the ELBO on the conditional log-likelihood defined in Section 3 using Adam optimizer with a learning rate of 10^{-4}. We employ a maximum number of training iterations equal to 500. To ensure the reliability of the performance of the proposed CVAEs, we report in Table 7.2 the test reconstruction error of the trained CVAE on ID data using Mean Absolute Error (MAE). We observe clearly that the proposed CVAE is able to learn well the ID space as the reconstruction error is relatively low. To compute the semantic patterns and remainders for in-distribution examples for training $\mathcal{M}_x$ and $\mathcal{M}_r$, we use the STLdecompose[1] python package.

- **Implementation of the baselines:** The baseline methods for ODIN[2], GM[3], HL[4] and LR[5] were implemented using their respective publicly available code with the recommended settings. To employ ODIN and GM, we have trained two different DNN models: a 1D-CNN and a LSTM for classification tasks with different settings. We report the average performance of the baseline OOD detectors in our experimental setting. To repurpose HL method from the AD setting to OOD setting, we have serialized the training data and use it during the training of the generator. For OOD detection at inference time, we serialize both the test ID data and OOD data and shuffle them. By providing the window size equal to the time-steps dimension of the original in-distribution inputs, we execute HL anomaly detection algorithm and report every anomaly as an OOD sample. We employed the default parameters of the generator. As

[1]https://github.com/jrmontag/STLDecompose.git
[2]https://github.com/facebookresearch/odin.git
[3]https://github.com/VectorInstitute/gram-ood-detection.git
[4]https://github.com/cchallu/dghl.git
[5]https://github.com/XavierXiao/Likelihood-Regret.git

recommended by the authors, we use a hierarchical level equal to 4 and 500 iterations for training and inference. We lower the learning rate to 10^{-6} to prevent exploding gradient occurred using the default 10^{-3} value. For a fair comparison, the VAE for Likelihood Regret (LR) has the same architecture as the CVAE used to estimate seasonal ratio (SR) and the naive LL score.

7.3.2 Results and Discussion

Reconstruction error of DGMs. Table 7.3 shows the test reconstruction error of the trained CVAE on ID data using Mean Absolute Error (MAE). We clearly observe that CVAE model is able to learn the ID space as the reconstruction error is relatively low. Table 7.3 shows analogous results for the same CVAE on some OOD data. We observe that DGMs perform well on OOD samples regardless of the different semantics of both ID and OOD data. The pre-trained CVAEs performed well on the OOD AWR dataset with a reconstruction error ≤ 0.1. For the OOD FingerMovement (Fmv) dataset, only two out of the six CVAEs exhibited an intuitive high reconstruction error.

Table 7.3 Average reconstruction error of CVAE is small on both ID and OOD data. The variance is ≤ 0.001.

ID Train Dataset		AWR	SWJ	Ckt	HMD	Hbt	ERg
Error on ID Test		0.025	0.012	0.010	0.118	0.011	0.045
Error on OOD	AWR	$\emptyset$	0.018	0.035	0.039	0.002	0.137
	Fmv	1.658	0.146	0.146	0.039	0.071	6.132

OOD detection via pre-trained classifier and DGMs. Our first hypothesis is that pre-trained DNN classifiers are not well-suited for OOD detection. To test this hypothesis, we train two DNN models: a 1D-CNN and an RNN classifier. We use these models for OOD detection using the ODIN and GM baselines. Table 7.4 shows that AUROC is low on all datasets. For datasets such as HMD and SWJ, the AUROC score does not exceed 0.6 for

any experimental setting. The accuracy of DNNs for time-series classification is not as high as those for the image domain for the reasons explained earlier. Hence, we believe that this uncertainty of DNNs causes the baselines ODIN and GM to fail in OOD detection. Our

Table 7.4 Average AUROC results for ODIN and GMM.

Dataset		AWR	SWJ	Ckt	HMD	Hbt	ERg
In-Domain	ODIN	0.65±0.03	0.54±0.01	0.65±0.03	0.55±0.01	0.71±0.03	0.70±0.03
	GM	0.70±0.03	0.58±0.01	0.64±0.03	0.58±0.02	0.80±0.01	0.75±0.03
Cross-Domain	ODIN	0.55±0.01	0.54±0.01	0.55±0.01	0.50	0.70±0.01	0.70±0.01
	GM	0.56±0.01	0.54±0.01	0.65±0.03	0.55±0.01	0.75±0.03	0.78±0.02

second hypothesis is that DGMs assign a high likelihood for OOD samples is also applicable to time-series. While results in Table 7.3 corroborated this hypothesis, we provide the use of a pre-trained CVAE for OOD detection (LL) in Table 7.7. We observe that AUROC score of LL does not outperform any of the other baselines. Hence, a new scoring method is necessary for CVAE-based OOD detection.

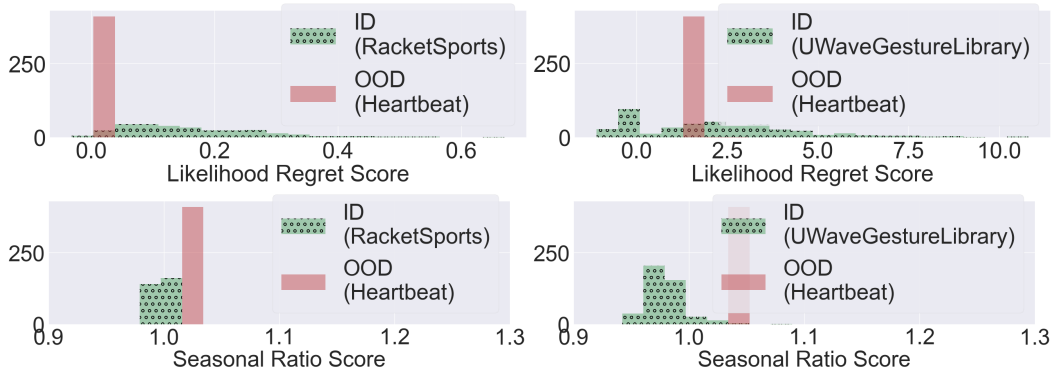


Figure 7.7 Histogram showing the non-separability of ID and OOD LR scores (Top row) and the separability using the seasonal ratio method on real-world time-series data (Bottom row).

Random Noise as OOD. An existing experimental setting for OOD detection tasks is to detect random noise. For this setting, we generate random noise as an input sampled from a Gaussian distribution or a Uniform distribution. Table 7.5 shows the LR baseline

performance on detecting the random noise as OOD examples. We observe in Table 7.5 that LR has an excellent performance on this task. This is explainable as time-series noise does not necessarily obey time-series characteristics. Hence, the existing baselines can perform strongly on the OOD examples. We motivate our seasonal ratio scoring approach for OOD detection based on real-world examples. We have shown in this chapter that existing baselines have poor performance in detecting real-world OOD examples, whereas SR has a significantly better performance.

Table 7.5 Average performance of LR on OOD examples sampled from Gaussian/Uniform distribution.

Dataset	AWR	SWJ	Ckt	HMD	Hbt	ERg
LR	1.00	1.00	1.00	1.00	0.95	1.00

Table 7.6 Results for the validity of Assumption 2. Average distance (MAE and DTW measures) between the semantic pattern from STL S_y and time-series example x with label y from the testing data (with a negligible variance$\leq$ 0.001).

Dataset	AWR	SWJ	Ckt	HMD	Hbt	ERg
MAE	0.047	0.031	0.029	0.078	0.002	0.086
DTW	0.039	0.018	0.022	0.068	0.002	0.069

Results for SR score. The effectiveness of SR score depends on the validity of Assumption 2. Table 7.6 shows both MAE and DTW measure between semantic pattern S_y from STL and different time-series examples of the same class y. We observe that the average difference measure is low. These results strongly demonstrate that Assumption 2 holds empirically. For qualitative results, Figure 7.7 shows the performance of SR in contrast to the performance of Likelihood Regret (LR) shown in Figure 7.7. This illustration shows that SRS provides significantly better OOD separability.

SR score vs. Baselines. Table 7.7 shows the OOD results for SRS and baseline methods. For a fair comparison, we use the same architecture for VAEs computing the LL, LR, and

Table 7.7 AUROC results for the baselines, SR, and SR with time-series alignment (SR_a) on different datasets for both in-domain and cross-domain OOD setting.

		In-domain OOD							**Cross-domain OOD**					
		DS1	DS2	DS3	DS4	DS5	DS6	DS7	D1	D2	D3	D4	D5	D6
AWR (Motion)	LL	0.80	0.85	0.54	∅	∅	∅	∅	∅	0.81	0.80	0.57	0.59	0.81
	HL	*0.50*	0.96	0.94	∅	∅	∅	∅	∅	*0.50*	0.75	0.98	*0.50*	0.56
	LR	0.90	0.95	0.66	∅	∅	∅	∅	∅	0.61	0.84	0.56	0.72	0.61
	SR	0.90	0.97	0.95	∅	∅	∅	∅	∅	1.00	0.97	1.00	1.00	1.00
	SR_a	0.90	**0.97**	**0.95**	∅	∅	∅	∅	∅	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
SWJ (ECG)	LL	0.55	∅	∅	∅	∅	∅	∅	0.61	∅	0.51	1.00	0.77	0.52
	HL	*0.50*	∅	∅	∅	∅	∅	∅	*0.50*	∅	*0.50*	*0.50*	*0.50*	*0.50*
	LR	**0.97**	∅	∅	∅	∅	∅	∅	0.64	∅	*0.50*	1.00	0.67	**0.99**
	SR	0.65	∅	∅	∅	∅	∅	∅	0.70	∅	0.61	1.00	0.96	0.61
	SR_a	0.67	∅	∅	∅	∅	∅	∅	**0.70**	∅	**0.61**	1.00	**0.96**	0.61
Ckt (HAR	LL	0.89	0.85	0.84	0.81	0.82	0.91	0.79	0.79	0.82	∅	0.95	0.90	0.81
	HL	0.97	*0.50*	0.99	*0.50*	*0.50*	0.51	*0.50*	*0.50*	*0.50*	∅	0.52	0.94	*0.50*
	LR	0.81	0.75	0.74	0.71	0.74	1.00	0.78	0.77	0.72	∅	0.95	0.88	0.71
	SR	0.99	0.98	0.99	0.99	0.99	0.98	0.98	0.98	0.99	∅	0.99	0.98	0.98
	SR_a	**0.99**	**0.99**	**1.00**	**1.00**	**1.00**	1.00	**0.98**	**0.98**	**0.99**	∅	**1.00**	**1.00**	**1.00**
HMD (EEG)	LL	0.88	0.88	0.89	0.89	∅	∅	∅	0.89	0.80	0.87	∅	0.90	0.91
	HL	**0.93**	0.57	0.78	0.87	∅	∅	∅	**0.97**	*0.50*	**0.98**	∅	0.58	0.66
	LR	0.68	0.68	0.68	0.68	∅	∅	∅	0.68	0.68	0.68	∅	0.80	0.68
	SR	0.75	0.75	0.75	0.75	∅	∅	∅	0.75	0.75	0.75	∅	0.83	0.75
	SR_a	0.90	**0.90**	**0.90**	**0.90**	∅	∅	∅	0.75	**0.84**	0.89	∅	**0.97**	0.91
Hbt (Audio)	LL	1.00	1.00	∅	∅	∅	∅	∅	0.90	0.95	0.94	0.85	∅	0.98
	HL	*0.50*	*0.50*	∅	∅	∅	∅	∅	0.96	*0.50*	0.78	0.62	∅	0.82
	LR	1.00	1.00	∅	∅	∅	∅	∅	0.93	0.94	0.94	0.75	∅	0.98
	SR	1.00	1.00	∅	∅	∅	∅	∅	0.96	0.97	0.94	1.00	∅	1.00
	SR_a	1.00	1.00	∅	∅	∅	∅	∅	0.96	**0.97**	0.94	**1.00**	∅	**1.00**
ERg (Other)	LL	0.83	0.77	0.75	∅	∅	∅	∅	0.88	0.86	0.82	0.77	0.76	∅
	HL	*0.50*	*0.50*	*0.50*	∅	∅	∅	∅	*0.50*	*0.50*	*0.50*	*0.50*	*0.50*	∅
	LR	0.83	0.72	0.78	∅	∅	∅	∅	0.88	0.78	0.81	0.71	0.78	∅
	SR	1.00	0.98	0.99	∅	∅	∅	∅	0.89	0.99	0.94	0.99	0.99	∅
	SR_a	**1.00**	**1.00**	**1.00**	∅	∅	∅	∅	**0.95**	**1.00**	**0.95**	**1.00**	**1.00**	∅
Absolute		LL 00.0%		HL 05.0%		LR 05.0%			LL 0.00%		HL 06.7%		LR 03.30%	
Wins (%)		Ties 20.0%				**SR_a 70.0%**			Ties 16.7%				**SR_a 73.3%**	
SR_a Improvement (%)		55.0%							40.0%					

SR scores. We make the following observations. **1)** The naive LL method fails to outperform any other approach, which demonstrates that DGMs are not reliable on their own as they

produce high likelihood for OOD samples. **2)** The time-series anomaly detection method HL fails drastically for various OOD settings as reflected by the poor AUROC score of 0.5. This demonstrates that AD methods are not appropriate for OOD detection in the multi-class setting. **3)** SR score outperforms LR score in identifying OOD examples on 80% of the total experiments. This means that the improvement is due to a better scoring function. **4)** For the in-domain OOD setting, AUROC score of LR is always lower than SRS. 3) For the cross-domain setting, SRS outperforms LR in all cases except one experiment on a single dataset SWJ. 4) LR and SRS have the same performance in 20% of the total experiments. Therefore, we conclude that SRS is better than LR in terms of OOD performance and execution time (LR requires new training for every single testing input unlike SR).

Alignment improves the accuracy of SR score. Our hypothesis is that extraction of an accurate semantic component using STL results in improved OOD detection accuracy. To test this hypothesis, we compare SR and SR_a (SR with aligned time-series data). Table 7.7 shows the AUROC scores of SR and SR_a. SR_a improves the performance of SR for around 50% of the overall experiments. For example, on HMD dataset, we observe that SR_a enhances the performance of SR by an average of 15% under the in-domain OOD setting. These results strongly corroborate our hypothesis that alignment improves OOD performance.

SR performance using F1-score. In addition to the AUROC score, we employ F1 score to assess the effectiveness of SR score in detecting OOD. Table 7.8 provides the results comparing SR score and LR score. Like AUROC score evaluation, we make similar observations on F1 score. **1)** SR score outperforms LR score in identifying OOD examples on 60% of the total experiments. This means improvement is due to better scoring function. **2)** For the in-domain OOD setting, F1 score of LR is mostly lower than SR. 3) For the cross-domain setting, SR outperforms LR in 66% of the cases. Hence, we conclude that SR is better than LR in terms of OOD performance measured as F1 metric.

Table 7.8 F1 metric results of LR, SR_a on the different datasets for both In-Domain and Cross-Domain setting. The last two rows show the percentage of datasets where SR_a is out-performing the LR score.

		In-Domain OOD							**Cross-Domain OOD**					
		DS1	DS2	DS3	DS4	DS5	DS6	DS7	D1	D2	D3	D4	D5	D6
AWR (Motion)	LR	0.58	**0.99**	0.80	∅	∅	∅	∅	∅	0.53	0.79	0.61	0.44	0.67
	SR_a	**0.69**	0.89	**0.97**	∅	∅	∅	∅	∅	**0.97**	**0.85**	**1.00**	**0.99**	**1.00**
SWJ (ECG)	LR	0.69	∅	∅	∅	∅	∅	∅	0.98	∅	0.82	1.00	0.97	0.97
	SR_a	0.69	∅	∅	∅	∅	∅	∅	0.98	∅	**0.86**	1.00	0.97	**0.98**
Ckt (HAR)	LR	0.70	0.90	0.97	0.92	0.93	0.91	0.95	0.96	0.48	∅	**1.00**	0.94	0.93
	SR_a	**0.98**	**0.96**	**0.99**	**0.98**	**0.99**	**0.99**	**0.98**	**0.98**	**0.94**	∅	0.99	**0.97**	**1.00**
HMD (EEG)	LR	**0.82**	**0.81**	0.84	0.81	∅	∅	∅	0.84	0.45	0.68	∅	0.80	0.82
	SR_a	0.80	0.75	0.84	0.81	∅	∅	∅	**0.94**	**0.65**	0.68	∅	**0.90**	**0.92**
Hbt (Audio)	LR	1.00	1.00	∅	∅	∅	∅	∅	0.93	0.88	0.94	0.75	∅	0.98
	SR_a	1.00	1.00	∅	∅	∅	∅	∅	0.93	0.88	0.94	**1.00**	∅	0.98
ERg (Other)	LR	0.44	0.85	0.76	∅	∅	∅	∅	**0.88**	0.90	0.79	0.83	0.88	∅
	SR_a	**0.99**	**0.99**	**0.95**	∅	∅	∅	∅	0.87	**0.94**	**0.88**	**0.98**	**1.00**	∅
Wins(%)	LR	15.0%							6.7%					
	Ties	25.0%							26.7%					
	SR_a	60.0%							66.6%					

Inference runtime comparison of the different OOD detection algorithms. We provide in Tables 7.9 and 7.10 a comparison of the number of parameters and the runtime between different OOD detection methods for time-series. Intuitively, both HL and SR methods are characterized by a larger number of parameters than LL and LR as the latter two methods only rely on a single VAE model to compute the OOD score. However, we can observe that LR has the longest score computation runtime: this is due to the new training iterations LR introduces to compute the OOD score of each example. On the other hand, SR algorithm only runs a single inference pass for each example, then computes the ratio between both computed likelihoods. This approach of SR algorithm yields a fast and accurate OOD detector.

Table 7.9 Number of parameters of each DNN used by the different OOD methods.

	Number of parameters
LL	454,628
HL	687,268
LR	454,628
SR	909,256

Table 7.10 OOD Inference runtime comparison.

	Runtime (seconds)					
	AWR (Motion)	SWJ (ECG)	CkT (HAR)	HMD (EEG)	Hbt (Audio)	ERg (Other)
LL	0.66	0.32	0.43	0.49	0.49	0.22
HL	1.06	0.65	0.55	0.81	0.90	0.47
LR	3.72	1.48	1.40	1.35	1.38	1.42
SR	1.5	0.66	0.80	1.01	1.02	0.52

7.4 Summary

In this chapter, we introduced a novel seasonal ratio (SR) score to detect out-of-distribution (OOD) examples in the time-series domain. The key idea of SR scoring is the Seasonal and Trend decomposition using Loess (STL) to extract class-wise semantic patterns and remainders from time-series signals; and estimating class-wise conditional likelihoods for both input time-series and remainders using deep generative models. The SR score of a given time-series signal and the estimated threshold interval from the in-distribution data enables OOD detection. Our strong experimental results demonstrate the effectiveness of SR scoring and alignment method in detecting time-series OOD examples over prior methods.

CHAPTER EIGHT

ALGORITHMS AND THEORETICAL GUARANTEES FOR RELIABLE MACHINE LEARNING FOR WEARABLE ACTIVITY MONITORING

T. Belkhouja*, D. Hussein*, G. Bhat, and J. Doppa. "Reliable Machine Learning for Wearable Activity Monitoring: Novel Algorithms and Theoretical Guarantees". *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2022. (* denotes equal contribution)

Originally published in the *Proceedings of International Conference on Computer-Aided Design*.

Attributions:

T. Belkhouja has contributed to this work by formulating the problem setting from a Machine Learning (ML) perspective, investigating the state of the art related to ML algorithms, formulating the theoretical contribution and the algorithmic solution, implementing the proposed solution as a general algorithm and running the required empirical analysis to highlight its performance improvement compared to the state of the art.

D. Hussein has contributed to this work by formulating the problem setting from the application domain (wearable sensors and mobile health) perspective, investigating the state of the art related to wearable sensors algorithms, formulating the theoretical contribution and the algorithmic solution, implementing the algorithm on microcontrollers and running the required empirical analysis to highlight the performance improvement of the proposed solution compared to the state of the art.

G. Bhat has contributed to this work by investigating the motivation for this work and

the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

J. Doppa has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

ALGORITHMS AND THEORETICAL GUARANTEES FOR RELIABLE MACHINE LEARNING FOR WEARABLE ACTIVITY MONITORING

In this chapter, we propose a real-world applications[1] for robust time-series deep learning algorithms using wearable sensors. Wearable devices that combine multiple sensors, low-power processors, and communication capabilities have the potential to transform fitness, rehabilitation, and health monitoring (Mosenia et al., 2017; Maetzler, Klucken, and Horne, 2016; Limaye and Adegbija, 2018). Human activity recognition (HAR) is an important component of these applications since it enables a fine-grained understanding of the users' activity patterns (Maetzler, Klucken, and Horne, 2016; Zappi et al., 2007; Kim et al., 2016). HAR approaches collect sensor data from a set of training users and employ the data to train a classifier using ML algorithms (Lara and Labrador, 2012). The classifier is then used to recognize the activities of the users. During the data collection step, the sensors are placed at fixed positions on the body. For instance, a number of prior HAR approaches place the sensors on the hip or waist of the user. For achieving high real-world accuracy, the sensors must be placed at the same location as the training setup to ensure that the distribution of the sensor data is the same (Kunze and Lukowicz, 2014). We propose to address in this chapter the challenge of the distribution change of the sensor data at runtime. Approaches to handle sensor disturbances can be classified into two main categories depending on whether they pre-process sensor data at runtime or augment training data. The first class of approaches to handle sensor disturbances require calibration and pre-processing at runtime (Mizell, 2003; Kunze, Lukowicz, et al., 2009). The calibration step adds an additional burden to the user *each* time the device is used, while the pre-processing steps increase the execution time and energy overhead for resource-constrained wearables (Yamin, Bhat, and Doppa, 2022; Hussein, Bhat, and Doppa, 2022). Therefore, we propose a concrete instantiation of the statistical optimization approach in (Belkhouja and Doppa, 2022) to enable reliable ML

[1]This work was developed in close collaboration with Dina Hussein and Ganapti Bhat

classifiers for HAR on low-power wearable devices. Starting with a set of training data and known candidate disturbances, the StatOpt approach employs statistical transformations to generate additional training examples that describe the sensor disturbances. This approach ensures that the ML classifier automatically learns to provide reliable activity classification in the presence of sensor disturbances.

8.1 Background and Problem Setup

This section first provides the background on HAR and introduces candidate sensor disturbances in HAR. Then we describe the problem setup for creating reliable ML classifiers for wearable activity monitoring. Figure 8.1 provides a high-level overview of the complete problem setup and the proposed StatOpt approach for reliable HAR.

8.1.1 Human Activity Recognition Preliminaries

HAR algorithm development involves three major steps as follows:

Sensor data collection and labeling: The first step in the development of HAR systems is to collect sensor data when users are performing the activities of interest. In this work, we consider accelerometer and stretch sensors since they are used in our experimental datasets. This is typically done in a laboratory environment where an expert places the sensors at appropriate locations on the body. After placing sensors, the expert collects sensor data while the user performs a pre-defined set of activities. Finally, the data is labeled to record the activity for the duration of the experiment.

Segmentation and feature generation: The second step after data collection is to segment the data into fixed- or variable-length windows for classification (A. Wang et al., 2016). Next, features are extracted from each activity segment: handcrafted features for conventional ML algorithms such as decision trees or learned directly from the raw sensor data using deep neural networks (DNNs) (Ismail Fawaz et al., 2019).

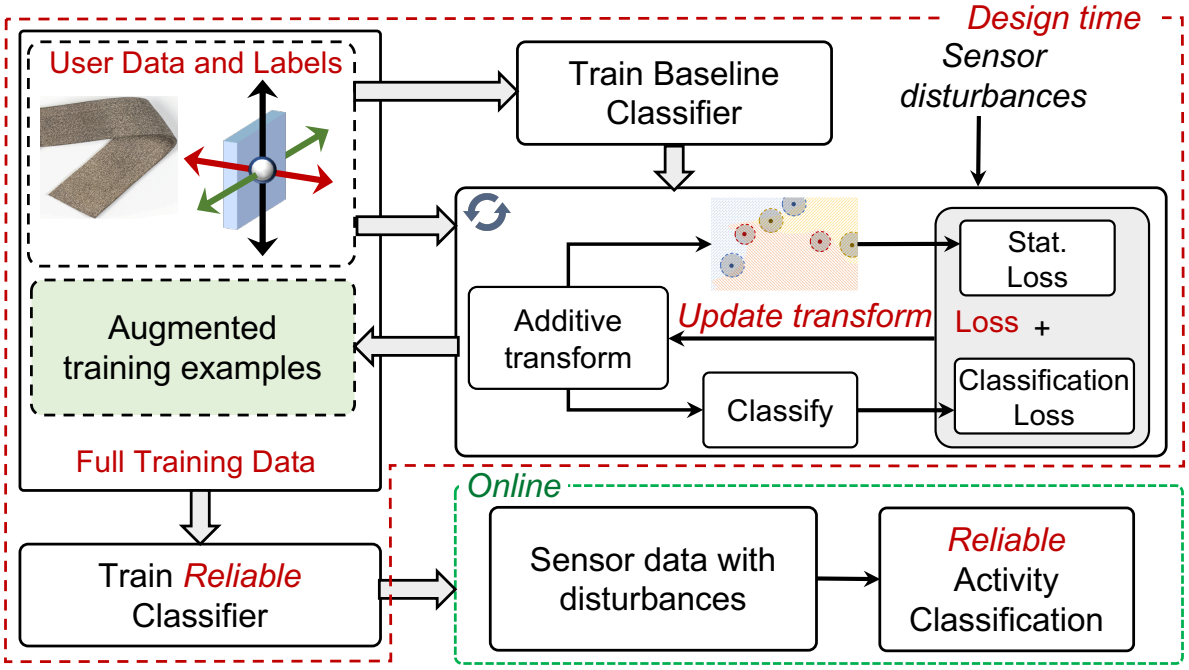


Figure 8.1 Overview of the proposed StatOpt approach. We start with the observed sensor data and train a baseline classifier. The baseline classifier, sensor data, and disturbances are provided as inputs to the StatOpt approach. At the end of the optimization, we obtain training examples that capture the overall structure of the sensor disturbances. The new training examples are combined with the observed sensor data to train a reliable HAR classifier. Finally, the reliable classifier is used to perform HAR in real-world settings.

Classifier training: The final step in the HAR design is to train the supervised learning classifier that takes the features of the user state s, $\phi(s) \in \Re^d$, as the input to predict the current activity $\hat{y}$. The labels assigned in the data collection step y^* are used as the ground truth data for the classifier training.

At runtime, the wearable device collects and segments sensor data. The trained classifier is then used on the segmented data to identify the activities of the user in their daily life.

8.1.2 Sensor Disturbances in HAR

State-of-the-art HAR approaches are trained on the assumption that the sensor position and data distribution will be the same during training and real-world deployment. However, in real-world usage, the data distribution of sensors can shift due to changes in sensor position,

sampling frequency, or energy constraints. Next, we summarize the sensor disturbances considered in this work.

On body orientation changes: Orientation changes occur when the sensor rotates from its initial position. Without loss of generality, consider that a sensor with three axes of measurement (e.g., accelerometer) is being used to identify the user activities, as shown in Figure 8.2(a). If the sensor is mounted on the legs, it captures motion in the forward, lateral, and vertical directions. The device may experience rotation along the vertical and lateral axes, as shown in Figures 8.2(b) and (c). We refer to these rotations as *heading* and *pitch rotation*, respectively. The *heading rotation* changes the sensor values in the forward and lateral directions, while the *pitch rotation* changes the values in the forward and vertical directions.

On body position changes: In addition to the orientation change, the sensor can move up or down from the desired position. For instance, a sensor mounted on the knee of a user may slip and move to the ankles. These changes can impact the amplitude of the signal observed by the sensors. We refer to this as *amplitude disturbance*.

Sensor hardware disturbances: Wearable devices operate under extremely limited energy budgets. During periods with insufficient energy, the energy management algorithm on the wearable devices reduces the sensor sampling frequency. The lower sampling frequency, in turn, reduces the fidelity of the sensor data. We refer to this disturbance as *sampling rate uncertainty*.

Next, we describe our general problem setup to overcome the sensor disturbances.

8.1.3 Problem Setup

Let $X \in \mathbb{R}^{n \times T}$ be the time-series sensor data in each activity window, where n is the number of sensor channels and T is the number of samples in each activity window. The activity label for each activity window is denoted by y^*. We denote the set of training examples $\mathcal{D}$

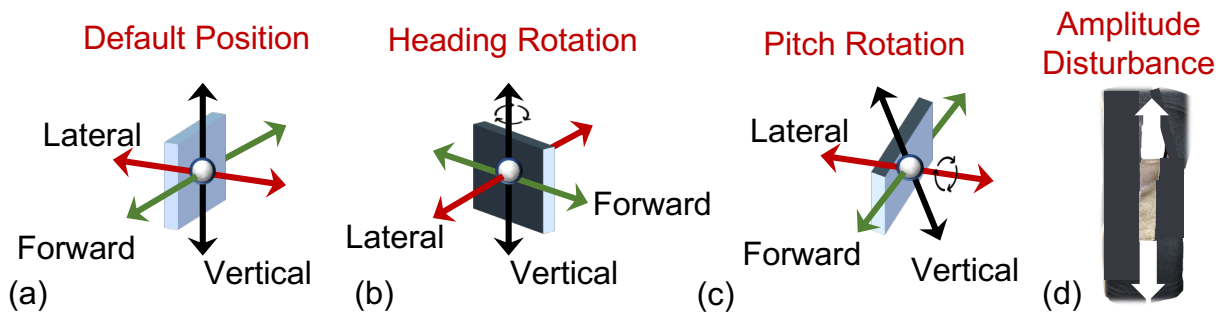


Figure 8.2 Illustration of sensor orientation and position changes.

as several input X and ground truth output y^* pairs. Standard ML algorithms use the given data $\mathcal{D}$ to train a classifier F_θ which takes time-series X as input to predict the corresponding activity label $\hat{y}$, where θ stands for parameters of the HAR classifier.

Next, let the set $\mathcal{S}$ denote the disturbances present in the sensor data. Each of these disturbances changes the sensor data $X \in \mathbb{R}^{n \times T}$ collected at runtime, which results in misclassifications by F_θ. To address this challenge, our goal is to design a reliable classifier F_θ^R that provides accurate classifications in the presence of one or more sensor disturbances. We achieve this by creating additional training examples $\mathcal{D}'$ using a statistical optimization approach StatOpt as follows. For each time-series input X from training data $\mathcal{D}$, StatOpt creates X' using an additive perturbation of X to accurately capture the distribution of sensor disturbances while maintaining the same activity label. The additional data $\mathcal{D}'$ is then combined with $\mathcal{D}$ to train a reliable HAR classifier F_θ^R.

8.2 HAR Related Work

The applicability of HAR in applications ranging from health monitoring to fitness has led to increased attention from both academia and industry (Lara and Labrador, 2012; Dempsey, 2015). In particular, advances in wearable technology have made it possible to perform HAR using on-body sensors (Lara and Labrador, 2012).

HAR approaches typically collect sensor data in a controlled laboratory environment (Kunze and Lukowicz, 2014). The collected sensor data is then used to train a ML classifier

(Lara and Labrador, 2012; Bhat et al., 2020; Zappi et al., 2007). The classifiers are trained with the assumption that the sensor data distribution during real-world deployment is same as the training data. However, the sensor data distribution can change at runtime due to a number of reasons such as user movement or error and energy limitations. The sensor disturbances, in turn, lead to a degradation of the classification accuracy (Stisen et al., 2015; Kunze and Lukowicz, 2014; Shi et al., 2020; Barshan and Yurtman, 2020). Therefore, there is a great need to create classifiers that provide reliable HAR under sensor disturbances.

Prior work has investigated methods to detect and correct sensor disturbances at runtime (Kunze, Lukowicz, et al., 2009; Mizell, 2003; Stisen et al., 2015; Hussein, Jain, and Bhat, 2022). Most of these approaches involve calibration and pre-processing of the sensor data to mitigate the disturbance. For instance, the approach in (Kunze, Lukowicz, et al., 2009) requires the user to walk a few steps every time the wearable device is used to identify the orientation of the user and account for any changes from the design time. Similarly, the technique in (Stisen et al., 2015) uses clustering and interpolation to account for sampling rate disturbances in HAR. While these techniques are useful, they add significant runtime overhead and cause inconvenience to the user, hindering the real-world deployment of wearable enabled applications.

Recent techniques have also proposed HAR methods that aim to be invariant to sensor position changes (Shi et al., 2020; Barshan and Yurtman, 2020). For instance, the approach in (Barshan and Yurtman, 2020) focuses on extracting position independent sequences from sensors placed on rigid body parts (e.g. arms). However, it does not eliminate the additional pre-processing of sensor data and is limited to body parts that are rigid in nature.

One of the most promising approaches to handle runtime disturbances is by including examples with candidate disturbances in the training data (Dietterich, 2017; Carlini and D. Wagner, 2017). For instance, data augmentation techniques (Carlini and D. Wagner, 2017) employ adversarial algorithms to create worst-possible input examples from the classifier's perspective. However, data augmentation without accounting for the specific characteristics

of the sensor data can lead to accuracy degradation. If the augmented training examples are similar to the observed sensor data of a different activity, it leads to ambiguous/inconsistent training data for the classifier. To address these unique challenges, StatOpt employs statistical optimization to ensure that additional training examples capture the disturbances while preserving the statistical properties of the original time-series data. As a result, classifiers trained with augmented data from StatOpt provide reliable classification under natural disturbances.

8.3 Statistical Optimization Approach

This section describes the statistical optimization approach *StatOpt* to significantly improve the reliability of ML-based HAR classifiers to sensor disturbances. StatOpt is an instantiation of the general statistical optimization framework (Belkhouja and Doppa, 2022) for ML-based HAR on wearable systems. The goal of StatOpt is to create new training examples to capture the overall structure of natural sensor disturbances that may occur in sensor observations and use them to improve the classifier reliability. Figure 8.3 provides a conceptual illustration of the key intuition behind the StatOpt algorithm. We aim to create new training examples to cover the natural disturbance region around time-series sensor input to improve the reliability of the ML classifier. We achieve this goal by optimizing a loss function that describes the natural sensor disturbances region where the ML classifier fails to predict the correct class label.

Intuition: As shown in Figure 8.3, we would like to create new training examples to describe the natural disturbances of a time-series sensor observation X, which are misclassified by the HAR classifier. We define the candidate sensor disturbances that can affect the time-series $X \in \mathbb{R}^{n \times T}$ as additive perturbations $p \in \mathbb{R}^{n \times T}$. Therefore, the new training example is X' $= X + p$. To improve the reliability of HAR classifier, the perturbation p should satisfy two main characteristics: *1) Misclassification:* The new example $X' = X + p$ is misclassified

by the HAR classifier (i.e., $F_\theta(X') \neq y^*$); and *2) Natural disturbance:* p describes natural sensor disturbances that may affect the observed time-series input. Due to the huge variety in disturbances, it is impractical to define p explicitly. Hence, we propose to use a statistical optimization approach for data-driven estimation of p w.r.t a given HAR classifier and a time-series X.

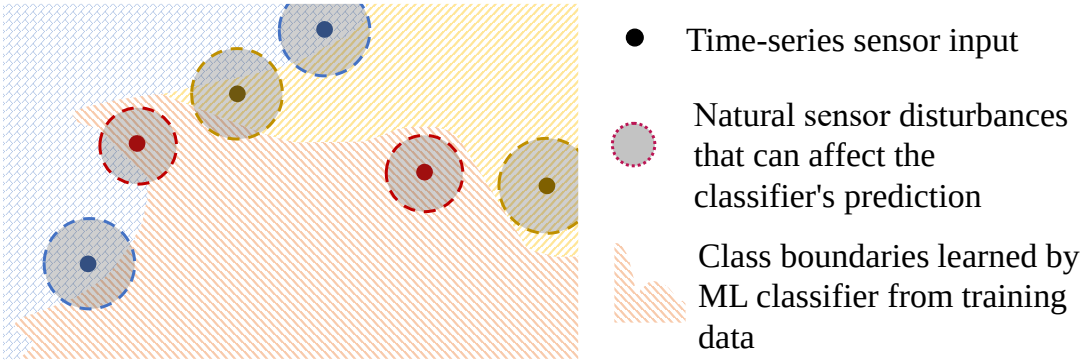


Figure 8.3 Conceptual illustration of the sensor disturbance regions for different time-series inputs within three classes shown in blue, orange, and yellow colors. The dotted circles represent the potential disturbance regions that may affect an input example. The dotted circles covering multiple class regions show the vulnerability of the ML classifier against natural disturbances. Our goal is to create training examples sampled from the dotted circles to improve the reliability of ML classifier against natural sensor disturbances.

8.3.1 Statistical Optimization Algorithm

We propose a novel approach to create new training examples $\mathcal{D}'$ using the given training data $\mathcal{D}$ and HAR classifier F_θ learned from $\mathcal{D}$. The combined training data $\mathcal{D} \cup \mathcal{D}'$ yields a HAR classifier that is reliable to natural sensor disturbances. For each training example $(X, y^*) \in \mathcal{D}$, we create additional examples of the form $\{(X' = X + p, y^*)\}$. We propose to employ constraints on the statistical features of X' to describe the natural sensor disturbances that can affect X. Our approach assumes that p can disturb the sensor input X freely under the condition of preserving the original statistical features of X. The goal of this perturbation is to yield a reliable HAR classifier against a variety of sensor disturbances at once (we do

not define an explicit form for p) in a controlled manner (we use statistical constraints to define p). In the context of wearable activity monitoring, we call to use the *Body Acceleration* feature described below. We assume that any candidate sensor disturbance has low effect on the body acceleration of the input. Furthermore, we propose to maintain similar distribution of X and X' values by employing the skewness and the kurtosis features. Our goal is to create a perturbation p such that $\|S_i(X) - S_i(X')\| < \epsilon$ for each statistical feature $S_i \in$ {Body acceleration, Skewness, Kurtosis }, where ϵ is a small threshold:

- Body acceleration is computed using the 3 axes (X_1, X_2, and $X_3 \in \mathbb{R}^T$) of the accelerometer $\sqrt{\sum_{i=1}^{3} X_i^2} - 1$.
- Skewness measures the symmetry of the distribution of the values of the input and is computed on every channel: Skewness(X_i)=$\left(\sum_{j=1}^{T}(X_{i,j} - \mu(X_i))^3/T\right) / (\sigma(X_i)^3)$.
- Kurtosis measures the distribution tail of the input's values and is computed on every channel:
  Kurtosis(X_i)=$-3 + \left(\sum_{j=1}^{T}(X_{i,j} - \mu(X_i))^4/T\right) / (\sigma(X_i)^4)$.

where μ and σ represent the mean and standard deviation of the time-series signal respectively. To achieve this goal, we propose a new statistical loss:

$$\mathcal{L}^{stat}(p, X) \triangleq \sum_{S_i} \|S_i(X + p) - S_i(X)\| \tag{8.1}$$

This loss function overcomes the impractical use of projection functions on the statistical feature space and guarantees that the perturbation p preserves the original statistical features of X. Indeed, our empirical results in Figure 8.4 show that the real-world sensor disturbances preserve the three statistical features. This provides a strong justification for the key intuition behind our statistical optimization approach and the statistical loss shown in Equation 8.1.

Overall StatOpt algorithm: To create a new example X' that is misclassified by the HAR classifier, we propose to estimate the confidence of the classifier to predict a given class label

$y' \neq y^*$ by computing the following loss function:

$$\mathcal{L}^{label}(p, X, y') = \max\left[\max_{y \neq y'}\left(\mathcal{Z}_y\left(X+p\right)\right) - \mathcal{Z}_{y'}\left(X+p\right), \ \ \rho\right] \tag{8.2}$$

where $\rho < 0$ and y' is a random class label different from y^*. This loss function will ensure that the new example $X'{=}X + p$ will be classified by the HAR classifier F_θ as class y' with a confidence $|\rho|$ using the output of the logits layer $\{\mathcal{Z}_y\}_{y \in Y}$. Equation 8.2 ensures that the logit output value corresponding to a class label $y' \neq y^*$ remains less than the output value of other class labels including y^*. Hence, the classifier is guaranteed to predict y' for the input X. The role of this loss function is to actively find naturally disturbed examples of X that the HAR classifier F_θ does not classify as y^*. These examples uncover scenarios where the classifier is prone to fail and help in improving the reliability of the classifier F_θ^R.

The final loss function $\mathcal{L}$ that we want to minimize to obtain the perturbation p is as follows:

$$\mathcal{L}(p, X) = \mathcal{L}^{label}(p, X, y') + \mathcal{L}^{stat}(p, X) \tag{8.3}$$

This loss function captures both desired misclassification and natural sensor disturbance characteristics of p. We employ a gradient descent based optimizer to minimize the loss function in Equation 8.3 over p. The prior knowledge about sensor disturbances $\mathcal{S}$ (when available) can be used to constrain the optimization variables of perturbation p over a subset of channels G_{ch}, i.e., those outside $\mathcal{S}$ will be set to zero (no perturbation for the channels not in G_{ch}). The parameter ρ introduced in Equation 8.2 plays an important role here. ρ will push gradient descent to minimize mainly the second term when the first one plateaus at ρ first. Otherwise, the gradient can minimize the general loss function by pushing $\mathcal{L}^{label}(p, X)$ to $-\infty$, which is counter-productive for our goal. The result of this optimization is new input examples that are used during the training phase in addition to the original examples to obtain a reliable HAR classifier against real-world sensor disturbances. We present an overall description of the proposed StatOpt-based reliable classifier training approach in Algorithm 8.

Algorithm 8 StatOpt-based Training for Reliable HAR Classifier

Input: Training set $\mathcal{D} = \{(X, y^*)\}$; F_θ, pre-trained HAR classifier on $\mathcal{D} = \{(X, y^*)\}$; $\{G_{ch}\}$, group of different possible channels to disturb; MAX, maximum iterations for gradient descent; ϵ, threshold for statistical features

Output: A reliable HAR classifier F_θ^R

Initialize the set of additional training examples $\mathcal{D}' \leftarrow \emptyset$
for each training example $(X, y^*) \in \mathcal{D}$ **do**
  for every $G_{ch} \in \{G_{ch}\}$ **do**
    Random initialization of perturbation $p \in \mathbb{R}^{n \times T}$
    Fix the channels of p not in G_{ch} to 0
    Fix a random class-label $y' \neq y^*$
    for k=1, $\cdots$, MAX **OR** $\mathcal{L}^{stat}(p, X) < \epsilon$ **do**
      Compute loss $\mathcal{L}(p, X) = \mathcal{L}^{label}(p, X, y') + \mathcal{L}^{stat}(p, X)$
      Estimate the gradient $\nabla_p \mathcal{L}(p, X)$
      Perform gradient descent and update p
    end for
    Add new training example $(X' = X + p, y^*)$ to $\mathcal{D}'$
  end for
end for
Use the combined training set $\mathcal{D} \cup \mathcal{D}'$ to train classifier F_θ^R
Return reliable HAR classifier F_θ^R

8.3.2 Theoretical Analysis

In this section, we derive theoretical upper bounds for sensor data disturbances for which StatOpt will produce HAR classifiers with reliability certificates (analogous to security certificates for software). Reliability/robustness certification (Bai Li et al., 2019) is a theoretically-sound upper bound that describes the maximum perturbation that can affect any input example without a change in the classifier's prediction. Therefore, if a classifier has a certification equal to E^{lim} for a given input X, then for any perturbation p such that $||p|| \leq E^{lim}$, the classifier's prediction for X and $X + p$ will remain the same (i.e., $F_\theta(X) = F_\theta(X + p)$).

Existing certification methods (Bai Li et al., 2019; Cohen, Rosenfeld, and J Zico Kolter, 2019) rely on the Euclidean distance to constrain the disturbance p. If we enforce l_2 certification in our time-series setting (i.e., for any $||p|| \leq E^{lim}$, $F_\theta(X) = F_\theta(X + p)$), then we risk to greatly confuse the classifier with ambiguous training examples as explained earlier. Table 8.1 shows results on WISDM dataset to corroborate this hypothesis noting that we see similar results on other real-world datasets. We show for some reference classes y^{ref} and a disturbance p describing the rotation at 210 degrees that the minimum l_2 distance between any time-series input X with an activity label y^{ref} and $X + p$ is significantly greater than the minimum l_2 distance between X and any other time-series input X' that has a different activity label $y \neq y^{\text{ref}}$. Therefore, if $E^{lim} > ||p||$, then the classifier will predict y^{ref} for all such time-series inputs, resulting in a classifier with poor accuracy. In summary, we conclude that the certification cannot be based on l_2 distance.

Therefore, we propose to derive a new certification guarantee that is suitable for our problem setting. The certification proposed in (Bai Li et al., 2019) constrains the Euclidean distance (l_2) between the inputs using:

1. The Rényi divergence of Gaussian distribution ($\mathcal{N}(x \in \mathbb{R}^T$, $\sigma^2 \in \mathbb{R})$) on univariate input space X_1 and X_2 is:
   $D(\mathcal{N}(X_1, \sigma^2)||\mathcal{N}(X_2, \sigma^2)) = \alpha \times l_2(X_1, X_2)$, where α is a constant.

2. The statement that $D(F_\theta(\mathcal{N}(X+p, \sigma^2))||F_\theta(\mathcal{N}(X, \sigma^2)))$ is always greater than a value E at which the classifier is not reliable, i.e., for $p \sim \mathcal{N}(\mu(p), \sigma^2)$ and $\epsilon > 0$, we have $Pr[F_\theta(X) \neq F_\theta(X + p)] \geq 1 - \epsilon$ (Cohen, Rosenfeld, and J Zico Kolter, 2019).

Using the above two statements, one can derive the certification (Bai Li et al., 2019) to constrain the perturbation p such that $l_2(X, X + p) < E/\alpha$. Such a constraint guarantees the reliability of the classifier when the perturbation is within this bound. However, this certification is not suitable for our problem setting for two main reasons: 1) We have already shown that l_2-based certification is impractical; and 2) The certification cannot generalize to

a multivariate input space such as the setting of this chapter. Therefore, we propose to use the appropriate Gaussian distribution ($\mathcal{N}(x \in \mathbb{R}^{nxT}, C \in \mathbb{R}^{nxn})$)) to derive the certification needed for our setting.

Table 8.1 Comparison of the minimum l_2 distance between examples from different classes.

	Rotated at 210°	Observed	
Reference Class	Same Class	Closest Class	Farthest class
Walking	9.57	3.91	8.84
Jogging	14.4	7.26	8.84
Upstairs	10.3	4.07	8.23

Theorem 7. *Let $X \in \mathbb{R}^{n \times T}$ be an input time-series signal, $\mu(X)$ be the mean value of X and C a given covariance matrix. Let E be the minimum Rényi divergence $D(F_\theta(\mathcal{N}(X + p, C))||F_\theta(\mathcal{N}(X, C)))$ at which the classifier is still reliable against a perturbation $p \in \mathbb{R}^{n \times T}$. We can estimate the certification upper bound ($||\mu(p)||_{max}$) of the mean feature of a disturbance p by $\boldsymbol{E = \alpha_\mu \times ||\mu(p)||^2_{max}}$, where $\boldsymbol{\alpha_\mu}$ is a constant.*

8.4 Experiments and Results

In this section, we describe our experiments and discuss results along different dimensions.

8.4.1 Experimental Setup

Wearable device: We use a device based on the TI CC2652R MCU for our experiments (Texas Instruments Inc., 2018). The MCU integrates an ARM Cortex-M4 processor to perform sensor data processing and classification. We implement the reliable classifier and baseline approach on the MCU to measure the execution time and energy consumption.

Datasets: We use two publicly available datasets to evaluate the performance of StatOpt.

w-HAR (Bhat et al., 2020): The w-HAR dataset (Bhat et al., 2020) includes accelerometer and stretch sensor data for human activity recognition from 22 users to recognize eight activities. The eight activities monitored by the w-HAR dataset are: {*Jump, lie down, sit, stand, walk, stairs up, stairs down, and transition*}. We apply the orientation disturbance to the accelerometer and position change to the stretch sensor in the w-HAR dataset. Moreover, sensor hardware disturbance is applied to the accelerometer by varying its sampling frequency. We choose the accelerometer for hardware disturbance since it has a higher power consumption than the stretch sensor (Bhat et al., 2020).

WISDM (Kwapisz, Weiss, and Moore, 2011): The WISDM dataset provides data for six activities: {*walking, jogging, ascending stairs, descending stairs, sitting, and standing*} from twenty nine users using the accelerometer sensor on a smartphone. The accelerometer is placed in the front pant pocket of the user. Since the pockets of pants are larger than a typical smartphone, the device may experience orientation changes within the pocket, thus leading to sensor data disturbance. Consequently, we apply the orientation disturbance to the WISDM dataset.

HAR classifier representation: We use a 1-D convolutional neural network (CNN) as the HAR classifier. Specifically, we use a 1-D CNN with one conv. and max-pooling layers, a flatten layer, and two fully connected layers with the ReLU activation. To train the classifier, we use the Adam optimizer (Kingma and Ba, 2015) for 50 epochs. We also note that the additional training examples generated by StatOpt can be used to improve the performance of a wide range of classifiers. To demonstrate this, we also train HAR classifiers using decision trees and random forest using data generated by StatOpt.

Evaluation metrics: We evaluate activity recognition accuracy and the overhead of the StatOpt-based reliable classifier with respect to the baseline approach that uses calibration and pre-processing to account for the sensor disturbances. The goal of our evaluation is to verify whether the reliable classifier achieves accuracy equal to a classifier without any sensor

disturbances with negligible overhead. Furthermore, we analyze the quality of the additional data generated by StatOpt using t-distributed stochastic neighbor embedding (t-SNE) visualization (Van der Maaten and Hinton, 2008) and the Maximum Mean Discrepancy (MMD) metric (Tolstikhin, Sriperumbudur, and Schölkopf, 2016). The MMD metric measures the distance between data distributions by estimating the mean embeddings of features in a high-dimensional space where matching distributions will be closer and dissimilar distributions will be farther.

8.4.2 Baseline Methods for Comparison

This section describes the baseline data recovery and augmentation methods that we use to validate the effectiveness of StatOpt.

Pitch rotation recovery: We use the method proposed in (Mizell, 2003) to recover pitch rotation. The method relies on the fact that when the user is standing, the only acceleration experienced by the device is due to gravity. Therefore, we obtain the average acceleration in each direction when the user is standing to determine the direction of the gravity vector. Then, the design time gravity vector is used to calculate the degree of pitch rotation and recover the sensor data.

Heading rotation recovery: The heading rotation changes the direction of motion observed by the sensor. For instance, if one of the accelerometer axes is pointing in the direction of motion, any heading rotation will result in the projection of the motion acceleration on two axes. To overcome this, we use the approach in (Kunze, Lukowicz, et al., 2009). The approach starts by performing offline characterization of the direction of motion using principal component analysis (PCA) when the user is walking. At runtime, the device instructs the user to walk a few steps to calculate the new PCA components of motion. The angle between the reference and runtime PCA components is used as the degree of heading rotation to recover the sensor data.

On body position recovery: On body position changes affect the amplitude of the stretch

sensor in the w-HAR dataset. To recover the amplitude change, we perform an offline characterization for each user to obtain the nominal sensor value when the user is standing. At runtime, the sensor is calibrated when the user is standing to determine the amplitude degradation due to the position change. The calibration is then applied to new sensor data from the user.

Sampling rate recovery: To overcome sampling rate uncertainty, the sensor data is re-sampled to the frequency used during training.

In summary, the baseline approaches require extensive calibration and pre-processing of the *runtime* data to compensate for the disturbances. This overhead makes them impractical for use in real-world settings where the energy budget is limited.

8.4.3 Baseline Data Augmentation Method

Data augmentation is a standard method to augment the training data to improve the generalization of a given classifier. Therefore, we compare the performance of StatOpt with the CW method (Carlini and D. Wagner, 2017) to highlight the role of statistical features in enhancing the reliability of HAR classifiers. Specifically, we use the same pre-trained baseline classifier employed for StatOpt to generate different CW-based examples for data augmentation. The additional training data generated by CW method is then used to train an activity classifier.

8.4.4 Evaluation of StatOpt-based Training Data

This section evaluates the quality of the training data generated by StatOpt along two dimensions. We first compare the statistical features of the generated time-series examples with real-world sensor data. We also compare the quality of the data generated by the CW method. Next, we empirically evaluate the theoretical upper-bound derived in Section 4.2 for the data generated by StatOpt.

Statistical Analysis for Training Data One of the goals of StatOpt is to generate ad-

ditional training examples that capture the overall structure of natural sensor disturbances while maintaining the statistical properties of the original sensor data and ground truth activity labels. This is an important goal because the augmented data samples that encode sensor disturbances must have the same activity label as the original sensor data to learn highly-effective HAR classifiers. In Figure 8.4, we show the distribution of the difference ($\|S_i(\text{Original data}) - S_i(\text{Disturbed data})\|$) for the three statistical features, namely, body acceleration, skewness, and kurtosis, between the disturbed sensor data and the corresponding original time-series sensor data. We observe that the disturbed data indeed preserves the statistical features of the original data, as the difference is well-centered at the value 0. Since StatOpt by design preserves the statistical features of the original time-series sensor data, the additional training examples generated by StatOpt capture the overall distribution of the real-world sensor disturbances.

We further analyze the data generated by StatOpt to assess its similarity to the real-world perturbations. First, we employ the t-SNE method to visualize the data generated by StatOpt, data with sensor disturbances, and the observed sensor data. t-SNE is a dimensionality reduction technique that aids in the visualization of high-dimensional data by reducing the data to two or three dimensions. If two distributions are close to each other, their samples will be close in the t-SNE map while dissimilar distributions will have larger distances between the samples. Second, we quantify the MMD distance between the generated StatOpt distribution to the original input distribution and rotated data distribution.

Figure 8.4(d) shows an illustration of t-SNE applied on the original w-HAR data, pitch and heading rotation, and the data generated by StatOpt. We see that there is a significant separation between the original data and the sensor data with orientation changes, indicating a change in sensor data distribution. In contrast, the examples generated by StatOpt overlap with the original data, indicating that the distribution of the data is close to the original training sensor data. This observation is further supported by Table 8.2, which shows the MMD distance between the original data and the disturbed or StatOpt data. The table

shows that the StatOpt distribution is closer to the original data than the rotated data. Additionally, we note that the MMD distance between StatOpt and the rotated data is the smallest. Both these empirical observations ensure that the HAR classifier learns the sensor disturbances correctly to provide reliable predictions at runtime without additional overhead.

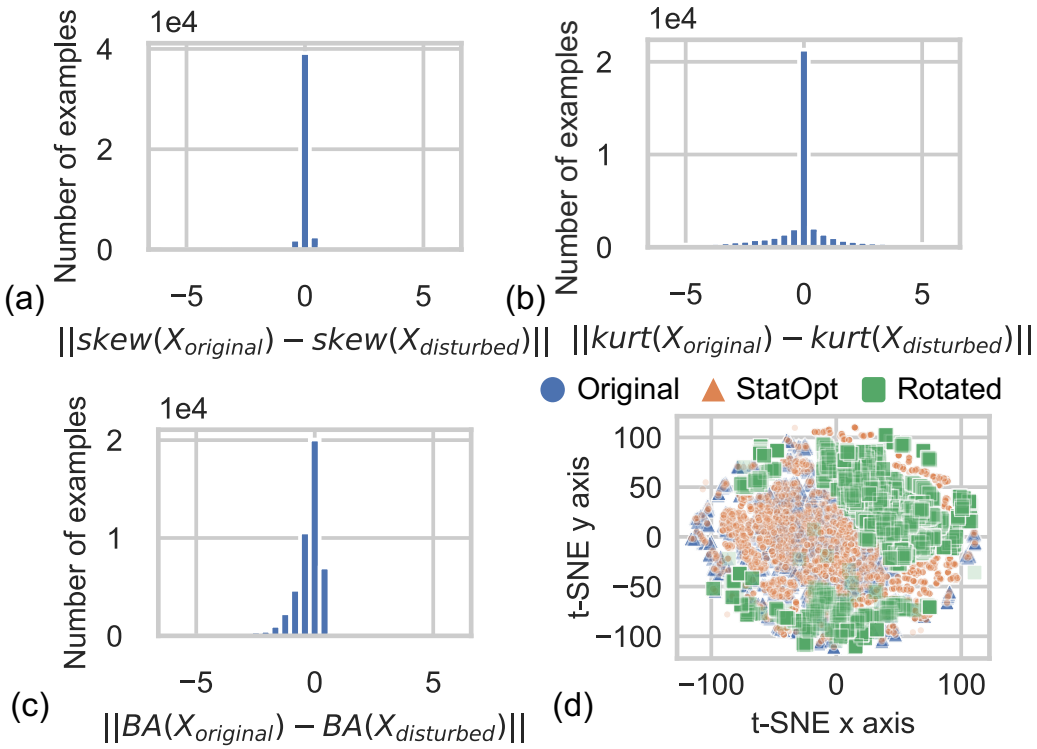


Figure 8.4 Distribution of the difference between the statistical features of the disturbed and the original data using (a) skewness (b) kurtosis, and (c) body acceleration. The zero-centered distribution shows the preservation of the features across the disturbances. (d) Illustration of the t-SNE for the observed sensor data, examples generated by StatOpt, and data with sensor disturbances. The x and y axes represent the coordinates in the reduced 2-D space. The original and StatOpt points overlap in the t-SNE representation shown in (d).

Theoretical Guarantees for Training Data We now empirically evaluate the theoretical bounds of performance for the classifiers trained with StatOpt and compare it with the naive classifiers and the CW method. To this end, Figure 8.5 shows the classification accuracy as a function of different certification bounds E^{lim}. Recall that

the certification bound E^{lim} defines the maximum perturbation in the statistical features that does not lead to a misclassification by the classifier. To sample p and compute E^{lim}, we

Table 8.2 MMD distance (in 10^{-3}) between original, disturbed, and StatOpt generated data distributions describing their pair-wise similarity.

		StatOpt	Disturbed
Original	wHAR	1.17	2.24
	WISDM	0.26	0.46
StatOpt	wHAR	-	0.83
	WISDM	-	0.20

use a random Gaussian noise $\mathcal{N}(\mu_P, C)$ where the diagonal elements of the covariance matrix C are equal to a constant value σ. As the default certification computes $E_{\mu}^{lim} = ||\mu_P||_{max}^2$ over the mean feature, we can compute E^{lim} for other statistical features as follows:

- E^{lim} over Skewness : $|\sum_{j=1}^{T}(X_j - E_{\mu}^{lim})^3|/T\sigma^3)$.
- E^{lim} over Kurtosis: $-3 + |\sum_{j=1}^{T}(X_j - E_{\mu}^{lim})^4|/T\sigma^4$.
- E^{lim} over Body acceleration: $\sqrt{3}E_{\mu}^{lim} - (1/T)$.

The higher the certification bound E^{lim} is, the more accurate the classifier is against larger disturbances. Figure 8.5 shows the value of E^{lim} for representative statistical features S_i to guarantee a given accuracy of the classifier on testing inputs with disturbances p such that $\|S_i(X+p) - S_i(X)\| \leq E_{S_i}^{lim}$. We clearly observe that StatOpt-based reliable classifier provides a higher certification in the mean and body acceleration for both datasets when compared to the naive classifier or the CW method. We note that other statistical features show similar behavior for the certification bounds. In summary, this analysis shows that StatOpt enables more reliable classifiers that are less prone to large real-world perturbations.

8.4.5 Accuracy Analysis of the Reliable Classifier

To evaluate the accuracy with disturbances, we first train the reliable classifier with the complete training data for the w-HAR and WISDM datasets consisting of the observed

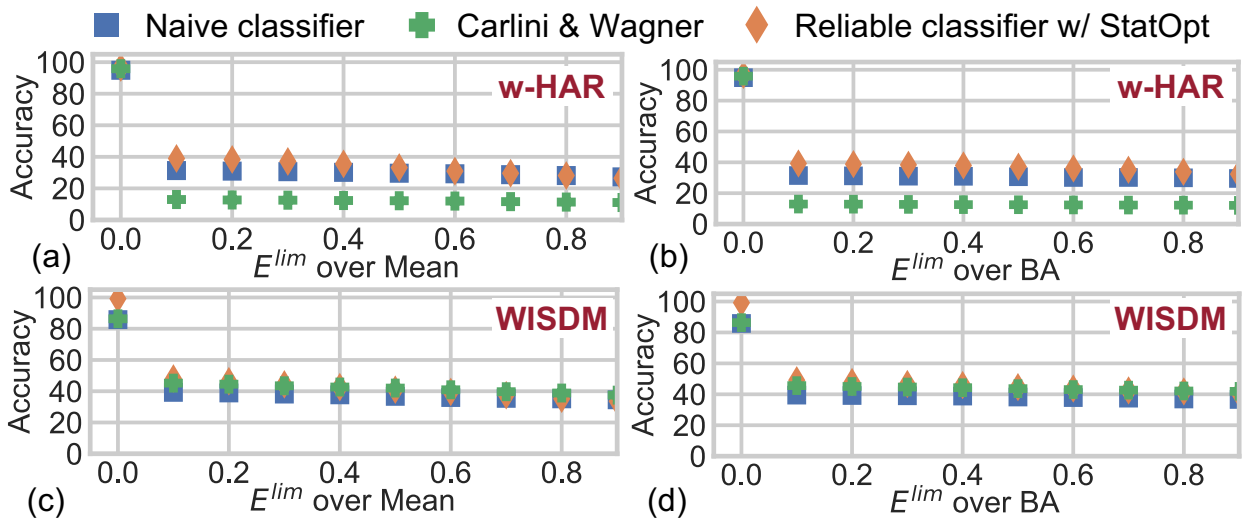


Figure 8.5 Theoretical certification over the mean and body acceleration features for the w-HAR and WISDM datasets.

samples and the data generated by StatOpt. To generate the StatOpt data, we run Algorithm 8 on the training data with $MAX = 10^3$ and $\epsilon = 10^{-2}$. We define the group of possible channels to disturb as different combinations of 1, 2, and all the channels of the time-series sensor data. After training, the sensor data with disturbances is used as input to the HAR classifier to obtain the accuracy. We compare the accuracy of the reliable classifier for both datasets with the standard ML classifier, baseline recovery method, and the CW method in Figure 8.6. We start with the performance of the classifiers under *heading* and *pitch* disturbance for the w-HAR dataset in Figure 8.6(a). As expected, the accuracy of the standard classifier degrades rapidly due to the change in the sensor data distribution. The accuracy improves with the baseline methods, however, the CW method is unable to recover the accuracy. The reliable classifier has accuracy that is equal to or better than the baseline method. We see similar results for the orientation disturbance for the WISDM dataset in Figure 8.6(b).

Next, we analyze the accuracy of the reliable classifier for the position changes and hardware disturbances for the w-HAR dataset in Figures 8.6(c) and (d). Specifically, we reduce the amplitude of the stretch sensor to emulate position change and reduce the frequency of

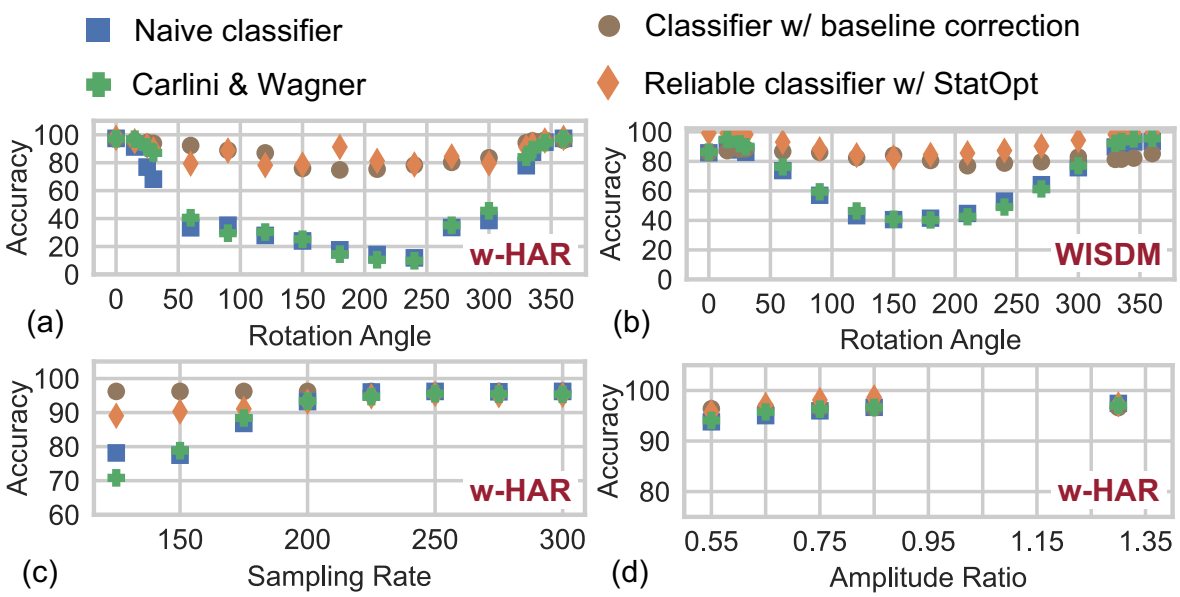


Figure 8.6 Accuracy comparison between the standard classifier, baseline, and StatOpt-enabled reliable classifier for the w-HAR and WISDM datasets.

Table 8.3 Accuracy of non-parametric models against 180° rotation disturbance.

	Decision Tree			Random Forest		
	Naive	Baseline	StatOpt	Naive	Baseline	StatOpt
wHAR	29	72	82	11	83	95
WISDM	33	70	75	36	75	85

the accelerometer to emulate hardware disturbances. The naive classifier is more resilient to these disturbances since the change in the sensor data distribution is lower compared to orientation changes. The reliable classifier effectively recovers the accuracy for both disturbances by using the optimized training data generated by StatOpt. For instance, the reliable classifier improves the accuracy from 78% to 89% when the sampling frequency is 125 Hz instead of 250 Hz. The baseline approach incurs significant overhead to achieve similar accuracy levels. Overall, compared to the standard classifier, the reliable classifier achieves up to 50% improvement over the accuracy performance with zero overhead at runtime.

8.4.6 Generalization beyond Deep Classifiers

One of the important considerations in the training data generated by StatOpt is the applicability to other classifiers, such as decision trees and random forest (James et al., 2013). To validate the applicability of StatOpt to generic classification methods, we use the data generated by StatOpt to train decision tree and random forest classifiers. Table 8.3 shows the HAR accuracy of the classifiers trained with StatOpt along with the naive and baseline classifiers. Due to space constraints, we show the accuracy of a specific rotation disturbance of 180°, which is one of the most challenging disturbances to handle. We observe that the classifiers trained with StatOpt obtain significantly higher classification accuracy when compared to the naive classifier and the baseline. This demonstrates that the StatOpt method provides reliable training examples that can be used by a wide range of classifiers depending on the needs of application.

8.4.7 Implementation Overhead

We implement the proposed reliable classifier and the baseline approach on the TI CC2652R MCU (Texas Instruments Inc., 2018) to characterize the execution time and energy consumption. Table 8.4 summarizes the execution time and energy consumption of each component of the HAR classifiers when using a 1D-CNN. The first two columns show the blocks required to handle each type of disturbance and their execution frequency. The first row shows the common measurement for the classifier used for the activity classification in both approaches. Each activity classification takes about 85.6 ms and 0.94 mJ of energy. Unlike the baseline, StatOpt only requires this amount of energy to perform the prediction as it is an offline approach for improving the reliability of HAR classifier. The next four rows represent the calibration steps needed to identify the degree of disturbances in the baseline approach each time the device is used. The calibration takes more than 6 s to execute and consumes 69.7 mJ energy. In addition to the calibration, the baseline takes about 0.55 mJ of energy to recover

the sampling rate and correct the sensor data for each activity, which in total amounts to 58% of the activity classifier.

Table 8.4 Energy and execution time measurements for StatOpt and baseline.

		Baseline		StatOpt	
Disturbance	Block	Exe. Time (ms)	Energy (mJ)	Exe. Time (ms)	Energy (mJ)
All (1×/activity)	Classifier	85.60	0.94	85.60	0.94
Heading	Walk	3000.00	34.68	-	-
(1×/session)	PCA	24.93	0.29	-	-
Pitch	Stand	3000.00	34.71	-	-
(1×/session)	Gravity detect	1.90	0.02	-	-
All (1×/activity)	Resampling	46.08	0.52	-	-
	Correction	2.31	0.03	-	-

Recall that in addition to the calibration and pre-processing, the baseline approach causes significant inconvenience to the user due to the need to perform a pre-defined set of activities. User convenience is critical because it is one of the primary reasons for users to stop using wearable devices (Ozanne et al., 2018). In contrast, the proposed reliable classifier does not have any overhead. Hence, it will aid in the broader adoption of wearable devices.

8.5 Summary

In this chapter, we proposed StatOpt, a statistical optimization approach that automatically accounts for the real-world sensor disturbances and enables reliable ML classifiers for wearable devices in HAR applications. We have argued how standard classifiers are unable to handle changes that occur to experimental settings (e.g., sensor position) during real-world deployment and the capability of StatOpt to produce a classifier with a reliable performance with low overhead cost. We also presented upper bounds on sensor data disturbances for StatOpt and provided reliability certificates for the ML models. Experiments on different

HAR datasets show that the reliable classifiers generated by StatOpt improve the accuracy up to 50% compared to standard classifiers while incurring zero overhead.

CHAPTER NINE

SEARCH-BASED APPROACH FOR ENERGY-EFFICIENT MISSING DATA RECOVERY IN WEARABLE DEVICES

T. Belkhouja*, D. Hussein*, G. Bhat, and J. Doppa. "Energy-Efficient Missing Data Recovery in Wearable Devices: A Novel Search-based Approach". *Proceedings of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2023. (* denotes equal contribution)

Originally published in the *Proceedings of ACM/IEEE International Symposium on Low Power Electronics and Design.*

Attributions:

T. Belkhouja has contributed to this work by formulating the problem setting from a Machine Learning (ML) perspective, investigating the state of the art related to ML algorithms, formulating the theoretical contribution and the algorithmic solution, implementing the proposed solution as a general algorithm and running the required empirical analysis to highlight its performance improvement compared to the state of the art.

D. Hussein has contributed to this work by formulating the problem setting from the application domain (wearable sensors and mobile health) perspective, investigating the state of the art related to wearable sensors algorithms, formulating the theoretical contribution and the algorithmic solution, implementing the algorithm on microcontrollers and running the required empirical analysis to highlight the performance improvement of the proposed solution compared to the state of the art.

G. Bhat has contributed to this work by investigating the motivation for this work and

the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

J. Doppa has contributed to this work by investigating the motivation for this work and the corresponding state of the art, assisting in the design of the proposed solution and in writing the scientific manuscript.

SEARCH-BASED APPROACH FOR ENERGY-EFFICIENT MISSING DATA RECOVERY IN WEARABLE DEVICES

In this chapter, we propose a second real-world applications[1] for robust time-series deep learning algorithms using wearable sensors. We address the challenge of missing sensors during real-world, runtime usage Sensor data may be missing due to energy limitations, user error, sensor malfunction, or data communication challenges (Shengzhong Liu et al., 2020; Kunze and Lukowicz, 2014). Missing data leads to significant degradation in the overall quality of applications since the underlying ML models are trained with the assumption that data from all sensors is available. Therefore, we propose a novel and energy-efficient search-based *Accuracy-Preserving Imputation (AIM)* approach to produce accuracy-preserving imputations for missing sensor data at runtime.

9.1 Background and Problem Setup

This section first provides the background on wearable devices and introduces the missing sensor data problem.

9.1.1 Wearable Devices Preliminaries

We consider wearable systems with multiple sensors mounted on the body, as shown in Figure 9.1. The sensors are used to monitor physical and physiological parameters for applications including mobile health, which we use as the running example. Since the sensor data is continuous and streaming in nature, wearable devices perform the following steps to enable health assessment.

Data Segmentation: The streaming time series sensor data must be divided into equal sized windows for periodic health assessment and to provide fixed-sized inputs to the ML

[1]This work was developed in close collaboration with Dina Hussein and Ganapti Bhat

models. Given n sensors and T samples in each window, we denote the time-series sensor data with the variable $X \in \mathbb{R}^{n \times T}$.

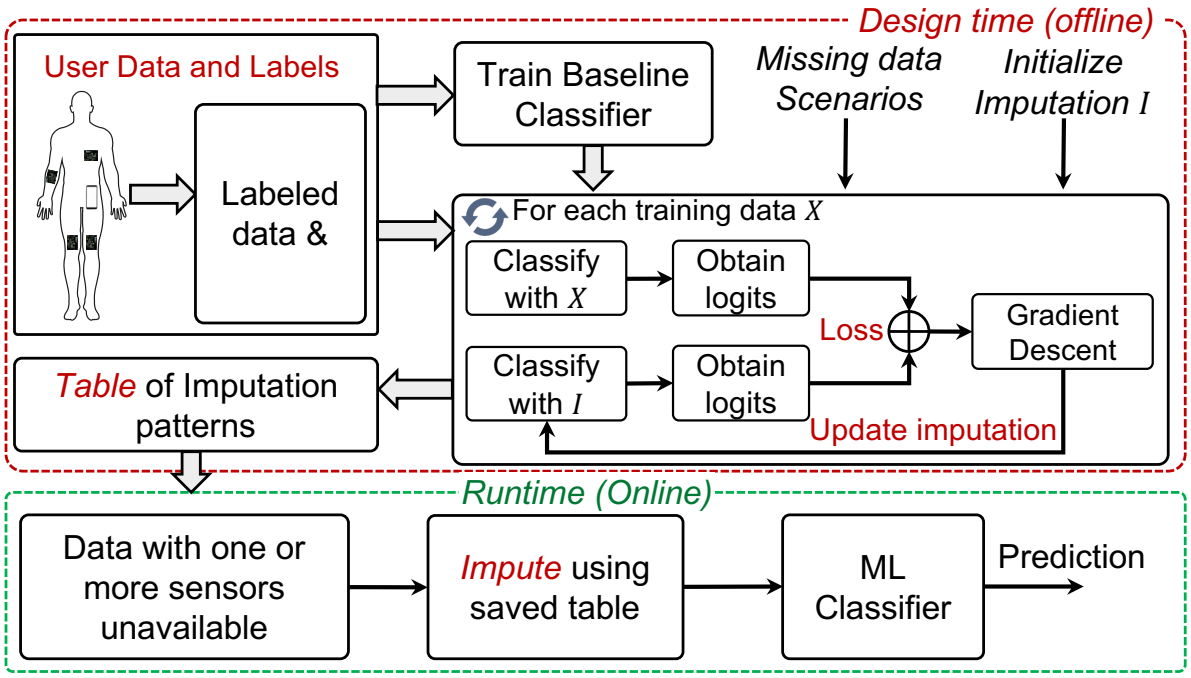


Figure 9.1 Overview of the proposed accuracy-preserving imputation approach.

Feature Generation and Classification: The time-series sensor data in each window $X \in \mathbb{R}^{n \times T}$ are fed into feature generation and classifier blocks to perform the health assessment. Labeled pairs of sensor data $X \in \mathbb{R}^{n \times T}$ and the class label y are used to train a classifier F_θ, where θ are the parameters for the classifier. At runtime, the sensor data and trained classifier are used to predict class labels of interest.

9.1.2 Missing Sensor Data and Imputation Challenges

Depending on the length of unavailability, we can classify the missing data patterns into two main categories as follows.

Random Missing Data: In this case, the sensor encounters isolated missing samples that are not clustered around any particular time instance. Prior work has proposed a number of approaches to handle random missing data (De Waal, Pannekoek, and Scholtus, 2011;

Pires et al., 2020). Random missing data is typically easier to handle since data around the missing instance is available for imputation. Therefore, we do not consider the random missing case and focus our attention on the more *challenging* block missing case described next.

Block Missing Data: Block missing data occurs when long sequences of sensor data are unavailable at runtime. The block missing data is challenging to recover since it does not contain any reference data for the missing sensors. Moreover, in a system with n sensors, we can have $2^n - 2$ possible combinations of block missing data.

9.1.3 Problem Setup

Let $X \in \mathbb{R}^{n \times T}$ be the time-series sensor data, where n is the total number of sensor channels and T is the number of samples in each input window. We denote every input $X \in \mathbb{R}^{n \times T}$ as $[X_1, \cdots, X_i, \cdots, X_n]$ where $X_i \in \mathbb{R}^T$ corresponds to a channel i of X. The class label for each window is denoted by y. We denote the set of training examples $\mathcal{D}$ as several input X and ground-truth output y pairs. Standard ML algorithms use the given data $\mathcal{D}$ to train a classifier F_θ which takes time-series X as input to predict the corresponding class label $\hat{y}$, where θ stands for parameters of the classifier.

During inference, the data from one or more sensors might be missing. Let $\{j\}_{0 \leq j \leq n}$ represent the subset of channels that are missing at runtime for a given input X. We denote the new input that has $\{j\}$ missing channels by $\tilde{X}_{\{j\}} \in \mathbb{R}^{n \times T}$:

$$\tilde{X}_{\{j\}} = \begin{cases} 0^T & \text{if } i \in \{j\} \\ X_i & \text{if } i \notin \{j\} \end{cases} \tag{9.1}$$

For example, for a human activity recognition (HAR) application, when one of the sensors goes missing, its three accelerometer channels data $\{j\} = \{1, 2, 3\}$ will have 0 value.

This missing data in the input results in misclassifications ($\hat{y} \neq y$) by the classifier F_θ. Consequently, the overall performance of the classifier and quality of application service

during deployment will degrade significantly.

9.2 HAR Related Work

Wearable devices are being increasingly used in health applications (Espay et al., 2016; Mosenia et al., 2017; Limaye and Adegbija, 2018; Bhat et al., 2020). Integrating multiple sensors increases the likelihood of one or more sensors being unavailable due to energy constraints, user error, or communication challenges. Therefore, there is a strong need for energy-efficient and accuracy-preserving methods to recover missing data.

Recent work has proposed several methods to handle missing data in sensor based applications (De Waal, Pannekoek, and Scholtus, 2011; Guo et al., 2019; Yoon, Jordon, and Schaar, 2018; Pires et al., 2020; Hussein, Jain, and Bhat, 2022). These approaches typically use statistical or deep generative methods to recover the missing data. Statistical methods, such as mean, median, or regression use available data around the missing instances to obtain an imputation (Pires et al., 2020; De Waal, Pannekoek, and Scholtus, 2011). As such, they are suitable to handle isolated missing data instances where data around the missing samples is available. However, they are not suitable for long sequences of sensor unavailability with no reference data for statistical methods, which is the focus of this chapter. Deep generative methods have been recently proposed to handle longer sequences of missing data (Yoon, Jordon, and Schaar, 2018; Talukder et al., 2022). For instance, (Yoon, Jordon, and Schaar, 2018) employs a generative adversarial imputation network (GAIN) to impute the data. Specifically, the GAIN approach imputes the missing data conditioned upon observed data. Similarly, the work in (Talukder et al., 2022) employs deep auto-encoder models to impute EEG recordings from multiple patients and data collection days. However, the primary limitation of deep learning methods is the high memory overhead and the energy cost of performing imputation at runtime.

To precisely fill this gap in the current knowledge, we develop a novel and energy-efficient

accuracy-preserving imputation method to handle missing sensor data at runtime.

9.3 Search based Accuracy-Preserving Imputation

This section describes the accuracy-preserving imputation (AIM) algorithm to solve missing sensor data challenge for wearable applications. We first provide an overview of the AIM approach and list the two synergistic design principles behind AIM. Next, we describe the details of the algorithmic approaches which instantiate those two design principles.

Overview of AIM Approach: During the offline configuration of AIM, we perform the following two steps sequentially, as shown in Figure 9.1. First, we train a robust ML classifier F_θ that can make accurate predictions for small perturbations of time-series signals in the training data (i.e., no missingness). This step ensures accuracy even if there are small errors in the imputed values. Second, given the trained ML classifier F_θ, for each candidate missing sensors configuration, we execute a search algorithm to compute the most likely imputation pattern that preserves the accuracy of the classifier F_θ using the training data. The output of this step is a lookup table $\mathcal{I}$ that stores one imputation pattern for each missing configuration. At runtime, given an input X with some missing channels (one for each sensor), we impute the missing data using appropriate imputation pattern from the lookup table $\mathcal{I}$ (i.e., negligible overhead) and then use the classifier F_θ to make prediction.

Design Principles: AIM meets the desiderata of an effective solution for missing sensor data based on two synergistic principles. *1) Accuracy-preserving imputation:* there is no need to recover the exact missing sensor data as long as the accuracy of ML classifier is preserved. *2) Training robust classifiers:* training the ML classifier to make accurate predictions even with small deviations from the exact sensor data distribution will exhibit robustness to small errors in imputed data. Below we provide algorithms to instantiate these two principles.

9.3.1 Search Algorithm for Accuracy-Preserving Imputation

Intuition: During the offline training phase with no missing data, the ML classifier for wearable application achieves high accuracy on the given classification task. Since the inputs are multivariate time-series signals, the classifier relies on the information spread across n different channels to make accurate predictions. Prior work has shown that classifiers rely on a subset of critical channels to predict output labels (Belkhouja and Doppa, 2020b). This means that in case of missing channels, the input possesses enough information to allow the classifier to predict its true label. Therefore, we set our goal to find a recovery data pattern that lets the classifier predict the correct labels from the available channels. The recovery pattern pushes the classifier to predict the output label as if the data from all sensors is available. In summary, we can view our solution as the search for the most likely data pattern to impute the data for missing sensors for preserving the accuracy of the given ML classifier.

Algorithm: Formally, given a ML classifier F_θ, for any input $X = [X_1, \cdots, X_n]$ and a fixed set of missing channels $\{j\}$, we search for an imputation pattern $\mathcal{I}_{j\in\{j\}} \in \mathbb{R}^T$ s.t.:

$$\mathcal{I}_{\{j\}} = \begin{cases} \mathcal{I}_j & \text{if } i \in \{j\} \quad \text{and} \quad F_\theta(X) \approx F_\theta(\mathcal{I}_{\{j\}}) \\ X_i & \text{if } i \notin \{j\} \end{cases} \tag{9.2}$$

We note that $\mathcal{I}_{j\in\{j\}}$ *does not depend* on the available sensor data of the input X. Every imputation pattern is stored in a look up table indexed by the combination of the missing channels $\{j\}_{0\leq j\leq n}$ where the samples are missing. We conduct this search for each missing sensor data configuration during design time (offline) where the goal is to find for every combination of $\{j\}$ missing channels, the corresponding imputation pattern $\mathcal{I}_{j\in\{j\}}$ that yields a prediction similar to the prediction on the original input without any missingness. Hence, at runtime, we use the stored lookup table to select the appropriate imputation pattern with negligible overhead.

We find imputation patterns based on a given set of missing channels $\{j\}_{0\leq j\leq n}$ (missing configuration) that preserves the accuracy of classifier F_θ. We define our overall objective for the search of imputation pattern as shown below:

$$\text{Given } \{j\}: \text{ We find } \mathcal{I}_{\{j\}} \text{ s.t. } \forall X, F_\theta(X) \approx F_\theta(\mathcal{I}_{\{j\}}) \tag{9.3}$$

We compute the imputation pattern to fill values of the missing channels by solving the minimization problem below:

$$\min_{\mathcal{I}_{j\in\{j\}}} \mathcal{L}\Big(\text{Logits}(F_\theta(X)), \text{Logits}(F_\theta(\mathcal{I}_{\{j\}}))\Big) \tag{9.4}$$

The loss function $\mathcal{L}$ over the logits outcome of the classifier is used for accuracy-preserving imputation pattern search. The logits of a classifier are interpreted as the unnormalized predictions for each candidate class label and input time-series signal pair. The role of this loss function is to compute the similarity between the prediction outcomes of the classifier F_θ for the original input example X (no missingness) and the input with the imputed pattern $\mathcal{I}_{\{j\}}$. For example, $\mathcal{L}$ can be the Mean Squared Error between the logits values of both predictions $F_\theta(X)$ and $F_\theta(\mathcal{I}_{\{j\}})$. *When $\mathcal{L} \to 0$, accuracy of the classifier over imputed and original inputs will be similar.*

Algorithm 9 shows the pseudo-code of proposed search approach to compute accuracy-preserving imputation pattern $\mathcal{I}_{\{j\}}$. We set $MAX_G = 100$ to ensure that AIM can find the optimal imputation pattern per example X. Additionally, we set $MAX = 50$ to ensure that AIM optimizes the imputation pattern across the training data. The output of this algorithm is used to populate a look-up table $\mathcal{I}$ that maps the set of missing channels $\{j\}$ to the corresponding imputation pattern $\{\mathcal{I}_j\}$. Therefore, given a missing sensor data configuration $\{j\}$ and any input X at test time, we construct $\mathcal{I}_{\{j\}}$ as shown in Eq. 9.2 and employ the classifier F_θ to predict the class label after using $\mathcal{I}_{\{j\}}$ to impute the missing data in input X.

Algorithm 9 AIM Search for accuracy-preserving imputation

Input: Training set $\mathcal{D} = \{X\}$; F_θ, pre-trained classifier on $\mathcal{D} = \{(X, y)\}$; $\{j\}$, missing sensors configuration; MAX_G, maximum iterations for gradient descent; MAX, maximum iterations over all inputs

Output: $\{\mathcal{I}_{j\in\{j\}}\}$, imputation pattern.

Random initialization of the set $\{\mathcal{I}_{j\in\{j\}}\}$
for i=$1, \cdots, MAX$ **do**
  for each training example X **do**
    for i_G=$1, \cdots, MAX_G$ **do**
      Compute the classifier's logits values: $\text{lg}X = \text{Logits}(F_\theta(X))$
      Compute the classifier's logits values: $\text{lg}I = \text{Logits}(F_\theta(\mathcal{I}_{\{j\}}))$
      Estimate the loss $\mathcal{L}(\text{lg}X, \text{lg}I)$
      Estimate the gradient $\nabla_{\{\mathcal{I}_j\}}\mathcal{L}$ for $j \in \{j\}$
      Perform gradient descent and update $\{\mathcal{I}_j\}$ for $j \in \{j\}$
    end for
  end for
end for
return imputation pattern $\{\mathcal{I}_{j\in\{j\}}\}$ as per the requirements of Eq.9.2

9.3.2 Training Robust Classifiers for Improved Effectiveness

Recall that we store *one* imputation pattern for each missingness configuration to preserve the accuracy and AIM's imputation strategy does not depend on the available sensor data of the given input X. As a result, AIM has negligible overhead, but ML classifier may not make correct predictions with generic imputation patterns for a small fraction of the input examples. Therefore, we propose to train ML classifiers to be robust to small errors in the imputed data.

The motivation to train robust ML classifiers is two-fold. First, the ML classifier is less sensitive to the natural noise in the training data. As a result, we will be able to find more robust imputation patterns using our search algorithm and robust ML classifier. Second, the ML classifier will be more robust to small errors in the imputed data at the runtime.

We train robust a ML classifier using data augmentation and propose to apply the recent framework of generating augmented data for time-series signals based on their statistical features (Belkhouja and Doppa, 2022). The key idea is to generate small perturbations over the original time-series signals in the training data which preserve the statistical features. To instantiate this framework for our specific use-case, we employ the following statistical features: mean absolute error, statistical average, and root mean square. Additionally, for HAR applications, we include the body acceleration feature due to its high relevance.

9.4 Experiments and Results

This section analyzes the performance of the proposed data recovery approach on four datasets along different dimensions.

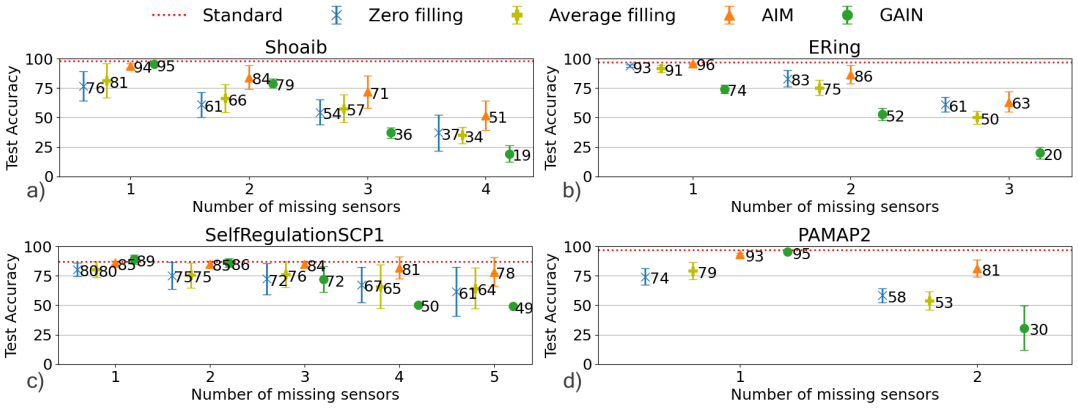


Figure 9.2 Accuracy (Mean and standard deviation) of the robust-trained ML classifier via different imputation methods on all combinations of missing sensors.

9.4.1 Experimental Setup

Wearable Device Setup We employ the Odroid-XU3 board (Hardkernel, 2014) for sensor data processing, while noting that any low-power processor can be used. Odroid-XU3 contains four high-performance ARM Cortex-A15 and four low-power Cortex-A7 cores. We use the Odroid-XU3 to store the imputation table and measure the overhead on A7 cores.

Datasets AIM is validated using four datasets described below. To validate the proposed missing data recovery approach, we vary the number of missing sensors for each dataset with n sensors from one to $n-1$.

- **Shoaib et al. (Shoaib et al., 2014):** The Shoaib dataset includes three-axis accelerometer data for 10 users performing seven activities. The dataset has accelerometer sensors at five locations on the body: left pocket, right pocket, wrist, belt and upper arm.

- **PAMAP2 (Reiss and Stricker, 2012b)**: PAMAP2 is a HAR dataset that provides data from three accelerometers for five activities with nine users.

- **eRing (Wilhelm et al., 2015)**: eRing is a smart health dataset that uses a ring to capture data along four dimensions. The eRing dataset allows us to test the efficacy of AIM in gesture recognition settings.

- **SelfRegulationSCP1 (SR-SCP1) (Birbaumer et al., 2001)**: SR-SCP1 is a health monitoring dataset that includes EEG data from six channels. The data from EEG sensors is used to develop a control system to drive spelling devices for completely paralyzed patients.

Evaluation Metrics We employ accuracy, memory, and energy consumption as evaluation metrics. Accuracy is used as a metric because accuracy is of utmost importance in health applications. Similarly, memory and energy are important for wearable devices due to resource constraints.

Classifier Representation We use a 1-D convolutional neural network (CNN) as the classifier for all datasets. Specifically, we use a 1-D CNN with one conv. and max-pooling layers, and two fully connected layers with the ReLU activation and dropout value of 20%. We use the Adam optimizer (Kingma and Ba, 2015) over 20 epochs for both standard and robust training.

9.4.2 Baseline Methods for Comparison

The proposed data recovery approach is compared against baseline approaches described below.

GAIN (Yoon, Jordon, and Schaar, 2018): GAIN is a generative approach that recovers missing data as a function of the observed data. GAIN trains a deep neural network that takes the observed data and a mask specifying the missing time instances as input. The output of the generator is a data matrix that consists of imputed values. One of the disadvantages of the GAIN approach is the high memory requirement for storage of the generator parameters and energy overhead for each imputation. Moreover, generative models for time-series data are challenging to train (Brophy et al., 2021), which can affect their accuracy in complex tasks.

Zero Filling: In the absence of any data recovery algorithm, missing data will typically be filled with zeros. Therefore, we use it as one of the baselines for comparison. Filling missing values with previously observed data is not feasible since we assume the sensor is missing for the entire experiment.

Average Filling: Another realistic alternative for zero-filling is to fill the missing data with pre-determined values that represent the average case over the training data with no missingness. These values are determined by averaging sensor data across all training time-series signals.

9.4.3 Application Accuracy with Imputed Data

We start the experimental evaluation by analyzing the accuracy of the health applications under different missing data scenarios. For each dataset, we first train a classifier to perform the application tasks. Once the classifier is trained, we use it with the proposed search algorithm to find likely patterns of sensor data when one or more sensors are missing.

Figure 9.2 shows the comparison of accuracy for all four datasets. Each point on the

figure shows the mean and standard deviation of the accuracy over all possible combinations of missing sensors. For example, in case of two missing sensors in the Shoaib dataset, we obtain the average and standard deviation over $\binom{5}{2}$ combinations of possible scenarios. We see that missing data with zero-filling or average-filling settings have a significant drop in accuracy. For example, for both HAR datasets, a single missing sensor results in more than 20% drop in average accuracy. In contrast, using the same classifier and missing data cases, AIM is able to efficiently recover the classification performance. Even when data from the entire window is missing, AIM is able to produce an average performance within 5% of the original accuracy for all datasets. AIM also succeeds in improving average performance of the classifiers on HAR datasets by 15% in the highly-unlikely case where almost all sensors are missing. Additionally, the confusion matrices in Figure 9.3 show that AIM improves the classification accuracy over different classes with equal importance in spite using a single imputation pattern.

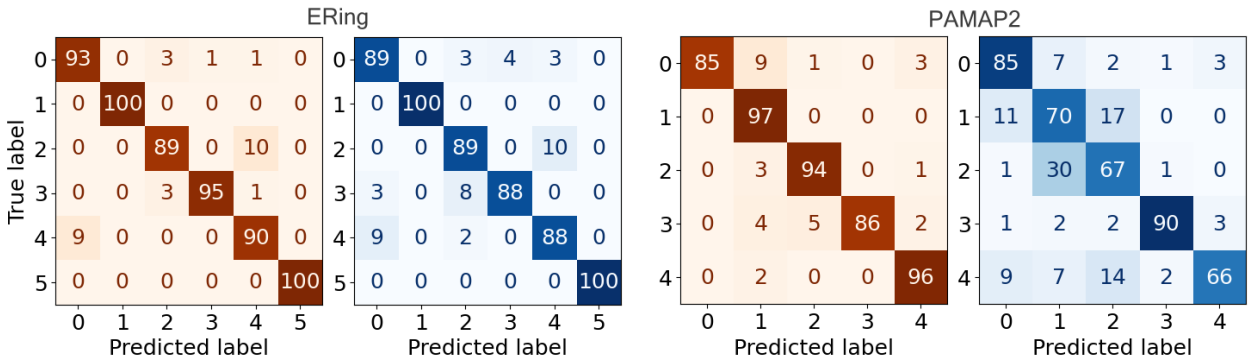


Figure 9.3 Confusion matrix normalized over the true labels of the deployed classifier on ERing and PAMAP2 datasets using AIM (red) imputation methods in the event of a single missing sensor. The same performance using zero-filling (blue) is provided for reference.

Compared to the imputation provided by the baseline GAIN approach, AIM produces better results for most of the cases. Notably, GAIN fails to recover the original accuracy when more than one sensor is missing. GAIN has lower accuracy than zero-filling in some cases because GAIN is unable to follow real data accurately and incurs higher error. For

PAMAP2, the average performance is reduced from 58% to 30% for the GAIN algorithm. In summary, the AIM approach is able to efficiently recover the data with low overhead, while the baseline GAIN approach is unable to recover the accuracy for more than one sensor missing and has a higher overhead.

9.4.4 Accuracy Improvement with Robust Classifiers

The AIM algorithm generates a pattern $\mathcal{I}_{\{j\}}$ to preserve the accuracy of the classifier in the case of $\{j\}$ missing input channels. Ideally, the generated pattern requires small adjustments to fit every input X to maintain $F_\theta(X) \approx F_\theta(\mathcal{I}_{\{j\}})$. To account for these adjustments, we use robust training for the ML classifier to overcome small errors in imputed data. Robust classifiers are also important because any health application must be able to handle small variations in data either due to natural disturbances or imputation. To this end, we compare the accuracy of the proposed robust classifiers with the standard classifier in Table 9.1. Indeed, the table shows that robust training overcomes errors due to small imputation deviations by improving the average accuracy for majority of cases while reducing standard deviation. For example, SR-SCP1 has an increase of 6% in recovered accuracy with a standard deviation of 1%. Overall, robust training is able to provide higher accuracy while reducing standard deviation.

9.4.5 Implementation Overhead

One of the primary advantages of AIM is low memory and energy overhead. Table 9.2 shows the memory overhead for AIM and GAIN, respectively. GAIN incurs high memory overhead to store the parameters of the generator network. In contrast, AIM has less than 1 MB memory overhead. AIM memory requirements are minimal even when the wearable device includes multiple health applications. The memory overhead for AIM can be further reduced by loading *only* the required imputation setting on detecting missing data.

Next, Figure 9.4(a) compares energy consumption of GAIN and AIM for all datasets.

Table 9.1 Classification accuracy of the imputed data of k missing sensors generated by AIM using different standard and robust training protocols. Table entries show mean with standard deviation in parantheses.

Dataset	k	Standard	Robust	Dataset	k	Standard	Robust
Shoaib	1	91 (2)	94 (1)	SR-SCP1	1	79 (8)	85 (1)
	2	80 (5)	84 (9)		2	79 (8)	85 (2)
	3	69 (13)	71 (13)		3	79 (8)	84 (2)
	4	50 (12)	51 (12)		4	81 (9)	81 (9)
ERing	1	95 (1)	96 (0)		5	73 (5)	78 (12)
	2	84 (8)	86 (7)	PAMAP2	1	92 (2)	93 (0)
	3	60 (2)	63 (8)		2	75 (3)	77 (11)

The energy consumption is obtained using power sensors on the Odroid-XU3 board. We see that AIM consumes less than 10 mJ per imputation while GAIN has significantly higher energy consumption. For instance, energy consumption for the SR-SCP1 dataset is close to 1 J for each imputation. Similarly, Figure 9.4(b) shows percentage energy savings achieved by AIM when compared to GAIN. The energy savings are close to 98% for all datasets except eRing. The eRing dataset has lower energy savings of about 74% since it has lower computation requirements for both GAIN and AIM, resulting in lower energy savings. In summary, the AIM approach provides superior performance over prior approaches while incurring significantly lower overhead.

9.5 Summary

In this chapter, we have presented a novel search-based algorithm (AIM) that obtains most likely imputation patterns of sensor data for different missing data scenario via offline analytics. AIM overcomes the challenges of missing data in the input space that may be due

Table 9.2 Summary of memory overhead of AIM and GAIN approaches

Dataset	AIM Memory (MB)	GAIN Memory (MB)
Shoaib	0.180	25
PAMAP2	0.055	60
eRing	0.007	0.19
SR-SCP1	0.667	81

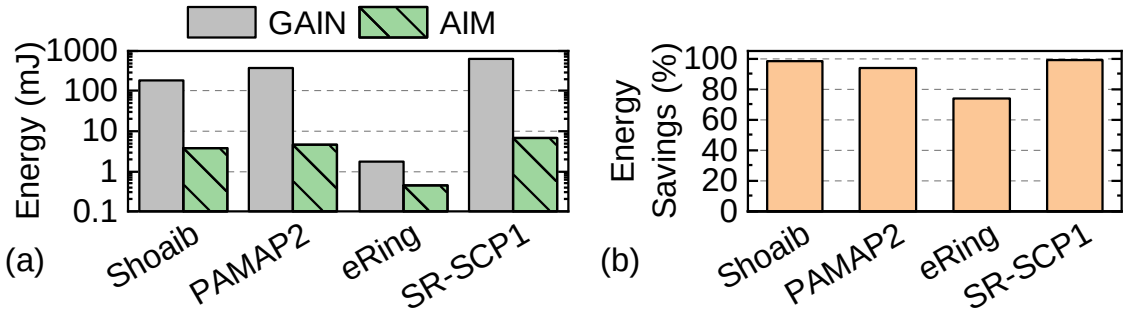


Figure 9.4 a) Comparison of energy consumption for GAIN and AIM approach. The y-axis is shown in log scale to represent the large range of values. b) Energy savings achieved by AIM when compared to GAIN.

to a loss in the quality of service. The key idea is to run an offline search for the most likely pattern that has the potential to preserve the accuracy of the classifier during runtime. Experiments on diverse wearable sensor based time-series benchmarks showed that the proposed approach is able to maintain accuracy within 5% of the ideal accuracy when the number of missing sensors is less than two, with negligible runtime overhead.

CHAPTER TEN

CONCLUSION

In this chapter, we summarize the main contributions of this dissertation, lessons learned, and list some promising future research directions.

10.1 Summary of Dissertation Contributions

In this dissertation, we have addressed several challenges related to the robustness of machine learning algorithms for time-series data.

1. We provided a preliminary analysis using a principled framework to analyze deep models for multivariate time-series classification in adversarial settings. Our comprehensive study showed that these deep learning methods are significantly vulnerable to adversarial attacks during real-world deployment.

2. We proposed two different approaches, namely TSA-STAT and DTW-AR, to create more effective adversarial examples for the time-series domain. Both frameworks create effective adversarial examples by overcoming the limitations of prior methods based on Euclidean distance and using novel designs suited for the time-series data space. We theoretically and empirically demonstrate their effectiveness in fooling deep models for time-series data and in improving their robustness.

3. We derived a novel theoretically-certified bound for adversarial robustness based on the TSA-STAT framework that applies to any deep model for the time-series domain.

4. We proposed a novel algorithm to train robust deep neural networks for time-series domain (RO-TS). The training problem was formulated as a min-max optimization problem to reason about the worst-case risk in a small neighborhood defined by time-series-similarity-based distances. We developed the theoretically-sound stochastic com-

positional alternating gradient descent and ascent (SCAGDA) algorithm that carefully leverages the structure of the optimization problem to solve it efficiently.

5. We have addressed the Out-of-Distribution (OOD) challenge for time-series data by introducing a novel seasonal ratio (SR) score to detect OOD examples in the time-series domain. SR scoring relies on Seasonal and Trend decomposition using Loess (STL) to extract class-wise semantic patterns and remainders from time-series signals, and estimating class-wise conditional likelihoods for both input time-series and remainders using deep generative models.

6. We successfully applied the developed robust machine learning algorithms for time-series for wearable sensors enabled mobile applications. StatOpt is a statistical optimization approach that automatically accounts for the real-world sensor disturbances and enables reliable ML classifiers. We have also presented a novel search-based algorithm (AIM) that obtains the most likely imputation patterns of sensor data to overcome missing data scenarios via offline analytics.

10.2 Lessons Learned

In this section, we describe the most important lessons we have learned from this work.

1. While the l_p-norm based distance measure was very effective in other domains such as computer vision and NLP, adopting existing algorithms to the time-series domain is ineffective. To capture the real similarity between examples and avoid the accuracy drop of the classifier during training, time-series data require the use of a specialized distance measure that handles their unique characteristics.

2. There is a wide variety of similarity measures that can be used for time-series data. We have found in our work that most of them require high computational overhead (e.g., elastic measures such as DTW and GAK are quadratic in complexity) which makes

them hard to adopt in any time-series framework. Therefore, we need to optimize these measures efficiently to be used in specific time-series problem settings such as adversarial frameworks.

3. The interpretability of time-series ML models is essential to designing reliable frameworks due to the ambiguity of evaluation methods. Interpretability is a significant challenge for time-series data, unlike other data modalities where human experts can manually validate different performances.

10.3 Future Research Directions

Inspired by the work developed in this dissertation, some important future directions include:

- The problem space of robustness for time-series ML is relatively new. An immediate future direction is the investigation of additional similarity measures that can enhance the performance reliability of deep learning methods using adversarial training.

- Due to the ambiguity of time-series data and the cost of interpretability, there is a need to investigate reliable evaluation methods for time-series frameworks beyond the classical evaluation procedure that heavily relies on collected data.

- An important direction to investigate is the performance of the time-series OOD detection framework to validate generative algorithms for synthetic time-series data.

- As seen in human activity recognition settings, other domain applications such as finance and monitoring systems require efficient adoption of time-series robust frameworks where the domain characteristics are employed during the design of reliable algorithms.

REFERENCES

Abadi, Martín et al. (2016). "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *CoRR* abs/1603.04467.

Athalye, Anish, Nicholas Carlini, and David Wagner (2018). "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples". In: *arXiv preprint arXiv:1802.00420*. Proceedings of Machine Learning Research 80. Ed. by Jennifer Dy and Andreas Krause, pp. 274–283.

Athalye, Anish, Logan Engstrom, et al. (2018). "Synthesizing Robust Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning (ICML*.

Bagnall, Anthony et al. (2020). *The UEA & UCR Time Series Classification Rep.* www.timeseriesclassification.com. Accessed: 2021-08-02.

Bai, S., J Z. Kolter, and V. Koltun (2018). "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". In: *arXiv preprint arXiv:1803.01271*.

Baluja, Shumeet and Ian Fischer (2018). "Learning to Attack: Adversarial Transformation Networks". In: *AAAI Conference on Artificial Intelligence*.

Barshan, Billur and Aras Yurtman (2020). "Classifying Daily and Sports Activities Invariantly to the Positioning of Wearable Motion Sensor Units". In: *IEEE Internet of Things J.* 7.6, pp. 4801–4815.

Belkhouja, Taha and Janardhan Rao Doppa (2020a). "Analyzing Deep Learning for Time-Series Data Through Adversarial Lens in Mobile and IoT Applications". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

— (2020b). "Analyzing Deep Learning for Time-Series Data Through Adversarial Lens in Mobile and IoT Applications". In: *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 39.11, pp. 3190–3201.

— (2022). "Adversarial Framework with Certified Robustness for Time-Series Domain via Statistical Features". In: *Journal of Artificial Intelligence Research (JAIR)* 73, pp. 1435–1471.

— (2023). "Adversarial Framework with Certified Robustness for Time-Series Domain via Statistical Features (Extended Abstract)". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*. Journal Track, pp. 6845–6850.

Belkhouja, Taha, Yan Yan, and Janardhan Rao Doppa (2022). "Training Robust Deep Models for Time-Series Domain: Novel Algorithms and Theoretical Analysis". In: *Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, pp. 6055–6063.

Belkhouja, Taha, Yan Yan, and Janardhan Rao Doppa (2023a). "Dynamic Time Warping based Adversarial Framework for Time-Series Domain". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

— (2023b). "Out-of-Distribution Detection in Time-Series Domain: A Novel Seasonal Ratio Scoring Approach". In: *ACM Trans. Intell. Syst. Technol.* 15.1. ISSN: 2157-6904.

Berndt, Donald J and James Clifford (1994). "Using dynamic time warping to find patterns in time series." In: *KDD workshop*. Vol. 10. 16. Seattle, WA, USA: pp. 359–370.

Bhat, Ganapati et al. (2020). "w-HAR: An Activity Recognition Dataset and Framework using Low-Power Wearable Devices". In: *Sensors* 20.18, p. 5356.

Birbaumer, N et al. (2001). "A Brain-Controlled Spelling Device for the Completely Paralyzed". In: *Nature*, pp. 297–298.

Blázquez-García, Ane et al. (2021). "A review on outlier/anomaly detection in time series data". In: *ACM Computing Surveys (CSUR)*.

Braei, Mohammad and Sebastian Wagner (2020). "Anomaly detection in univariate time-series: A survey on the state-of-the-art". In: *arXiv preprint arXiv:2004.00433*.

Brendel, Wieland, Jonas Rauber, and Matthias Bethge (2018). "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models". In: *International Conference on Learning Representations (ICLR)*.

Brockwell, Peter J and Richard A Davis (2016). *Introduction to time series and forecasting*. springer.

Brophy, Eoin et al. (2021). "Generative Adversarial Networks in Time Series: A Survey and Taxonomy". In: *arXiv preprint arXiv:2107.11098*.

Buja, Andreas et al. (2008). "Data visualization with multidimensional scaling". In: *Journal of computational and graphical statistics* 17.2, pp. 444–472.

Canizo, Mikel et al. (2019). "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study". In: *Neurocomputing*.

Cao, Longbing (2022). "Ai in finance: challenges, techniques, and opportunities". In: *ACM Computing Surveys (CSUR)* 55.3, pp. 1–38.

Cao, Tianshi et al. (2020). "A benchmark of medical out of distribution detection". In: *arXiv preprint arXiv:2007.04250*.

Carlini, Nicholas and David Wagner (2017). "Towards evaluating the robustness of neural networks". In: *IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, pp. 39–57.

Challu, Cristian et al. (2022). "Deep Generative model with Hierarchical Latent Factors for Time Series Anomaly Detection". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (2010). "Anomaly detection for discrete sequences: A survey". In: *IEEE transactions on knowledge and data engineering* 24.5, pp. 823–839.

Chen, Guilin et al. (2018). "Latent feature learning for activity recognition using simple sensors in smart homes". In: *Multimedia Tools and Applications* 77, pp. 15201–15219.

Chen, Jinghui and Quanquan Gu (2020). "Rays: A ray searching method for hard-label adversarial attack". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Chen, Jinghui, Dongruo Zhou, et al. (2020). "A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chen, T., Y. Sun, and W. Yin (2020). "Solving stochastic compositional optimization is nearly as easy as solving stochastic optimization". In: *arXiv preprint arXiv:2008.10847*.

Christ, Maximilian, Andreas W Kempa-Liehr, and Michael Feindt (2016). "Distributed and parallel time series feature extraction for industrial big data applications". In: *arXiv preprint arXiv:1610.07717*.

Cleveland, Robert B et al. (1990). "STL: A seasonal-trend decomposition". In: *J. Off. Stat* 6.1, pp. 3–73.

Cohen, Jeremy M, Elan Rosenfeld, and J Zico Kolter (2019). "Certified adversarial robustness via randomized smoothing". In: *arXiv preprint arXiv:1902.02918*, pp. 1310–1320.

Cullinane, Michael J (2011). "Metric axioms and distance". In: *The Mathematical Gazette* 95.534, pp. 414–419.

Cuturi, M. (2011). "Fast global alignment kernels". In: *ICML*.

Cuturi, M. et al. (2007). "A kernel for time series based on global alignments". In: *ICASSP*.

Cuturi, Marco and Mathieu Blondel (2017). "Soft-dtw: a differentiable loss function for time-series". In: *International Conference on Machine Learning*. PMLR, pp. 894–903.

Dan Hendrycks, Kevin Gimpel (2017). "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *5th International Conference on Learning Representations, ICLR*.

Dau, Hoang Anh, Anthony Bagnall, et al. (2019). "The UCR time series archive". In: *IEEE - CAA Journal of Automatica Sinica*.

Dau, Hoang Anh, Diego Furtado Silva, et al. (2018). "Optimizing dynamic time warping's window width for time series data mining applications". In: *Data mining and knowledge discovery*.

De Waal, Ton, Jeroen Pannekoek, and Sander Scholtus (2011). *Handbook of Statistical Data Editing and Imputation*. Vol. 563. John Wiley & Sons.

Dempsey, Paul (2015). "The Teardown: Apple Watch". In: *Engg. & Tech.* 10.6, pp. 88–89.

Dietterich, Thomas G (2017). "Steps Toward Robust Artificial Intelligence". In: *AI Magazine* 38.3, pp. 3–24.

Dodge, Samuel and Lina Karam (2017). "A study and comparison of human and deep learning recognition performance under visual distortions". In: *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE, pp. 1–7.

Doersch, Carl (2016). "Tutorial on variational autoencoders". In: *arXiv preprint arXiv: 1606.05908*.

Drensky, Vesselin and Ralf Holtkamp (2006). "Constants of formal derivatives of non-associative algebras, Taylor expansions and applications". In: *Rendiconti del Circolo Matematico di Palermo* 55.3, pp. 369–384.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml.

Espay, Alberto J et al. (2016). "Technology in Parkinson's Disease: Challenges and Opportunities". In: *Movt. Disorders* 31.9, pp. 1272–1282.

Fawaz, H Ismail et al. (2019). "Adversarial attacks on deep neural networks for time series classification". In: *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

Fawaz, Hassan Ismail et al. (2018). "Data augmentation using synthetic data for time series classification with deep residual networks". In: *arXiv preprint arXiv:1808.02455*.

Fischetti, Matteo and Jason Jo (2018). "Deep neural networks and mixed integer linear optimization". In: *Constraints* 23.3, pp. 296–309.

Fulcher, Ben D and Nick S Jones (2014). "Highly comparative feature-based time-series classification". In: *IEEE Transactions on Knowledge and Data Engineering*.

Gao, Ji et al. (2018). "Black-box generation of adversarial text sequences to evade deep learning classifiers". In: *IEEE Security and Privacy Workshops (SPW)*, pp. 50–56.

Ge, Li and Li-Juan Ge (2016). "Feature extraction of time series classification based on multi-method integration". In: *Optik* 127.23, pp. 11070–11074.

Gil, Manuel, Fady Alajaji, and Tamas Linder (2013). "Rényi divergence measures for commonly used univariate continuous distributions". In: *Information Sciences* 249, pp. 124–131.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2014). "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572*.

Guo, Zijian et al. (2019). "A Data Imputation Method for Multivariate Time Series Based on Generative Adversarial Network". In: *Neurocomputing* 360, pp. 185–197.

Hardkernel (2014). *ODROID-XU3*.

Hein, Matthias and Maksym Andriushchenko (2017). "Formal guarantees on the robustness of a classifier against adversarial manipulation". In: *Advances in Neural Information Processing Systems*, pp. 2266–2276.

Hendrycks, Dan et al. (2019). "Scaling out-of-distribution detection for real-world settings". In: *arXiv preprint arXiv:1911.11132*.

Hosseini, Hossein et al. (2017). "On the limitation of convolutional neural networks in recognizing negative images". In: *16th International Conference on Machine Learning and Applications (ICMLA)*. IEEE.

Hu, Yupeng et al. (2021). "Artificial intelligence security: Threats and countermeasures". In: *ACM Computing Surveys (CSUR)* 55.1, pp. 1–36.

Huang, Chen et al. (2016). "Learning Deep Representation for Imbalanced Classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, Xiaowei et al. (2017). "Safety verification of deep neural networks". In: *International conference on computer aided verification*. Springer, pp. 3–29.

Hussein, Dina, Taha Belkhouja, et al. (2022). "Reliable Machine Learning for Wearable Activity Monitoring: Novel Algorithms and Theoretical Guarantees". In: *Proceedings of 41st International Conference on Computer-Aided Design (ICCAD)*. ACM, 33:1–33:9.

— (2023). "Energy-Efficient Missing Data Recovery in Wearable Devices: A Novel Search-Based Approach". In: *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, pp. 1–6.

Hussein, Dina, Ganapati Bhat, and Janardhan Rao Doppa (2022). "Adaptive Energy Management for Self-Sustainable Wearables in Mobile Health". In: *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pp. 11935–11944.

Hussein, Dina, Aaryan Jain, and Ganapati Bhat (2022). "Robust Human Activity Recognition Using Generative Adversarial Imputation Networks". In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 84–87.

Ignatov, Andrey (2018). "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks". In: *Applied Soft Computing* 62, pp. 915–922.

Ismail Fawaz, Hassan et al. (2019). "Deep learning for time series classification: a review". In: *Data mining and knowledge discovery*.

James, Gareth et al. (2013). *An Introduction to Statistical Learning*. Vol. 112. Springer.

Jiang, Kaiyong, Changbiao Huang, and Bin Liu (2011). "Part decomposing algorithm based on STL solid model used in shape deposition manufacturing process". In: *The International Journal of Advanced Manufacturing Technology*.

Junchi Y. Negar K., Niao H. (2020). "Global Convergence and Variance Reduction for a Class of Nonconvex-Nonconcave Minimax Problems". In: *NeurIPS*.

Karim, Fazle, Somshubra Majumdar, and Houshang Darabi (2020). "Adversarial attacks on time series". In: *IEEE Transactions on pattern analysis and machine intelligence*.

Karimi, H., J. Nutini, and M. Schmidt (2016). "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition". In: *ECML*.

Kim, Hyunchoong et al. (2016). "Collaborative Classification for Daily Activity Recognition with a Smartwatch". In: *2016 IEEE Int. Conf. on Syst., Man, and Cybernetics (SMC)*, pp. 003707–003712.

Kingma, Diederik P and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *The Int. Conf. on Learning Representations (Poster)*.

Kolter, Z and A Madry (2018). "Tutorial adversarial robustness: Theory and practice". In: *NeurIPS*.

Kunze, Kai and Paul Lukowicz (2014). "Sensor Placement Variations in Wearable Activity Recognition". In: *IEEE Perv. Comput.* 13.4, pp. 32–41.

Kunze, Kai, Paul Lukowicz, et al. (2009). "Which Way am I Facing: Inferring Horizontal Device Orientation From an Accelerometer Signal". In: *2009 Int. Symp. on Wearable Computers*, pp. 149–150.

Kurakin, Alexey, Ian Goodfellow, and Samy Bengio (2016). "Adversarial examples in the physical world". In: *arXiv preprint arXiv:1607.02533*.

Kwapisz, Jennifer R, Gary M Weiss, and Samuel A Moore (2011). "Activity recognition using cell phone accelerometers". In: *ACM SigKDD Explorations Newsletter* 12.2, pp. 74–82.

Laidlaw, Cassidy and Soheil Feizi (2019a). "Functional Adversarial Attacks". In: *Advances in Neural Information Processing Systems (Neur'IPS)*.

— (2019b). "Functional adversarial attacks". In: *Advances in Neural Information Processing Systems (Neur'IPS)*, pp. 10408–10418.

Laptev, Nikolay, Saeed Amizadeh, and Ian Flint (2015). "Generic and Scalable Framework for Automated Time-Series Anomaly Detection". In: KDD '15. Association for Computing Machinery.

Lara, Oscar D and Miguel A Labrador (2012). "A survey on human activity recognition using wearable sensors". In: *IEEE communications surveys & tutorials* 15.3, pp. 1192–1209.

Lee, Kimin, Honglak Lee, et al. (2018). "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples". In: *International Conference on Learning Representations (ICLR)*.

Lee, Kimin, Kibok Lee, et al. (2018). "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Li, B. et al. (2019). "Certified Adversarial Robustness with Additive Noise". In: *NeurIPS*.

Li, Bai et al. (2019). "Certified Adversarial Robustness with Additive Noise". In: *Advances in Neural Information Processing Systems*, pp. 9459–9469.

Liang, Shiyu, Yixuan Li, and Rayadurgam Srikant (2018). "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *International Conference on Learning Representations (ICLR)*.

Limaye, Ankur and Tosiron Adegbija (2018). "HERMIT: A Benchmark Suite for the Internet of Medical Things". In: *IEEE Internet Things J.* 5.5, pp. 4212–4222.

Lin, T., C. Jin, and M. I. Jordan (2020). "Near-optimal algorithms for minimax optimization". In: *Conference on Learning Theory*.

Lin, Zinan et al. (2020). "Using GANs for sharing networked time series data: Challenges, initial promise, and open questions". In: *Proceedings of the ACM Internet Measurement Conference (IMC)*.

Liu, Shengzhong et al. (2020). "Handling Missing Sensors in Topology-Aware IoT Applications with Gated Graph Neural Network". In: *Proc. IMWUT* 4.3, pp. 1–31.

Liu, Weitang et al. (2020). "Energy-based Out-of-distribution Detection". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Luo, Yonghong, Xiangrui Cai, et al. (2018). "Multivariate Time Series Imputation with Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by S. Bengio et al. Curran Associates, Inc.

Luo, Yonghong, Ying Zhang, et al. (2019). "E^{2}GAN: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research (JMLR)* 9.Nov, pp. 2579–2605.

Madry, Aleksander et al. (2017). "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083*.

Maetzler, Walter, Jochen Klucken, and Malcolm Horne (2016). "A Clinical View on the Development of Technology-Based Tools in Managing Parkinson's Disease". In: *Movement Disorders* 31.9, pp. 1263–1271.

Martín Abadi and et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL:

McKinney, Wes, Josef Perktold, and Skipper Seabold (2011). "Time series analysis in python with statsmodels". In: *Jarrodmillman Com*, pp. 96–102.

Mizell, David (2003). "Using Gravity to Estimate Accelerometer Orientation". In: *Int. Symp. Wearable Comput.* Pp. 252–252.

Mode, Gautam Raj and Khaza Anuarul Hoque (2020). "Adversarial examples in deep learning for multivariate time series regression". In: *arXiv preprint arXiv:2009.11911*.

Montgomery, Douglas C, Cheryl L Jennings, and Murat Kulahci (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, Omar Fawzi, et al. (2017). "Universal adversarial perturbations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE Computer Society, pp. 1765–1773.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard (2016). "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582.

Mosenia, Arsalan et al. (2017). "Wearable Medical Sensor-Based System Design: A Survey". In: *IEEE Trans. Multi-Scale Comput. Syst.* 3.2, pp. 124–138.

Müller, Meinard (2007). "Dynamic time warping". In: *Information retrieval for music and motion*, pp. 69–84.

Nalisnick, Eric et al. (2018). "Do deep generative models know what they don't know?" In: *arXiv preprint arXiv:1810.09136*.

Nawir, Mukrimah et al. (2016). "Internet of Things (IoT): Taxonomy of security attacks". In: *2016 3rd international conference on electronic design (ICED)*. IEEE, pp. 321–326.

Ozanne, Anneli et al. (2018). "Wearables in Epilepsy and Parkinson's disease – A Focus Group Study". In: *Acta Neurologica Scandinavica* 137.2, pp. 188–194.

Ozbayoglu, Ahmet Murat, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey". In: *arXiv preprint arXiv:2002.05786* 93, p. 106384.

Pang, Guansong, Chunhua Shen, Longbing Cao, et al. (2021). "Deep learning for anomaly detection: A review". In: *ACM computing surveys (CSUR)* 54.2, pp. 1–38.

Pang, Guansong, Chunhua Shen, and Anton van den Hengel (2019). "Deep anomaly detection with deviation networks". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 353–362.

Paparrizos, John and Michael J. Franklin (2019). "GRAIL: Efficient Time-Series Representation Learning". In: *Proc. VLDB Endowment.*

Papernot, Nicolas, Fartash Faghri, et al. (2018). "Technical Report on the CleverHans v2.1.0 Adversarial Examples Library". In: *arXiv preprint arXiv:1610.00768.*

Papernot, Nicolas, Patrick McDaniel, Ian Goodfellow, et al. (2017). "Practical black-box attacks against machine learning". In: *Proceedings of Asia Conference on Computer and Communications Security (ASIACCS).* ACM.

Papernot, Nicolas, Patrick McDaniel, Xi Wu, et al. (2016). "Distillation as a defense to adversarial perturbations against deep neural networks". In: *IEEE Symposium on Security and Privacy (SP)*, pp. 582–597.

Pires, Ivan Miguel et al. (2020). "Improving Human Activity Monitoring by Imputation of Missing Sensory Data: Experimental Study". In: *Future Internet* 12.9, p. 155.

Raghunathan, Aditi, Jacob Steinhardt, and Percy Liang (2018). "Certified defenses against adversarial examples". In: *arXiv preprint arXiv:1801.09344.*

Rajpurkar, Pranav et al. (2022). "AI in health and medicine". In: *Nature medicine* 28.1, pp. 31–38.

Rawat, MINA, Martin Wistuba, and Maria-Irina Nicolae (2017). "Harnessing model uncertainty for detecting adversarial examples". In: *NIPS Workshop on Bayesian Deep Learning.*

Reiss, Attila and Didier Stricker (2012a). "Creating and benchmarking a new dataset for physical activity monitoring". In: *Proceedings of the 5th international conference on pervasive technologies related to assistive environments*, pp. 1–8.

— (2012b). "Introducing a New Benchmarked Dataset for Activity Monitoring". In: *ISWC*, pp. 108–109.

Ren, Jie et al. (2019). "Likelihood Ratios for Out-of-Distribution Detection". In: *Advances in Neural Information Processing Systems (NeurIPS).*

Roggen, Daniel et al. (2010). "Collecting complex activity datasets in highly rich networked sensor environments". In: *2010 Seventh international conference on networked sensing systems (INSS)*. IEEE, pp. 233–240.

Ruff, Lukas et al. (2021). "A unifying review of deep and shallow anomaly detection". In: *Proceedings of the IEEE*.

Sakoe, Hiroaki (1971). "Dynamic-programming approach to continuous speech recognition". In: *1971 Proc. the International Congress of Acoustics, Budapest.*

Salvador, Stan and Philip Chan (2007). "Toward Accurate Dynamic Time Warping in Linear Time and Space". In: *Intelligent Data Analalysis.*

Samanta, Suranjana and Sameep Mehta (2017). "Towards crafting text adversarial samples". In: *arXiv preprint arXiv:1707.02812*.

Sastry, Chandramouli Shama and Sageev Oore (2020). "Detecting Out-of-Distribution Examples with Gram Matrices". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Shafahi, Ali et al. (2020). "Universal Adversarial Training". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.

Shi, Junhao et al. (2020). "Sensor-based Activity Recognition Independent of Device Placement and Orientation". In: *Trans. on Emerging Telecommun. Tech.* 31.4, e3823.

Shoaib, Muhammad et al. (2014). "Fusion of Smartphone Motion Sensors for Physical Activity Recognition". In: *Sensors* 14.6, pp. 10146–10176.

Shokoohi-Yekta, Mohammad et al. (2017). "Generalizing DTW to the multi-dimensional case requires an adaptive approach". In: *Data mining and knowledge discovery* 31.1, pp. 1–31.

Siddiqui, Shoaib Ahmed et al. (2019). "TSViz: Demystification of Deep Learning Models for Time-Series Analysis". In: *IEEE Access* 7, pp. 67027–67040.

Smith, Kaleb E and Anthony O Smith (2020). "Conditional GAN for timeseries generation". In: *arXiv preprint arXiv:2006.16477*.

Stisen, Allan et al. (2015). "Smart Devices are Different: Assessing and Mitigating mobile Sensing Heterogeneities for Activity Recognition". In: *Proc. ACM Conf. on Embedd. Networked Sensor Syst.* Pp. 127–140.

Talukder, Sabera et al. (2022). "Deep Neural Imputation: A Framework for Recovering Incomplete Brain Recordings". In: *arXiv:2206.08094*.

Texas Instruments Inc. (2018). *CC2652R Microcontroller*. [Online] , accessed August 12, 2022.

Tolstikhin, Ilya O, Bharath K. Sriperumbudur, and Bernhard Schölkopf (2016). "Minimax Estimation of Maximum Mean Discrepancy with Radial Kernels". In: *Advances in Neural Information Processing Syst.* Vol. 29.

Tramer, Florian et al. (2020). "On adaptive attacks to adversarial example defenses". In: *arXiv preprint arXiv:2002.08347*.

Tramèr, Florian et al. (2018). "Ensemble adversarial training: Attacks and defenses". In: *International Conference on Learning Representations (ICLR)*.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing Data Using t-SNE". In: *J. of Machine Learning Research* 9.11.

Van Erven, Tim and Peter Harremos (2014). "Rényi divergence and Kullback-Leibler divergence". In: *IEEE Transactions on Information Theory* 60.7, pp. 3797–3820.

Wang, Aiguo et al. (2016). "A Comparative Study on Human Activity Recognition Using Inertial Sensors in a Smartphone". In: *IEEE Sensors J.* 16.11, pp. 4566–4578.

Wang, Hanchen et al. (2023). "Scientific discovery in the age of artificial intelligence". In: *Nature* 620.7972, pp. 47–60.

Wang, M., E. X Fang, and H. Liu (2017). "Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions". In: *Mathematical Programming, Springer*.

Wang, William Yang, Sameer Singh, and Jiwei Li (2019). "Deep adversarial learning for nlp". In: *Proceedings of the Conference of the NAACL: Tutorials*. Ed. by Anoop Sarkar and Michael Strube. Association for Computational Linguistics, pp. 1–5.

Wang, Zhiguang, Weizhong Yan, and Tim Oates (2017). "Time series classification from scratch with deep neural networks: A strong baseline". In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Wang, Ziyu et al. (2020). "Further Analysis of Outlier Detection with Deep Generative Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Wen, Qingsong et al. (2020). "Time series data augmentation for deep learning: A survey". In: *arXiv preprint arXiv:2002.12478*.

Wilhelm, Mathias et al. (2015). "eRing: Multiple Finger Gesture Recognition with One Ring Using an Electric Field". In: *Proc. Int. Work. on Sensor-based Activity Recognition and Interaction*, pp. 1–6.

Wu, Lingfei et al. (2018). "Random warping series: A random features method for time-series embedding". In: *International Conference on Artificial Intelligence and Statistics*. PMLR.

Wu, Renjie and Eamonn J. Keogh (2020). "FastDTW is approximate and Generally Slower than the Algorithm it Approximates". In: URL:

Xiao, Chaowei et al. (2018). "Spatially transformed adversarial examples". In: *arXiv preprint arXiv:1801.02612*.

Xiao, Zhisheng, Qing Yan, and Yali Amit (2020). "Likelihood Regret: An Out-of-Distribution Detection Score For Variational Auto-encoder". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by H. Larochelle et al.

Xiong, Yuanhao and Cho-Jui Hsieh (2020). "Improved Adversarial Training via Learned Optimizer". In: *European Conference on Computer Vision*. Springer, pp. 85–100.

Yamin, Nuzhat, Ganapati Bhat, and Janardhan Rao Doppa (2022). "DIET: A Dynamic Energy Management Approach for Wearable Health Monitoring Devices". In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6.

Yan, Y. et al. (2020). "Optimal Epoch Stochastic Gradient Descent Ascent Methods for Min-Max Optimization". In: *NeurIPS*.

Yang, Jianbo et al. (2015). "Deep convolutional neural networks on multichannel time series for human activity recognition." In: *Ijcai*. Vol. 15. Buenos Aires, Argentina, pp. 3995–4001.

Yang, Jingkang et al. (2021). "Generalized out-of-distribution detection: A survey". In: *arXiv preprint arXiv:2110.11334*.

Yang, Yao-Yuan et al. (2020). "A Closer Look at Accuracy vs. Robustness". In: *Proceedings of Neural Information Processing Systems (NeurIPS)*.

Yoon, Jinsung, James Jordon, and Mihaela Schaar (2018). "GAIN: Missing Data Imputation Using Generative Adversarial Nets". In: *ICML*, pp. 5689–5698.

Yu, Qing and Kiyoharu Aizawa (2019). "Unsupervised out-of-distribution detection by maximum classifier discrepancy". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Zabib, David Zooker et al. (2017). "Vulnerability of secured IoT memory against localized back side laser fault injection". In: *2017 Seventh International Conference on Emerging Security Technologies (EST)*. IEEE, pp. 7–11.

Zappi, Piero et al. (2007). "Activity Recognition from On-Body Sensors by Classifier Fusion: Sensor Scalability and Robustness". In: *Proc. Int. Conf. on Intell. Sensors, Sensor Netw. and Info.* Pp. 281–286.

Zheng, S. et al. (2016). "Improving the Robustness of Deep Neural Networks via Stability Training". In: *CVPR*.

Zheng, Stephan et al. (2016). "Improving the Robustness of Deep Neural Networks via Stability Training". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 4480–4488.

Zheng, Zibin et al. (2017). "Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids". In: *IEEE Transactions on Industrial Informatics*.

Zhou, Da-Wei, Yang Yang, and De-Chuan Zhan (2021). "Learning to classify with incremental new class". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.6, pp. 2429–2443.

Zhou, Da-Wei, Han-Jia Ye, and De-Chuan Zhan (2021). "Learning placeholders for open-set recognition". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410.

APPENDIX

APPENDIX A

THEORETICAL ANALYSIS FOR TSA-STAT FRAMEWORK

A.1 Proof of Theorem 1

For a given input space $\mathbb{R}^{n\times T}$ *and* $d \geq 1$*, polynomial transformations allow more candidate adversarial examples than additive perturbations in a constrained space. If* $X \in \mathbb{R}^{n\times T}$ *and* $\mathcal{PT} : X \to \sum_{k=0}^{d} a_k\ X^k$*, then* $\forall X_{adv}$ *s.t.* $\|S_i(X_{adv}) - S_i(X)\|_\infty \leq \epsilon_i$*:*

$$\left\{X_{adv} = \mathcal{PT}(X),\ \forall a_k\right\} \supsetneq \left\{X_{adv} = X + \delta,\ \forall \delta\right\}$$

, $S_i \in \mathcal{S}^m(X) \bigcup Identity$*.*

Let $X \in \mathbb{R}^{n\times T}$ and $d \geq 1$. Let $\mathcal{PT}(\cdot)$ a polynomial adversarial transformation such that $\mathcal{PT} : X \to \sum_{k=0}^{d} a_k\ X^k$. We want to prove that a polynomial transformation can create an adversarial example X_{adv} that is out of the scope for additive perturbation with a constant δ. The main condition on X_{adv} is that $\|S_i(X_{adv}) - S_i(X)\|_\infty \leq \epsilon_i$ with $S_i \in \mathcal{S}^m(X) \bigcup Identity$.

In other words, if the given condition is satisfied, we will have:

$$\left\{X_{adv} = \mathcal{PT}(X),\ \forall a_k\right\} \supsetneq \left\{X_{adv} = X + \delta,\ \forall \delta\right\}$$

Suppose $\mathcal{A}$ be the space of all possible adversarial examples $\left\{X_{adv} = \mathcal{PT}(X),\ \forall a_k\right\}$ and $\mathcal{B}$ be the space of all possible adversarial examples $\left\{X_{adv} = X + \delta,\ \forall \delta\right\}$

- $S_i = Identity$: For $X_{adv} = \mathcal{PT}(X)$:

$$\begin{aligned} \|X_{adv} - X\|_\infty &\leq \epsilon_i \\ \|\sum_{k=0}^{d} a_k\ X^k - X\|_\infty &\leq \epsilon_i \\ \|a_0 + (a_1 - 1)X + \sum_{k=0}^{d} a_k\ X^k\|_\infty &\leq \epsilon_i \end{aligned}$$

Without loss of generality, let us consider $\|\cdot\|_\infty$ on the component $l \leq n$.

$$|a_0 + (a_1 - 1)X_l + \sum_{k=0}^{d} a_k \; X_l^k| \leq \epsilon_i$$

$$|a_0 + \beta(\{a_k, X_l\})| \leq \epsilon_i$$

Then $X_{adv} \in \mathcal{B}$ only if the function $\beta(\{a_k, X_l\}) = 0$ and $|a_0| \leq \epsilon_i$. Hence, by construction on the set of $\{a_k\}$, if $|a_0| > \epsilon_i$, we can create X_{adv} such that $|a_0 + \beta(\{a_k, X_l\})| \leq \epsilon_i$. Hence, we have $X_{adv} \in \mathcal{A}$ and $X_{adv} \notin \mathcal{B}$ (β depends on X, so it cannot be considered as a constant perturbation δ to be in $\mathcal{B}$).

• $S_i \in \mathcal{S}^m(X)$: Let us start with $S_i(\cdot) = \mu(\cdot)$. Similar to the previous case, and if we consider $\|\cdot\|_\infty$ on the component $l \leq n$:

$$\|\mu(X_{adv}) - \mu(X)\|_\infty \leq \epsilon_i$$

$$\left|\mu\left(\sum_{k=0}^{d} a_k \; X_l^k\right) - \mu(X_l)\right| \leq \epsilon_i$$

$$\left|\sum_{j=0}^{T} \frac{\sum_{k=0}^{d} a_k \; X_{l,j}^k}{T} - \sum_{j=0}^{T} \frac{X_{l,j}}{T}\right| \leq \epsilon_i$$

$$\left|\sum_{j=0}^{T} \frac{a_0 + (a_1 - 1)X_{l,j} + \sum_{k=0}^{d} a_k \; X_{l,j}^k}{T}\right| \leq \epsilon_i$$

If $X_{adv} \in \mathcal{B}$, then $\|\mu(X_{adv}) - \mu(X)\|_\infty = |\sum_{j=0}^{T} \frac{a_0}{T}|$. With the same construction logic as in the previous case, we can end with $X_{adv} \in \mathcal{A}$ and $X_{adv} \notin \mathcal{B}$. For the remaining cases of $S_i(\cdot)$ used in this work, as they are correletaed with μ, similar construction can be used.

A.2 Proof of Theorem 2

Let $X \in \mathbb{R}^{n \times T}$ be an input time-series signal. Let $n_P \sim \mathcal{N}(\mu_P, \sum)$ and $n_0 \sim \mathcal{N}(0, \sum)$. Given a classifier $F_\theta : \mathbb{R}^{n \times T} \to Y$ that produces a probability distribution $(p_1, \cdots, p_k)$ over k labels for $F_\theta(X + n_P)$ and another probability distribution $(p_1^0, \cdots, p_k^0)$ for $F_\theta(X + n_0)$. To guarantee that $arg\max_{p_i} \; p_i = arg\max_{p^0} \; p^0$, the following condition must be satisfied:

$$\|\mu_P\|_\infty^2 \leq \max_{\alpha \neq 1} \frac{2}{\alpha \cdot \sum^{(S)}} \cdot \left(-ln \left(1 - p_{(1)} - p_{(2)} + 2 \left(\frac{1}{2} \left(p_{(1)}^{1-\alpha} + p_{(2)}^{1-\alpha} \right) \right)^{\frac{1}{1-\alpha}} \right) \right)$$

where $\|\mu_P\|_\infty$ *is the maximum perturbation over the mean of the input's channels and* $\sum^{(S)}$ *is the sum of all elements of* $\sum$.

To prove this theorem, we call for a second Lemma provided in Bai Li et al., 2019):

Lemma 3. *Let* EP *and* $E0$ *be two probability distributions where* EP=$(p_1, \cdots, p_k)$ *and* $E0$=$(p_1^0, \cdots, p_k^0)$. *If* $arg \max_{p_i \in EP} \ p_i \neq arg \max_{p_i^0 \in E0} \ p_i^0$, *then:*

$$D_\alpha(EP \| E0) \geq -ln \left(1 - p_{(1)} - p_{(2)} + 2 \left(\frac{1}{2} \left(p_{(1)}^{1-\alpha} + p_{(2)}^{1-\alpha} \right) \right)^{\frac{1}{1-\alpha}} \right) \tag{A.1}$$

where $p_{(1)}$ *and* $p_{(2)}$ *are respectively the largest and second largest* $p_i \in EP$.

This Lemma provides a lower bound of the Rényi divergence for changing the index of the maximum of EP, which is useful for the derivation of our certification bound. If the estimated distributions EP and $E0$ have different indices for the maximum class probabilities, then $D_\alpha(EP \| E0) <$ RHS of Equation A.1.

Let $X \in \mathbb{R}^{n \times T}$ an input time-series signal, $n_P \sim \mathcal{N}(\mu_P, \sum)$ and $n_0 \sim \mathcal{N}(0, \sum)$, and a DNN classifier $F_\theta : \mathbb{R}^{n \times T} \to Y$ that produces a probability distribution over k candidate class labels: EP=$(p_1, \cdots, p_k)$ for $F_\theta(X + n_P)$ and another probability distribution $E0$=$(p_1^0, \cdots, p_k^0)$ for $F_\theta(X + n_0)$.

As a direct result from Lemma 1:

$$D_\alpha(EP \| E0) = \tfrac{\alpha}{2}(\mu_P - 0)^T \textstyle\sum_\alpha (\mu_P - 0) - \frac{1}{2(\alpha-1)} \ ln \frac{|\sum_\alpha|}{|\sum|^{1-\alpha}|\sum|^\alpha}$$

where $\sum_\alpha = \alpha \sum + (1 - \alpha) \sum = \sum$.

This results to:

$$D_\alpha(EP\|E0) = \frac{\alpha}{2}\mu_P^T \sum \mu_P - \frac{1}{2(\alpha-1)}\ ln(1)$$
$$= \frac{\alpha}{2}\mu_P^T \sum \mu_P$$

Since $\forall i : \mu_{P,i} \leq \|\mu_P\|_\infty$, we get

$$D_\alpha(EP\|E0) = \frac{\alpha}{2}\mu_P^T \sum \mu_P = \frac{\alpha}{2}\sum_i \sum_j \mu_{P,i} \times \mu_{P,j} \times \sum_{i,j}$$
$$\leq \frac{\alpha}{2}\|\mu_P\|_\infty^2 \sum^{(S)}$$

where $\sum^{(S)}$ is the sum of all elements of $\sum$.

To guarantee that $arg\max_{p_i}\ p_i = arg\max_{p_i^0}\ p_i^0$, the following condition must be satisfied from Lemma 3:

$$D_\alpha(EP\|E0) < -ln(1 - p_{(1)} - p_{(2)} + 2\left(\frac{1}{2}\left(p_{(1)}^{1-\alpha} + p_{(2)}^{1-\alpha}\right)\right)^{\frac{1}{1-\alpha}}$$

This implies:

$$\frac{\alpha}{2}\|\mu_P\|_\infty^2 \sum^{(S)} < -ln(1 - p_{(1)} - p_{(2)} + 2\left(\frac{1}{2}\left(p_{(1)}^{1-\alpha} + p_{(2)}^{1-\alpha}\right)\right)^{\frac{1}{1-\alpha}}$$

which leads us to:

$$\|\mu_P\|_\infty^2 < \frac{2}{\alpha \times \sum^{(S)}} \times \left(-ln(1 - p_{(1)} - p_{(2)} + 2\left(\frac{1}{2}\left(p_{(1)}^{1-\alpha} + p_{(2)}^{1-\alpha}\right)\right)^{\frac{1}{1-\alpha}}\right)$$

Hence, our result.

While the proposed certification in Theorem 2 uses the Lemma 3 provided in Bai Li et al., 2019, the derivation of the bound is different in terms of the following aspects that are not covered in Bai Li et al., 2019 :

- The use of a multivariate Gaussian distribution of the noise. The main difference in our work is that we use a multivariate Gaussian distribution that is characterized by mean vector μ and a covariance matrix $\sum$. This general formulation of multivariate

Gaussian distributions results in the conclusion of Theorem 2 because it is applied on multivariate inputs $X \in \mathcal{R}^{n \times T}$. However, the standard Gaussian noise with 0-value mean and a single-value standard deviation σ employed in (Li et. al., 2019) is limited for our case. The theoretical analysis in Bai Li et al., 2019 is not general to the multivariate Gaussian distributions used in our work. This is due to the fact that the Renyi divergence of the noise distribution in (Li et. al., 2019) can be upper-bounded by a factor of the L_2 norm. This upper bound is not applicable for the multivariate Gaussian noise. Hence, we provide the proof of Theorem 2 to derive a certification robustness. The approach in Bai Li et al., 2019 can only be used for *univariate* time-series data. We provided a more general derivation for multi-variate time-series.

- The certification in Bai Li et al., 2019 is only provided for the Euclidean distance. For the theoretical analysis of TSA-STAT, we had to introduce the statistical features of time-series instead of euclidean distance. Hence, we proposed in our proof the use of a mean vector μ_P. This proof is not similar to the one provided in Bai Li et al., 2019. Additionally, we provided Lemma 2 with proof to extend the certification to other statistical features.

A.3 Proof of Lemma 2

If a certified bound δ has been generated for the mean of input time-series signal $X \in \mathbb{R}^{n \times T}$ and classifier F_θ, then certified bounds for other statistical/temporal features can be derived consequently.

In this section, we will work on other statistical constraints used in our experimental evaluation. Let $X \in \mathbb{R}^{n \times T}$ an time-series input signal. Let Σ be the positive semi-definite covariance matrix used for the additive multivariate Gaussian noise. The bound on the mean δ value is given by TSA-STAT certification algorithm. As $\|S_i(X)\|_\infty$ is equal to the value of S_i on one of the channels n, let us consider for simplicity of this proof only that channel.

Hence, the derivation of the bounds for other statistical features is as follows:

- RMS $= \sum \frac{x_i^2}{n}$

  $$\sigma^2 = \sum \frac{(x_i - \mu)^2}{n} = \sum \frac{x_i^2 - 2x_i\mu + \mu^2}{n} = RMS^2 - 2\mu \sum \frac{x_i}{n} + \sum \frac{\mu^2}{n} = RMS^2 - \mu^2$$

  $$\Rightarrow \max\|RMS\|_\infty = \delta^2 + \sigma^2$$

- Skewness $g = \sum \frac{(x_i - \mu)^3}{n \times \sigma^3}$

  Let $G(\mu) = \sum \frac{(x_i - \mu)^3}{n}$

  $$\frac{\partial G}{\partial \mu} = \sum \frac{\partial}{\partial \mu} \frac{(x_i - \mu)^3}{n} = -3 \times \sum \frac{(x_i - \mu)^2}{n} \neq 0 \ \forall \mu \text{ as } (\sigma \neq 0)$$

  Therefore, $G(\mu)$ is monotonic $\Rightarrow \max\|g\|_\infty = \frac{G(\delta)|}{\sigma^3}$

- Kurtosis $k = \sum \frac{(x_i - \mu)^4}{n \times \sigma^4} - 3$

  Let $K(\mu) = \sum \frac{(x_i - \mu)^4}{n}$, following the previous result on the skewness:

  $$\frac{\partial K}{\partial \mu} = \sum \frac{\partial}{\partial \mu} \frac{(x_i - \mu)^4}{n} \neq 0 \ \forall \mu$$

  Therefore, $K(\mu)$ is monotonic $\Rightarrow \max\|k\|_\infty = \frac{|K(\delta)|}{\sigma^4} - 3$

APPENDIX B

THEORETICAL ANALYSIS FOR DTW-AR FRAMEWORK

B.1 Proof of Observation 1

Let l_2 be the equivalent of Euclidean distance using the cost matrix in the DTW space. $\forall X \in \mathbb{R}^{n \times T}$, there exists $\epsilon \in \mathbb{R}^{n \times T}$ and an alignment path P such that $dist_P(X, X+\epsilon) \leq \delta$ and $l_2(X, X+\epsilon) > \delta$.

The existence of ϵ is guaranteed as follows: We know from the nature of the DTW algorithm and the alignment paths that for two time-series signals X and X', the optimal alignment path is not always the diagonal path. If ϵ does not exist, it means that for all signals X' that are different from X, the diagonal path is an optimal alignment path, which is absurd. Thus, $\epsilon = X' - X$ and it always exists for any time-series signal .

Let P_{diag} be the diagonal alignment path in the cost matrix C.

For $X \in \mathbb{R}^{n \times T}$, let $\epsilon \in \mathbb{R}^{n \times T}$ such that the optimal alignment path P^* between X and $X + \epsilon$ is different than P_{diag}.

Let us suppose that there is no alignment path P between X and $X + \epsilon$ such that $dist_P(X, X+\epsilon) \leq \delta$ and $l_2(X, X+\epsilon) > \delta$. The last statement is equivalent to: $dist_P(X, X+\epsilon) < dist_{P_{diag}}(X, X+\epsilon)$.

Since we assumed that there is no alignment path P that satisfies this statement, this implies:

$$\begin{aligned} &\forall P \ , \ dist_P(X, X+\epsilon) \geq dist_{P_{diag}}(X, X+\epsilon) \\ &\Rightarrow dist_{P^*}(X, X+\epsilon) \geq dist_{P_{diag}}(X, X+\epsilon) \\ &\Rightarrow DTW(X, X+\epsilon) \geq dist_{P_{diag}}(X, X+\epsilon) \end{aligned}$$

Therefore, from the definition of $DTW(\cdot, \cdot)$ as a *min* operation during backtracing of the

DP process, we get:

$$\Rightarrow DTW(X, X+\epsilon) = dist_{P_{diag}}(X, X+\epsilon)$$

Hence, $P_{diag} = P^*$, which contradicts our main assumption in constructing ϵ such that $P_{diag} \neq P^*$.

Therefore, we conclude that:

$$\exists P \text{ s.t. } dist_P(X, X+\epsilon) \leq \delta \text{ and } l_2(X, X+\epsilon) > \delta$$

B.2 Proof of Theorem 3

For a given input space $\mathbb{R}^{n\times T}$, a constrained DTW space for adversarial examples is a strict superset of a constrained euclidean space for adversarial examples. If $X \in \mathbb{R}^{n\times T}$:

$$\left\{X_{adv} \middle| DTW(X, X_{adv}) \leq \delta\right\} \supset \left\{X_{adv} \middle| \|X - X_{adv}\|_2^2 \leq \delta\right\} \tag{B.1}$$

We want to prove that a constrained DTW space allows more candidates adversarial examples than a constrained Euclidean space. Let $X \in \mathbb{R}^{n\times T}$ be an input time-series and X_{adv} denote a candidate adversarial example generated from X. In the DTW-space, this requires that $DTW(X, X_{adv}) \leq \delta$. In the Euclidean space, this requires that $\|X - X_{adv}\|_2^2 \leq \delta$, which is equivalent to $dist_{P_{diag}}(X, X_{adv}) \leq \delta$.

Suppose $\mathcal{A}$ be the space of all candidate adversarial examples in DTW space $\left\{X_{adv} / DTW(X, X_{adv}) \leq \delta\right\}$ and $\mathcal{B}$ be the space of all candidate adversarial examples in Euclidean space $\left\{X_{adv} / \|X - X_{adv}\|_2^2 \leq \delta\right\}$.

To prove $\mathcal{A} \supsetneq \mathcal{B}$, we need to prove:

1. $\forall X_{adv} \in \mathcal{B} / X_{adv} \in \mathcal{A}$
2. $\exists X_{adv} / X_{adv} \in \mathcal{A}$ and $X_{adv} \notin \mathcal{B}$

Statement 1: Let $X_{adv} \in \mathcal{B}$. For the optimal alignment path P^* between X and X_{adv}, if:

- $P^* = P_{diag} \Rightarrow DTW(X, X_{adv}) = dist_{P_{diag}}(X, X_{adv})$

  $\Rightarrow X_{adv} \in \mathcal{A}$

- $P^* \neq P_{diag} \Rightarrow$ According to Observation 1:

  $DTW(X, X + \epsilon) < dist_{P_{diag}}(X, X + \epsilon)$

  $\Rightarrow X_{adv} \in \mathcal{A}$

Hence, we have $\forall X_{adv} \in \mathcal{B} / X_{adv} \in \mathcal{A}$.

Statement 2: Let $X_{adv} \in \mathcal{A}$ such that $P^* \neq P_{diag}$. Consequently, according to Observation 1, $dist_{P_{diag}}(X, X + \epsilon) > DTW(X, X + \epsilon)$.

$\Rightarrow dist_{P_{diag}}(X, X + \epsilon) > \delta$.

As the diagonal path corresponds to the Euclidean distance, we conclude that $X_{adv} \notin \mathcal{B}$.

Hence, $\exists X_{adv} / X_{adv} \in \mathcal{A}$ and $X_{adv} \notin \mathcal{B}$.

B.3 Proof of Observation 2

Given any alignment path P and two multivariate time-series signals $X, Z \in \mathbb{R}^{n \times T}$. If we have $dist_P(X, Z) \leq \delta$, then $DTW(X, Z) \leq \delta$.

Let P any given alignment path and P^* be the optimal alignment path used for DTW measure along with the DTW cost matrix C. Let us suppose that $dist_P(X, Z) > DTW(X, Z)$.

We denote P=$\{(1,1), \cdots, (i,j), \cdots, (T,T)\}$ and P^*=$\{(1,1), \cdots, (i^*,j^*), \cdots, (T,T)\}$. Let us denote by k the index at which, P and P^* are not using the same cells anymore, and by l, the index where P and P^* meet again using the same cells until (T,T) in a continuous way. By definition, $k > 1$ and $l < \min(len(P), len(P^*))$. For example, if P=$\{(1,1),(1,2),(2,2),(3,3),(3,4),(4,5),(5,5)\}$ and P^*=$\{(1,1),(1,2),(2,3),(3,4),(4,4),(5,5)\}$, then k=3 and l=6.

- If k=l, then P=P^*. Therefore, $dist_P(X, Z) > DTW(X, Z)$ is absurd.

- If $k \neq l$: To provide the $(k+1)^{th}$ element of P^*, we have $C_{(i^*_{k+1},j^*_{k+1})} = d(X_{i^*_{k+1}}, Z_{j^*_{k+1}}) + C_{(i^*_k,j^*_k)}$. To provide the $(k+1)^{th}$ element of P, we have $C_{(i_{k+1},j_{k+1})} = d(X_{i_{k+1}}, Z_{j_{k+1}}) + C_{(i_k,j_k)}$. Using the definition of the optimal alignment path provided in Equation 1, we have $C_{(i^*_{k+1},j^*_{k+1})} \leq C_{(i_{k+1},j_{k+1})}$.

  If we suppose that the remaining elements of P would lead to $dist_P(X,Z) < dist_{P^*}(X,Z)$, then this would lead to $C_{T,T} < DTW(X,Z)$, which contradicts the definition of DTW. Hence, we have $dist_P(X,Z) \leq dist_{P^*}(X,Z)$ implying that $dist_P(X,Z) > DTW(X,Z)$ is absurd.

Therefore, if we upper-bound $dist_P(X,Z)$ by δ for any given P, then we guarantee that $DTW(X,Z) \leq \delta$.

B.4 Proof of Theorem 4

For a given input space $\mathbb{R}^{n\times T}$ and a random alignment path P_{rand}, the resulting adversarial example X_{adv} from the minimization over $dist_{P_{rand}}(X, X_{adv})$ is equivalent to minimizing over $DTW(X, X_{adv})$. For any X_{adv} generated by DTW-AR using P_{rand}, we have:

$$\begin{cases} \texttt{PathSim}(P_{rand}, P_{DTW}) = 0 \ \ \& \\ dist_{P_{rand}}(X, X_{adv}) = DTW(X, X_{adv}) \end{cases} \tag{B.2}$$

where P_{DTW} is the optimal alignment path found using DTW computation between X and X_{adv}.

Let P_{rand} be the random alignment path over which the algorithm would minimize $dist_{P_{rand}}(X, X_{adv})$. For the ease of notation, within this proof, we will refer to X_{adv} by X'.

We have $dist_{P_{rand}}(X,X') = \sum_{(i,j)\in P_{rand}} d(X_i, X'_j)$. As $\forall i,j, d(X_i,X'_j) \geq 0$, then minimizing $dist_{P_{rand}}(X,X')$ translates to minimizing each $d(X_i,X'_j)$.

Let us denote $\min d(X_i,X'_j)$ by $d_{min}(X_i,X'_j)$, then
$\min dist_{P_{rand}}(X,X') = \sum_{(i,j)\in P_{rand}} d_{min}(X_i,X'_j)$.

Using the back-tracing approach of DTW to define the optimal alignment path, we want to verify if $\texttt{PathSim}(P_{rand}, P_{DTW}) = 0$. Let P_{rand} be the sequence of cells $\{c_{k,l}\}$ and P_{DTW} be the sequence $\{c_{k',l'}\}$. Every cell $\{c_{k',l'}\}$ in P_{DTW} is defined to be the successor of one of the cells $\{c_{k'-1,l'}\}$, $\{c_{k',l'-1}\}$, $\{c_{k'-1,l'-1}\}$ which will make the distance sum along P_{DTW} until the cell $\{c_{k',l'}\}$ be the minimum distance. As we have minimized the distance over the path P_{rand} to be $d_{min}(X_i, X'_j)$, the cells of P_{DTW} and P_{rand} will overlap. This is due to the recursive nature of DTW computation and the fact that the last cells in both sequences P_{DTW} and P_{rand} is the same (by the definition of DTW alignment algorithm).

Hence, $\forall (k,l) \in P_{rand}, (k',l') \in P_{DTW}$, we have $k = k'$ and $l = l'$.

Therefore, the optimal alignment between between X and X_{adv} will overlap with P_{rand} and we obtain $\texttt{PathSim}(P_{rand}, P_{DTW}) = 0$.

B.5 Proof of Corollary 1

Let P_1 and P_2 be two alignment paths such that $\texttt{PathSim}(P_1, P_2) > 0$. If X^1_{adv} and X^2_{adv} are the adversarial examples generated using DTW-AR from any given time-series X using paths P_1 and P_2 respectively such that $DTW(X, X^1_{adv}) = \delta$ and $DTW(X, X^2_{adv}) = \delta$, then X^1_{adv} and X^2_{adv} are not necessarily the same.

Let P_1 and P_2 be two alignment paths such that $\texttt{PathSim}(P_1, P_2) > 0$. We want to create an adversarial example from time-series signal X using one given alignment path. When using P_1, we will obtain $X_{adv,1}$ such that $DTW(X, X_{adv,1}) = \delta$, and when using P_2, we will obtain $X_{adv,2}$ such that $DTW(X, X_{adv,2}) = \delta$.

To show that $X_{adv,1}$ and $X_{adv,2}$ are more likely to be different, let us suppose that given P_1 and P_2, we always have $X_{adv,1} = X_{adv,2}$.

Again, to simplify notations, let us notate $X_{adv,1}$ by Z and $X_{adv,2}$ by Z' for this proof. As we have $DTW(X, Z) = DTW(X, Z') = \delta$, then $\sum_{(i,j)\in P_1} d(X_i, Z_j) = \sum_{(i,j)\in P_2} d(X_i, Z'_j)$. If we suppose that by construction Z is always equal to Z', this means that for $Z \neq Z'$, the

statement $\sum_{(i,j)\in P_1} d(X_i, Z_j) = \sum_{(i,j)\in P_2} d(X_i, Z'_j)$ does not hold. The last claim is clearly incorrect. Let us suppose that Z is pre-defined and we assume $Z \neq Z'$. Let the ensemble of indices $\{k\}$ refer to the indices where $Z_k \neq Z'_k$. This means that we have $k-1$ degrees of freedom to modify Z'_k to fix the equality $\sum_{(i,j)\in P_1} d(X_i, Z_j) = \sum_{(i,j)\in P_2} d(X_i, Z'_j))$.

Therefore, considering $\texttt{PathSim}(P_1, P_2) > 0$, we can construct $X_{adv,1} \neq X_{adv,2}$ such that $DTW(X, X_{adv,1}) = \delta$ and $DTW(X, X_{adv,2})$.

APPENDIX C

THEORETICAL ANALYSIS OF RO-TS FRAMEWORK

In this section, we present novel theoretical convergence analysis for SCAGDA algorithm, which helps us in understanding its behaviour and performance. As mentioned in the previous section, for the problem in Equation (6.7), existing theoretical analysis of SGDA (T. Lin, Jin, and Jordan, 2020; Y. Yan et al., 2020), stochastic alternating gradient descent ascent (SAGDA) (Junchi Y., 2020) require to compute exact gradient of $g(h_i(a_i))$ at each iteration, and stochastic compositional gradient algorithms for minimization problems (M. Wang, Fang, and H. Liu, 2017; T. Chen, Sun, and Yin, 2020) cannot handle the min-max optimization form.

Summary of results. Specifically, we answer the following question: *can we establish convergence guarantee of our SCAGDA algorithm for nonconvex-nonconcave compositional min-max optimization problems?* Our result shows the iteration complexity of Algorithm 6 to converge to an ϵ-primal gap is $O(\frac{L^4}{\mu^6\epsilon^2})$ (L and μ will be introduced later). In addition, our results demonstrate the efficacy of the MA strategy: the approximation error $\|\frac{1}{n}\sum_{i=1}^n(\omega_K^i - h_i(a_i))\|^2$ is also bounded by ϵ in expectation when the ϵ-primal gap is achieved.

Setup and assumptions. We first state some basic notations and the required assumptions for our convergence analysis. As denoted below (6.7), $\mathcal{P}(w) = \max_a \sum_{i=1}^n \phi_i(w, a_i)$ is the primal function. The primal gap at w is defined as $P(w) - P^*$, where $P^* = \min_w P(w)$. Our theoretical analysis focuses on the convergence of primal gap. We also analyze the convergence of approximation error $\|\omega - h(a)\|^2$. Below are assumptions used for our analysis.

Definition 1. *If a function $f(x)$ is C-Lipschitz continuous, then we have*

$$|f(x_1) - f(x_2)| \leq C\|x_1 - x_2\| \text{ and } \|\nabla f(x)\| \leq C.$$

Definition 2. *If a function $f(x)$ is L-smooth, then its gradient is L-Lipschitz continuous:*

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq L\|x_1 - x_2\|,$$

and equivalently

$$f(x_1) - f(x_2) \leq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{L}{2}\|x_1 - x_2\|^2.$$

Definition 3. *If a function $f(x)$ has a non-empty solution set and satisfies μ-PL condition, then we have*

$$\|\nabla f(w)\|^2 \geq 2\mu(f(x) - \min_{x'} f(x')).$$

Assumption 3. *Suppose $\mu, L, C_g, C_h, L_g, L_h \geq 0$.*

(i) $\phi(w, a)$ satisfies two side μ-PL condition:

$$\|\nabla_w \phi(w, a)\|^2 \geq 2\mu(\phi(w, a) - \min_{w'} f(w, a)),$$

$$\|\nabla_a \phi(w, a)\|^2 \geq 2\mu(\max_{a'} f(w, a) - \phi(w, a)).$$

(ii) $\phi(w, a)$ is L-smooth in w for fixed a.

(iii) $\phi(w, a)$ is L-smooth in a_i for fixed w.

(iv) g and h are C_g and C_h-Lipschitz continuous, respectively

(v) g and h are L_g and L_h-smooth, respectively.

(vi) $\exists\ \sigma > 0$ *s.t.* $\mathbb{E}[\|\nabla_w \phi_i(w, a_i) - \nabla_w \phi(w, a)\|^2] \leq \sigma^2$, $\mathbb{E}[\|\nabla_a f_i(w, a_i) - \nabla_a f_i(w, a_i)\|^2] \leq \sigma^2$, $\mathbb{E}[\|h_{i,j}(a_i) - h(a)\|^2] \leq \sigma^2$, *and* $\mathbb{E}[\|\nabla_{i,j} h(a_i) - \nabla h(a)\|^2] \leq \sigma^2$

Remark 1. In the (vi) assumption, $h_{i,j}(a_i)$ and $\nabla h_{i,j}(a_i)$ are also unbiased estimation of $h(a)$ and $\nabla h(a)$, respectively , so the bounded variance assumption is mild. We highlight that all the above assumptions are mild and are commonly used in the nonconvex optimization literature (Junchi Y., 2020; M. Wang, Fang, and H. Liu, 2017; T. Lin, Jin, and Jordan, 2020).

From the above assumptions, we have the following lemmas.

Lemma 4. *(Lemma A.2 and A.3 in Junchi Y., 2020) If ϕ is L-smooth and satisfies two side μ-PL condition, then $P(w)$ also satisfies μ-PL conditions, and $P(w)$ is L_P-smooth with $L_P = L + L^2/\mu \leq 2L^2/\mu$ when $L \geq 1$ and $\mu \leq 1$.*

Lemma 5. *((Karimi, Nutini, and Schmidt, 2016)) If $f(x)$ is L-smooth and satisfies μ-PL condition, then it also satisfies μ-error bound condition $\|\nabla f(x)\| \geq \mu\|x_p - x\|$ where x_p is the projection of x onto the optimal set. Also it satisfies μ-quadratic growth condition $f(x) - \min_x f(x) \geq \frac{\mu}{2}\|x_p - x\|^2$. Conversely, if $f(x)$ is L-smooth and satisfies mu-error bound, then it satisfies $\frac{\mu}{L}$-PL condition.*

C.1 Main Results

In this section, we denote $\omega = (\omega_1, \cdots, \omega_n)$ as the concatenation of all ω_i. The following theorem is our main convergence result for SCAGDA as shown in Algorithm 6.

Theorem 8. *Suppose Assumptions 1 hold. Set $\eta_k = \eta = O(\frac{\mu^5}{L_g^2 L^4}\epsilon^2)$, $\gamma_k = \gamma = O(\frac{\mu^2\eta}{L^2})$ and $\beta = \sqrt{18\mu\eta}$. After running Algorithm 6 for the total number of iterations $K = \tilde{O}(\frac{L_g^2 L^4}{\mu^5\epsilon^2})$ ($\tilde{O}$ hides logarithmic factor), we have*

$$\mathbb{E}[P(w_K) - P^*] + \frac{1}{8}\mathbb{E}[P(w_K) - \phi(w_K, a_K)] + \Big(\frac{4C_h^4 L_g^4 \eta_K}{\mu^5}\Big)^{1/2}\mathbb{E}[\|\omega_K - h(a_{K-1})\|^2] \leq \epsilon$$

Remark 2. The above theorem gives us two critical observations of the behavior of SCAGDA. **(1)** After running K iterations of SCAGDA, the primal gap $P(w_{K+1}) - P^*$ converges to ϵ in expectation, since all terms in the left hand side of the inequality are non-negative. This result shows that SCAGDA is able to effectively solve the compositional min-max optimization problem shown in Equation (6.7). **(2)** The iteration complexity of SCAGDA is $O(1/\epsilon^2)$. To put this result in perspective, we compare to related theoretical results. The rate for nonconvex-nonconcave min-max problem without compositional structure is shown to be $O(1/\epsilon)$ (Junchi Y., 2020). However, this improvement requires unbiased

estimation of the gradient and computing the of $g(h_i(a_i))$ at each iteration. Our iteration complexity is in the same order of that for (T. Chen, Sun, and Yin, 2020), whose convergence result is $O(1/\epsilon^2)$ for nonconvex compositional minimization problems. The difference is that their convergence metric is the average squared norm of gradients, while ours is for the primal gap. We would like to highlight that *ours is the first result on convergence rate for stochastic compositional min-max problems.*

Corollary 2. *After $K = \tilde{O}(\frac{L_g^2 L^4}{\mu^6 \epsilon^2})$ iterations of Algorithm 6, we have*

$$\mathbb{E}[\|\omega_{K+1,i} - h_i(a_{K+1,i})\|^2] \leq O(\epsilon)$$

Remark 3. The above result shows that as SCAGDA algorithm is executed, the approximation error of $\|\omega_{K+1,i} - h_i(a_{K+1,i})\|^2$ converges to ϵ in the expectation. For the condition numbers, we always have $L \geq \mu$. In practice, we usually set the accuracy level ϵ to a very small value, so the condition $\epsilon \leq O(L^3/\mu^2)$ generally hold. This result provides strong theoretical support that if we apply SCAGDA to optimize our ROTS problem in (6.5), it is able to approximate k_{GAK} on-the-fly. We only need a constant number of alighments for computing k_{GAK} in each iteration of SCAGDA. When we have ϵ-primal gap, we also achieve ϵ-accurate estimation of k_{GAK}.

C.2 Proof of Theorem 5 And Corollary 2

Proof of Theorem 5: Before we can start proof of the main theorem, we need the following three technical lemmas.

Lemma 6. *Suppose Assumption 1 holds. Assume P is L_P-smooth. For iteration k of Algorithm 6, we have*

$$\begin{aligned}\mathbb{E}[P(w_{k+1}) - P^*] \leq & P(w_k) - P^* - \frac{\eta_k}{2}\|\nabla P(w_k)\|^2 + \frac{L\eta_k^2\sigma^2}{2} \\ & + \frac{\eta_k}{2}\|\nabla_w \phi(w_k, a_k) - \nabla P(w_k)\|^2 \end{aligned} \tag{C.1}$$

Lemma 7. *Suppose Assumption 1 hold. For iteration k of Algorithm 6, we have*

$$\begin{aligned}
&\mathbb{E}[P(w_{k+1}) - \phi(w_{k+1}, a_{k+1})] \\
&\leq (1 - \frac{\mu\gamma_k}{2})\mathbb{E}[P(w_k) - \phi(w_k, a_k)] \\
&+ (1 - \frac{\mu\gamma_k}{2})(\frac{\eta_k}{2} + 2\eta_k + L\eta_k^2)\|\nabla P(w_k) - \nabla\phi_w^k\|^2 \\
&+ (1 - \frac{\mu\gamma_k}{2})(-\frac{\eta_k}{2} + 2\eta_k + L\eta_k^2)\|\nabla P(w_k)\|^2 \\
&+ 2\gamma_k C_h^2 L_g^2 \mathbb{E}[\|\omega_{k+1} - h(a_k)\|^2] + L\sigma^2(\eta_k^2 + \gamma^2(1 + 2C_g^2))
\end{aligned} \tag{C.2}$$

Lemma 8. *(Lemma 2 of (M. Wang, Fang, and H. Liu, 2017))*

$$\begin{aligned}
\mathbb{E}[\|\omega_{k+1} - h(a_k)\|^2] \leq &(1 - \beta_k)\|\omega_k - h(a_{k-1})\|^2 \\
&+ \frac{C_h^2\|a_k - a_{k-1}\|^2}{\beta_k} + 2\beta_k^2\sigma^2
\end{aligned} \tag{C.3}$$

Now we have the three important inequalities and can proceed to our proof for Theorem 5. The key idea is to construct a Lyapunov function that combines the above three inequalities together, and derive the convergence for this Lyapunov function. Let

$$\begin{aligned}
L_{k+1} :=&\mathbb{E}[P(w_{k+1}) - P^*] + \frac{1}{8}\mathbb{E}[P(w_{k+1}) - \phi(w_{k+1}, a_k)] \\
&+ \Big(\frac{4C_h^4 L_g^4 L^4 \eta_k}{\mu^5}\Big)^{1/2}\mathbb{E}[\|\omega_{k+1} - h(a_k)\|^2]
\end{aligned}$$

Set $\eta_k = \eta$, $\gamma_k = \gamma = \frac{36L^2\eta}{\mu^2}$, $\beta = (18\mu\eta)^{1/2}$. Recall that $\|\nabla P(w_k) - \nabla\phi_w^k\|^2 \leq \frac{2L^2}{\mu}(P(w_k) -$

$\phi(w_k, a_k)$). By Lemma 6 and 7, we have

$$\begin{aligned} L_{k+1} \leq & P(w_k) - P^* - \frac{\eta}{2}\|\nabla P(w_k)\|^2 + \frac{\eta}{2}\|\nabla P(w_k) - \nabla \phi_w^k\|^2 \\ & + \frac{L_P \eta^2 \sigma^2}{2} + \frac{1}{8}\Bigg((1 - \frac{\mu\gamma}{2})\mathbb{E}[P(w_k) - \phi(w_k, a_k)] \\ & + (1 - \frac{\mu\gamma}{2})(\frac{\eta}{2} + 2\eta + L\eta^2)\|\nabla P(w_k) - \nabla \phi_w^k\|^2 \\ & + (1 - \frac{\mu\gamma}{2})(-\frac{\eta}{2} + 2\eta + L\eta^2)\|\nabla P(w_k)\|^2 \\ & + L\sigma^2(\eta^2 + \gamma^2(1 + 2C_g^2))\Bigg) \\ & + \underbrace{\Big(2\gamma C_h^2 L_g^2 + \Big(\frac{L^4\eta}{\mu^5}\Big)^{1/2} 2C_h^2 L_g^2\Big)\mathbb{E}[\|\omega_{k+1} - h(a_k)\|^2]}_{=A} \end{aligned} \tag{C.4}$$

Recall that $\eta \leq \frac{1}{4L}$. Before bounding term A, we first simplify the coefficients of $\mathbb{E}[\|\nabla P(w_k)\|^2]$ and $\mathbb{E}[\|\nabla P(w_k) - \nabla \phi_w(w_k, a_k)\|^2]$ as follows

$$\begin{aligned} & \text{for } [\|\nabla P(w_k)\|^2]: \\ & -\frac{\eta}{2} + \frac{1}{8}(1 - \frac{\mu\gamma}{2})(-\frac{\eta}{2} + 2\eta + L\eta^2) \leq -\frac{\eta}{4}, \\ & \text{for } \mathbb{E}[\|\nabla P(w_k) - \nabla \phi_w(w_k, a_k)\|^2]: \\ & \frac{\eta}{2} + \frac{1}{8}(1 - \frac{\mu\gamma}{2})(\frac{\eta}{2} + 2\eta + L\eta^2) \leq \frac{7\eta}{8}. \end{aligned}$$

Due to L-smoothness of ϕ in a and Lemma 5, we can merge $\mathbb{E}[\|\nabla P(w_k) - \nabla \phi_w(w_k, a_k)\|^2]$ with $P(w_k) - \phi(w_k, a_k)$. Then the coefficient of $P(w_k) - \phi(w_k, a_k)$ in (C.4) can be derived as follows

$$\frac{1}{8}(1 - \frac{\mu\gamma}{2}) + \frac{2L^2}{\mu} \cdot \frac{7\eta}{8} = \frac{1}{8}(1 - \frac{18L^2\eta}{\mu} + 14\frac{L^2\eta}{\mu}) \leq \frac{1}{8}(1 - \frac{\mu\eta}{4}),$$

where the inequality is due to $\mu \leq L$. Now we employ Lemma 8 to bound term A in (C.4)

as follows

$$
\begin{aligned}
&\Big(2\gamma_k C_h^2 L_g^2 + \Big(\frac{L^4\eta}{\mu^5}\Big)^{1/2} 2C_h^2 L_g^2\Big)\mathbb{E}[\|\omega_{k+1} - h(a_k)\|^2] \\
\leq& 2C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(18\sqrt{\mu\eta}+1)(1-\beta)\|\omega_k - h(a_k)\|^2 \\
&+ 2C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(18\sqrt{\mu\eta}+1)(\frac{C_h^2\gamma^2 C_h^2 C_g^2}{\beta} + 2\beta^2\sigma^2) \\
\leq& 2C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(1-\frac{\mu\eta}{4})\|\omega_k - h(a_k)\|^2 \\
&+ 20C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(\frac{36^2 C_h^2 C_g^2 L^4\eta^2}{18\mu^4\sqrt{\mu\eta}} + 2\sigma^2 18^2\mu\eta). \qquad \text{(C.5)}
\end{aligned}
$$

By plugging the above simplified coefficients and merging (C.5) into (C.4), we have the following

$$
\begin{aligned}
L_{k+1} \leq& P(w_k) - P^* - \frac{\eta}{4}\|\nabla P(w_k)\|^2 \\
&+ \frac{1}{8}(1-\frac{\mu\eta}{4})(P(w_k) - \phi(w_k, a_k)) \\
&+ \frac{L_P\eta^2\sigma^2}{2} + \frac{1}{8}L\sigma^2(\eta^2 + \gamma^2(1+2C_g^2)) \\
&+ 2C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(1-\frac{\mu\eta}{4})\|\omega_k - h(a_k)\|^2 \\
&+ 20C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(\frac{36^2 C_h^2 C_g^2 L^4\eta^2}{18\mu^4\sqrt{\mu\eta}} + 2\sigma^2 18^2\mu\eta) \\
\leq& (1-\frac{\mu\eta}{4})(P(w_k) - P^*) + \frac{1}{8}(1-\frac{\mu\eta}{4})(P(w_k) - \phi(w_k, a_k)) \\
&+ 2C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(1-\frac{\mu\eta}{4})\|\omega_k - h(a_k)\|^2 + B \\
=& (1-\frac{\mu\eta}{4})L_k + B
\end{aligned}
$$

where the last inequality is due to μ-PL condition of P and we define

$$
\begin{aligned}
B :=& \frac{L_P\eta^2\sigma^2}{2} + \frac{1}{8}L\sigma^2(\eta^2 + \gamma^2(1+2C_g^2)) \\
&+ 20C_h^2 L_g^2 L^2/\mu^2 \sqrt{\frac{\eta}{\mu}}(\frac{36^2 C_h^2 C_g^2 L^4\eta^2}{18\mu^4\sqrt{\mu\eta}} + 2\sigma^2 18^2\mu\eta)
\end{aligned}
$$

Finally, we expand the above recursion from K to 1 and have

$$
L_K \leq (1-\frac{\mu\eta}{4})^K L_0 + B\sum_{k=0}^{K-1}(1-\frac{\mu\eta}{4})^k \leq \exp(-\frac{\mu\eta K}{4})L_0 + \frac{4B}{\mu\eta}
$$

We can verify that $\frac{4B}{\mu\eta} \leq 3\epsilon/4$ if η satisfies the following three inequalities:

$$\begin{aligned}
\eta \leq & \frac{\mu^4\epsilon}{16\sigma^2(163+324C_g^2)L^5}, \quad \eta \leq \frac{\mu^8}{23040C_h^6C_g^2L_g^2L^6} \\
\eta \leq & \frac{\mu^5\epsilon^2}{(4\cdot 160\cdot 324)^2C_h^2L_g^2L^4}
\end{aligned}$$

In practice, ϵ is usually set to a very small value, so we employ the last inequality as the upper bound of η. Then, to make $\exp(-\frac{\mu\eta K}{4})L_0 \leq \epsilon/4$, K has the following lower bound.

$$K \geq \frac{4}{\mu\eta}\log(4L_0/\epsilon) = \frac{(8\cdot 160\cdot 324)^2C_h^2L_g^2L^4}{\mu^6\epsilon^2}\log(4L_0/\epsilon)$$

Since L_{k+1} is an upper bound of $\mathbb{E}[P(w_{k+1})-P^*]$, to ϵ-primal gap, we have the total number of iterations $K = \tilde{O}\Big(\frac{L^4}{\mu^6\epsilon^2}\Big)$.

Proof of Corollary 2: To prove Corollary 2, let $\Delta_{K+1} = \mathbb{E}[\|\omega_{k+1}-h(a_k)\|^2]$ and use Lemma 8:

$$\begin{aligned}
\Delta_K \leq & (1-\beta)\Delta_{K-1} + \frac{C_h^2\gamma^2C_h^2C_g^2}{\beta} + 2\beta^2\sigma^2 \\
\leq & (1-\beta)^{K-1}\Delta_1 + \frac{C_h^2\gamma^2C_h^2C_g^2}{\beta^2} + 2\beta\sigma^2 \leq \epsilon,
\end{aligned}$$

where the second inequality is due to $\sum_{k=0}^{K-1}(1-\beta)^k \leq \frac{1}{\beta}$. The last inequality is due to

$$\begin{aligned}
& O((1-\beta)^{K-1}) \leq O(\exp(-\beta(K-1))) \\
& = O(\exp(-\sqrt{\mu\eta}\frac{L^6\log(1/\epsilon)}{\mu^6\epsilon^2})) \leq O(\epsilon^{1/\epsilon}) \leq O(\epsilon), \\
& O(\frac{\gamma^2}{\beta^2}) \leq O(\frac{L^6\eta^2}{\mu^4\mu\eta}) \leq O(\frac{L^6\eta}{\mu^4\mu}) \leq O(\epsilon), \\
& O(\beta) \leq O(\sqrt{\mu\eta}) \leq O(\epsilon).
\end{aligned}$$

APPENDIX D

ADDITIONAL EXPERIMENTAL RESULTS ON DTW-AR FRAMEWORK

Discussion on kNN-DTW based classification. It has been shown previously (Dau, Silva, et al., 2018; Shokoohi-Yekta et al., 2017) that the Nearest-Neighbour (NN) algorithm is suitable for time-series classification using DTW measure. Nevertheless, DNN classifiers show promising results (e.g., high accuracy, ease of deployment) in their use for the time-series domain:

- While kNN-DTW can be effective in many settings, as demonstrated by the results in Table D.1, the accuracy of kNN-DTW algorithm remains lower when compared to the deep models considered in our study on the real-world *multivariate* datasets used for evaluation in the main paper. To implement kNN-DTW, we consider the implementation of a regular kNN algorithm, and we use cDTW [36] with its provided public python implementation with $c = 10$ (This choice is based on the empirical evaluation in the cDTW paper (Dau, Silva, et al., 2018)).

- Using kNN-DTW ($k > 1$) instead of 1NN-DTW is not appropriate and principled for multivariate time-series data. By definition, DTW is an elastic similarity measure (the triangle inequality does not hold for DTW). Hence, if we have

  - X and X_1 are DTW-similar according to a warping path $Path_1$
  - X and X_3 are DTW-similar according to a warping path $Path_2$,

  then we cannot draw a straightforward conclusion that X_1 and X_2 are DTW-similar. Such an assumption yields a weak voting accuracy on the predicted label for high-dimensional and multivariate data. Consequently, kNN-DTW with $k > 1$ for multivariate data can fail. Our empirical results in Table D.1 corraborate this hypothesis

by showing that 1NN-DTW performs better then kNN-DTW ($k > 1$) in most cases.

- To be able to use kNN-DTW algorithms, the training data must be stored and accessible by the end-user. This rises many concerns including
  - Data privacy: For applications such as healthcare and finance, the data is privileged and needs to be secured. A deployed classification model should not have direct access to the data.
  - Storage space and scalability: For applications such as Human Activity Recognition where the models are deployed on resource-constrained hardware platforms, the use of resources for merely storing the data is inefficient. The data cannot be stored along with the classifier.

Therefore, the use of DNNs for time-series data is well-motivated. Hence, there is a clear need for focused investigation to study robustness of deep models for the time-series domain.

Table D.1 Accuracy (%) of kNN-DTW classifier vs. 1D-CNN classifier task on the clean multivariate time-series data.

	Atrial Fibrillation	Epilepsy	ERing	Heartbeat	RacketSports
1NN-DTW	38	56	85	63	75
5NN-DTW	13	40	86	67	70
10NN-DTW	36	32	75	68	65
1D-CNN	40	95	94	70	86

Comparison of DTW-based adversarial example generation. A different approach to create new examples that can be used for data augmentation is proposed by using resampling perturbations (Dau, Silva, et al., 2018). This approach ensures a close DTW measure to the original example. However, we would like to clarify that the algorithm proposed in Dau, Silva, et al., 2018 and our DTW-AR have different goals. While the method in Dau, Silva,

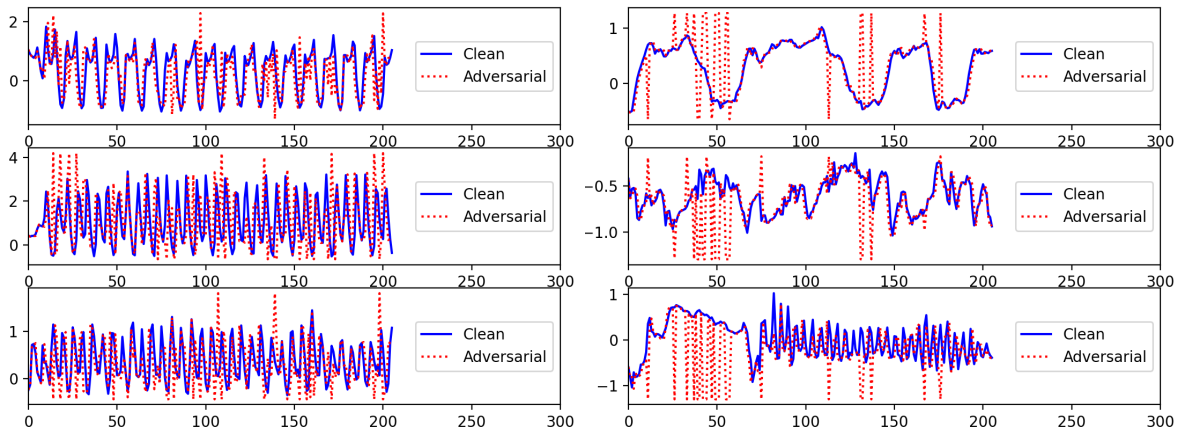


Figure D.1 DTW-AR adversarial examples from Epilepsy dataset using pre-defined warping path from user.

et al., 2018 aims for simple and effective DTW-based data augmentation, these examples cannot be considered as adversarial. Additionally, it does not provide control to the user over the warping path. Using DTW-AR and the tightness guarantees provided in Section 3.3, the user can generate several warped examples with full control over the warped timesteps. Finally, using Equation (5), the generated example is adversarial by definition. As a result, we obtain adversarial examples that remain close to the clean time-series input DTW-wise as illustrated in Figure D.1.

We use the method from Dau, Silva, et al., 2018 to generate additional examples to be used for adversarial training. Next, we compare the accuracy of the learned predictive models from adversarial training using DTW-AR and method in Dau, Silva, et al., 2018. Table D.2 shows the results on multiple real-world time-series datasets. The accuracy is evaluated on 1) Clean examples, 2) Examples generated by the warping function provided by (Dau, Silva, et al., 2018), and 3) DTW-AR adversarial examples. Table D.2 clearly show that adversarial training based on (Dau, Silva, et al., 2018) does not yield a robust model that can improve the performance of deep models against perturbations. Additionally, the same table explains that examples generated by method in Dau, Silva, et al., 2018 cannot be considered as adversarial. We clearly observe that these examples have low effect on decreasing the

models' classification accuracy performance (unlike DTW-AR based examples). Hence, we can conclude that the method in Dau, Silva, et al., 2018 and DTW-AR are complementary for DTW-based data generation tasks, where (Dau, Silva, et al., 2018) aims for simple and training-effective warped examples for standard classification tasks, and DTW-AR aims for improving the robustness of deep classifiers over time-series data.

Table D.2 Accuracy (%) of method in Dau, Silva, et al., 2018 and DTW-AR based adversarial training on testing examples from different datasets.

	Atrial Fibrillation			Epilepsy		
	Clean Exp.	(Dau, Silva, et al., 2018) Exp.	DTW-AR Exp.	Clean Exp.	(Dau, Silva, et al., 2018) Exp.	DTW-AR Exp.
(Dau, Silva, et al., 2018) Adv. training	42	42	29	90	90	26
DTW-AR Adv. training	42	38	82	98	93	96
	Heartbeat			RacketSports		
	Clean Exp.	(Dau, Silva, et al., 2018) Exp.	DTW-AR Exp.	Clean Exp.	(Dau, Silva, et al., 2018) Exp.	DTW-AR Exp.
(Dau, Silva, et al., 2018) Adv. training	70	68	20	86	83	52
DTW-AR Adv. training	75	72	96	86	80	92
	ERing					
	Clean Exp.	(Dau, Silva, et al., 2018) Exp.	DTW-AR Exp.			
(Dau, Silva, et al., 2018) Adv. training	90	89	43			
DTW-AR Adv. training	96	90	99			

Results on the full UCR multivariate dataset. First, we provide in Figures D.2, D.3 and D.4 the experiments conducted in the main paper on the **Effectiveness of adversarial attacks** and the **DTW-AR based adversarial training** on all the UCR multivariate dataset to our DTW-AR framework is general and highly-effective for all datasets.

In Figure D.2, we can observe that DTW-AR performs lower ($\alpha_{Eff} \leq 0.5$) for some cases. We explain below how the other baseline attacks fail to outperform the proposed DTW-AR method on the same datasets. In Figure D.5 and D.6, we show results to evaluate the effectiveness of baseline attacks against the models shown in Figure D.2. These results show that DTW-AR is more effective in fooling DNNs created using baseline attacks-based adversarial training. For datasets where DTW-AR did not succeed in fooling the deep models with a high score, Figure D.5 and D.6 show that baselines fail to outperform our proposed DTW-AR attack. We also demonstrate in Figure D.4 that the baselines are not suitable for

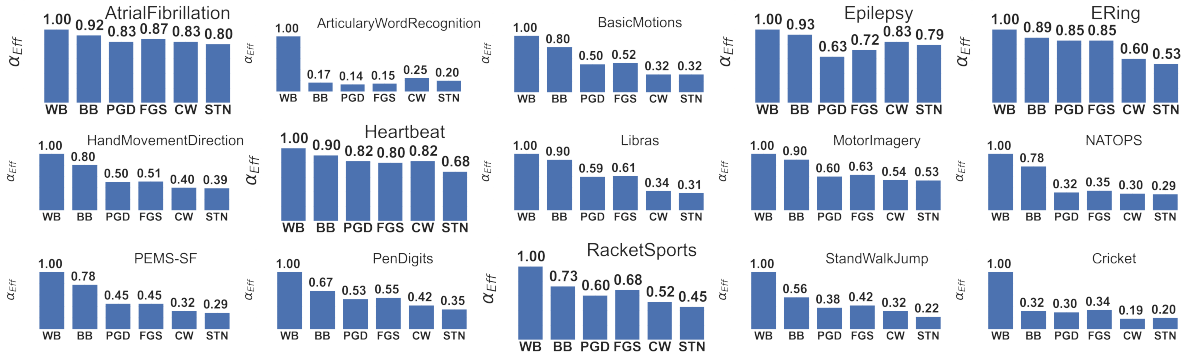


Figure D.2 Results for the effectiveness of adversarial examples from DTW-AR on different DNNs under white-box (WB) and black-box (BB) settings, and using adversarial training baselines (PGD, FGS, CW and STN) on all the UCR multivariate datasets.

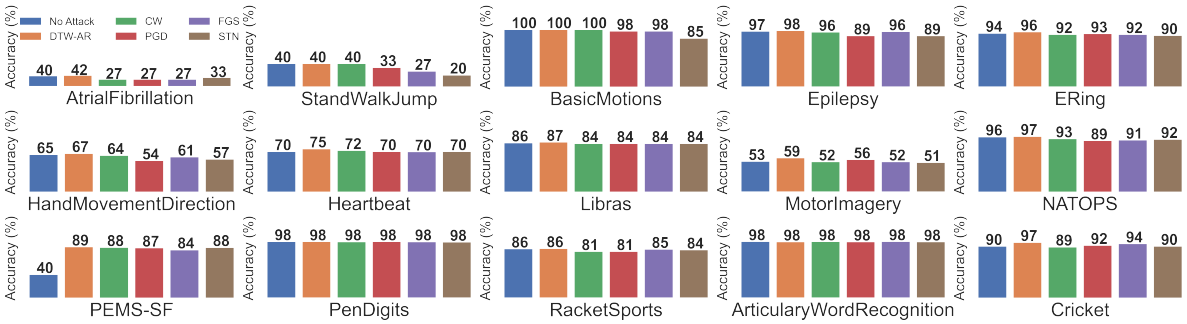


Figure D.3 Results of adversarial training using baseline attacks and DTW-AR on all the UCR multivariate datasets, and comparison with standard training without adversarial examples (No Attack) to classify clean data.

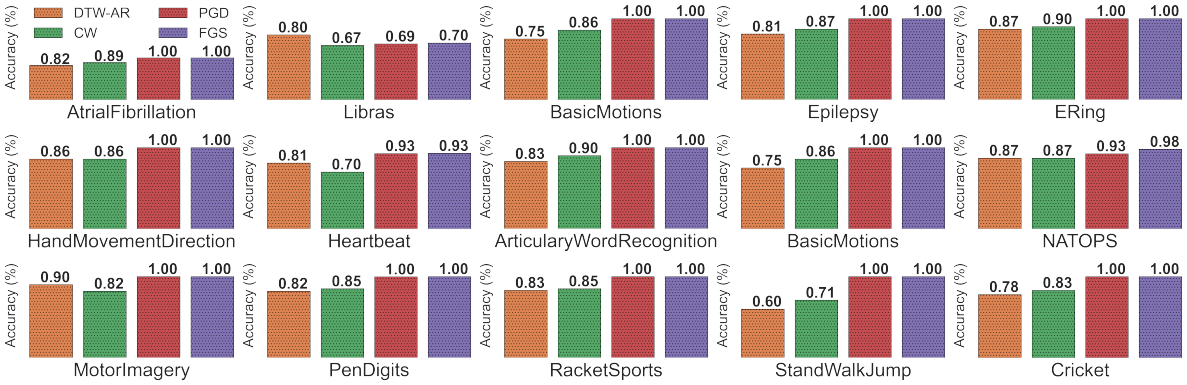


Figure D.4 Results of DTW-AR based adversarial training to predict the true labels of adversarial examples generated by DTW-AR and the baseline attack methods on all the UCR multivariate datasets. The adversarial examples considered are those that successfully fooled DNNs that do not use adversarial training.

time-series domain since DTW-AR based adversarial training is able to defend against these attacks.

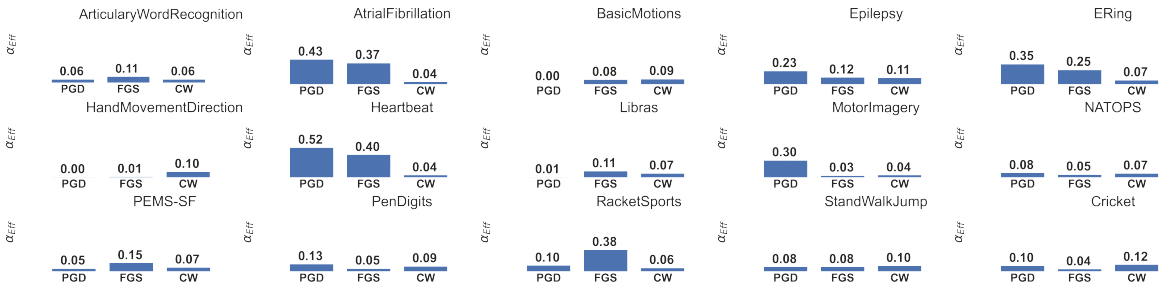


Figure D.5 Results for the effectiveness of adversarial examples from CW on different deep models using adversarial training baselines (PGD, FGS, CW).

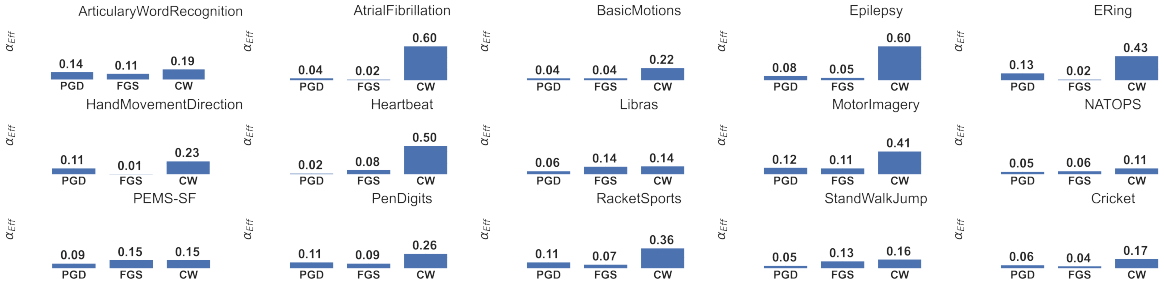


Figure D.6 Results for the effectiveness of adversarial examples from PGD and FGS on different deep models using adversarial training baselines (PGD, FGS, CW).

Results and Discussion on l_1 and l_∞. Figure D.7 shows the MDS results of `SC` and `Plane` in the space using l_1 as a similarity measure (left) and in the space using l_∞ as a similarity measure (right). We can observe that similar to the Euclidean space, there is a substantial entanglement between different classes. Thus, using l_1 and l_∞ comes with similar drawbacks to using the Euclidean space for adversarial studies.

Figure D.8 and D.9 show the results of DTW-AR based adversarial training against adversarial attacks generated using l_1 and l_∞ as a metric.

We conclude that DTW-AR is able to generalize against attacks in other spaces than the Euclidean one. Since the Manhattan distance is similar to the Euclidean distance in the point-to-point matching, and ∞-norm describes a signal by solely its maximum value, DTW measure is still considered a better similarity measure. The empirical success of the DTW-AR suggests that the framework can be further analyzed theoretically and empirically for future research into adversarially robust classification when compared to different alternative

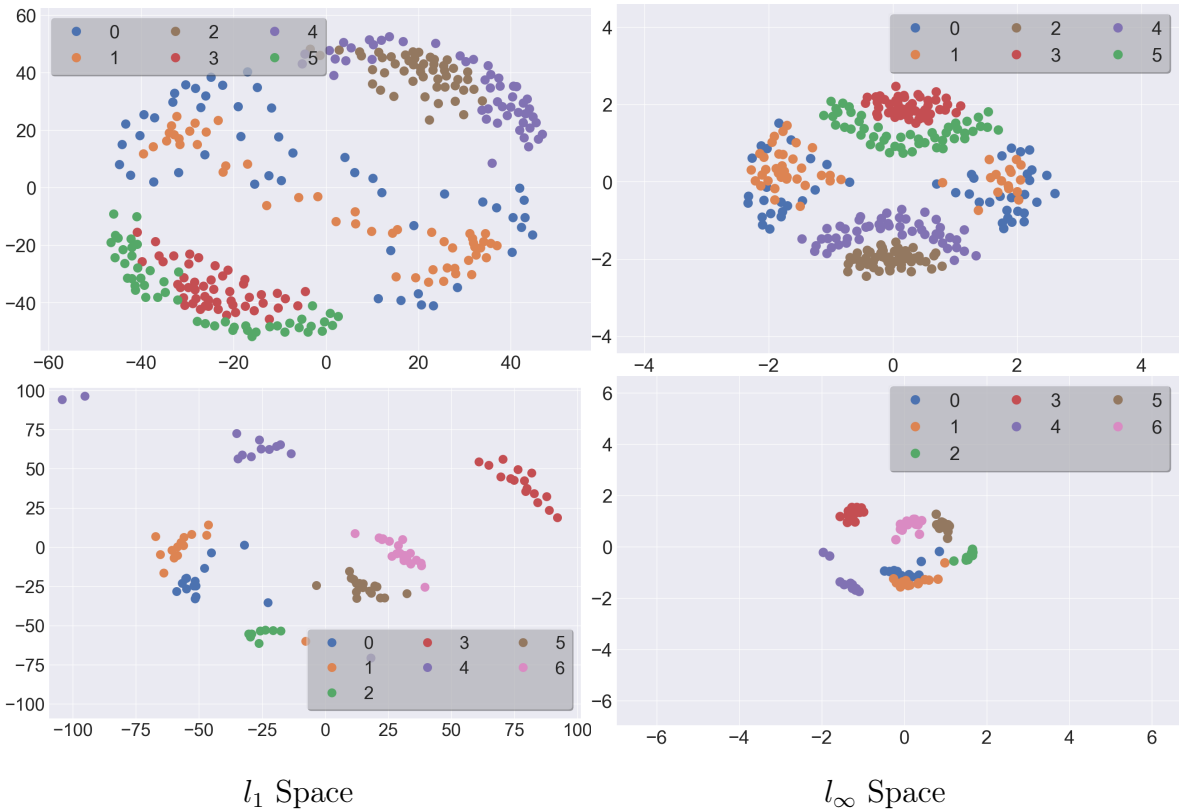


Figure D.7 Multi-dimensional scaling results showing the labeled data distribution in spaces using l_1 as a similarity measure (left column) and l_∞ (right column) for two datasets: SC (top row) and Plane (bottom row).

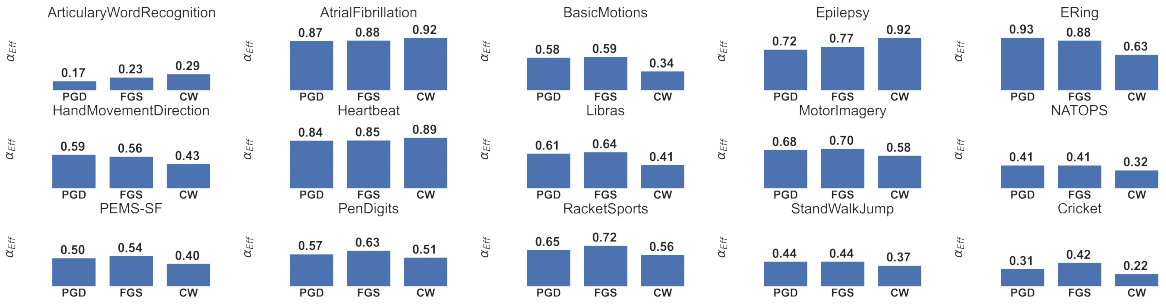


Figure D.8 Results for the effectiveness of adversarial examples from DTW-AR on different deep models using adversarial training baselines (PGD, FGS, CW) with l_1-norm.

similarity measures.

Comparison with (Karim, Majumdar, and Darabi, 2020) on the full MV UCR dataset. Figure D.10 shows that the observations made within the main paper are still valid over the different datasets.

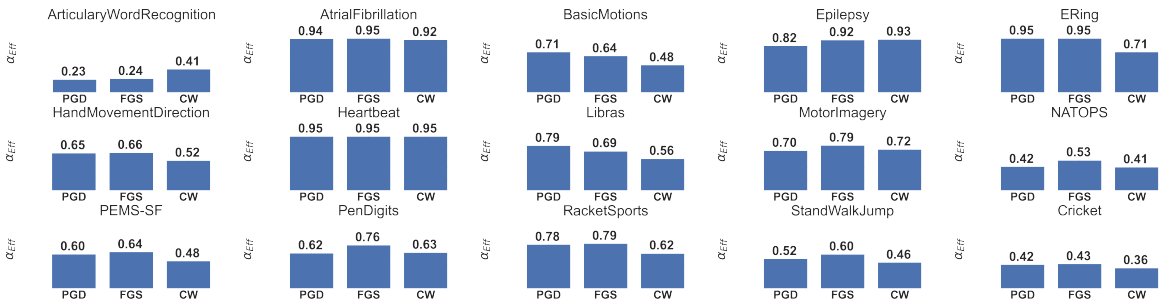


Figure D.9 Results for the effectiveness of adversarial examples from DTW-AR on different deep models using adversarial training baselines (PGD, FGS, CW) with l_∞-norm.

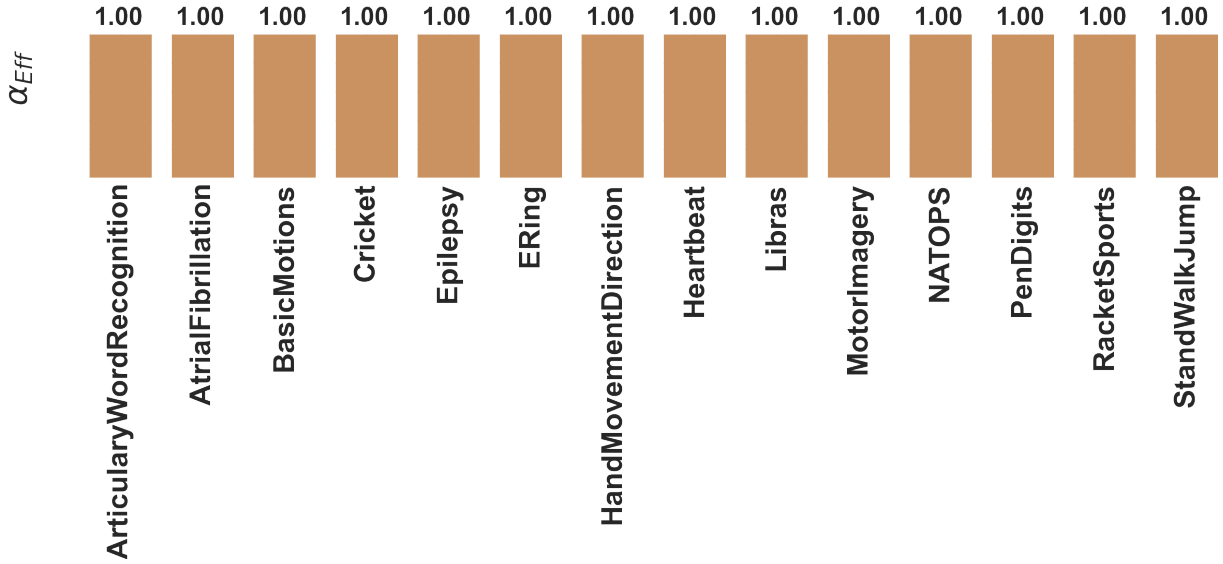


Figure D.10 Results of the success rate of DTW-AR adversarial trained model to predict the true label of adversarial attack generated from Karim, Majumdar, and Darabi, 2020.

DTW-AR on non-CNN models We want to clarify that the proposed DTW based adversarial framework is model-agnostic. The adversarial attack algorithm is based on Equations (5) and (7). Equation (7) relies on the output of the pre-softmax layer of the deep model and is independent of the model's core architecture. If the main layers of the model are based on convolutional layers, recurrent layers or attention layers, our proposed algorithm (Algorithm 1 in the main paper) is the same. As both LSTM and Transformer based models are typically used for forecasting applications [1,2], we focus on 1D-CNNs in this work. In Tables

D.3 and D.4, we provide additional results on non-CNN models. We clearly observe that DTW-AR remains effective on non-CNN models using the efficiency metric $\alpha_{Eff} \in [0, 1]$ over the created adversarial examples. $\alpha_{Eff} = \frac{\#\text{ Adv. examples s.t.} F(X)==y_{target}}{\#\text{ Adv. examples}}$ (higher means better attacks) was introduced in the paper to measure the capability of adversarial examples to fool a given DNN F_θ to output the true class-label. Therefore, DTW-AR generalizes over any given deep model. Furthermore, we note that all our assumptions and claims made in the paper are not based specifically on CNN models, but are general to any DNN classifier F_θ. Finally, we note that LSTM-based models are slower in execution than CNN models and Transformer-based models are too complex for classification tasks whereas CNNs perform similarly with less computational resources.

Table D.3 Results for the effectiveness α_{Eff} of adversarial examples from DTW-AR using LSTM-based deep neural network in a black-box (BB) setting.

	Clean Test Accuracy	α_{Eff}		
Dataset	Standard	Standard	PGD Adv. Trn.	FGS Adv. Trn.
Atrial Fibrillation	0.26	0.97	0.93	0.95
Epilepsy	0.61	0.89	0.75	0.75
ERing	0.74	0.98	0.91	0.91
Heartbeat	0.73	0.65	0.55	0.54
RacketSports	0.86	0.91	0.85	0.85

Runtime comparison of DTW-AR vs. Carlini & Wagner. As explained in Section 3, one main advantage of DTW-AR is reducing the time complexity of using DTW to create adversarial examples. We provide in Figure D.11 a comparison of the average runtime per iteration to create one targeted adversarial example by iterative baseline methods. We note that we only compare to CW because FGSM and PGD are not considered targeted attacks, and Karim et al. (Karim, Majumdar, and Darabi, 2020), fails to create adversarial examples for every input. While we observe that CW is faster, we note that we have already

Table D.4 Results for the effectiveness α_{Eff} of adversarial examples from DTW-AR using Transformer-based deep neural network in a black-box (BB) setting.

	Clean Test Accuracy	α_{Eff}		
Dataset	Standard	Standard	PGD Adv. Trn.	FGS Adv. Trn.
Atrial Fibrillation	0.22	0.95	0.95	0.95
Epilepsy	0.61	0.85	0.65	0.64
ERing	0.38	0.73	0.69	0.65
Heartbeat	0.71	0.91	0.85	0.85
RacketSports	0.66	0.73	0.66	0.65

demonstrated empirically (Figure D.2 and D.5) that DTW-AR always outperforms CW in both effectiveness of adversarial examples and adversarial training. We also observe differences in the DTW-AR's runtime across datasets. DTW-AR is relatively quick for small-size data such as *RacketSports* (30×6) and slower for large-size data such as *HeartBeat* (405×61). The additional runtime cost is explained by the proposed loss function in Equation 5.7 that guarantees a highly-similar adversarial example. For future work, we aim to optimize the implementation of DTW-AR to further reduce the runtime on large time-series datasets.

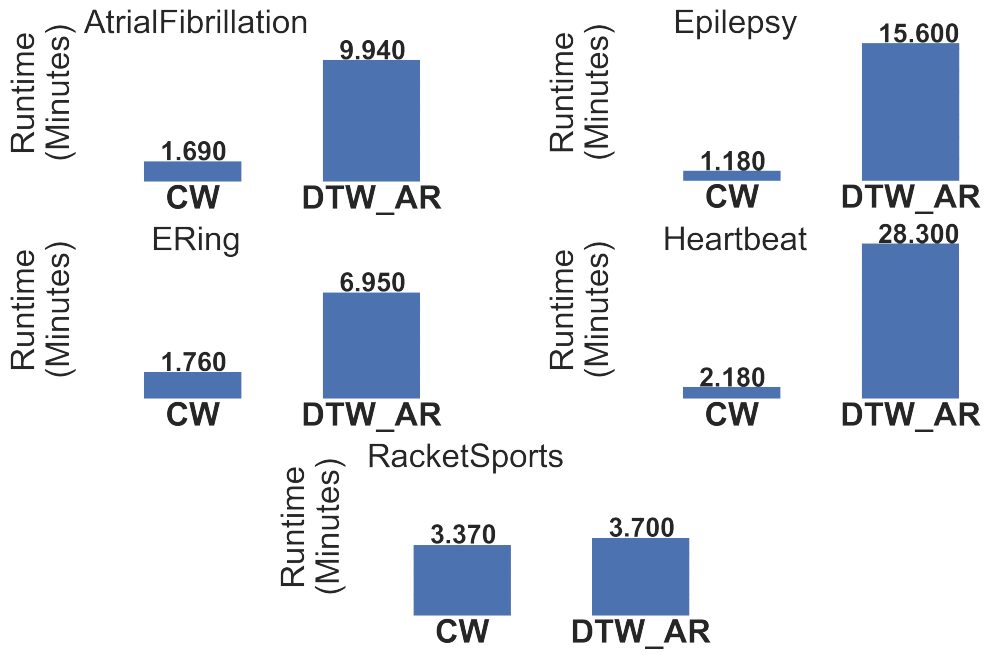


Figure D.11 Average runtime for CW and DTW-AR to create one targeted adversarial example (run on NVIDIA Titan Xp GPU).