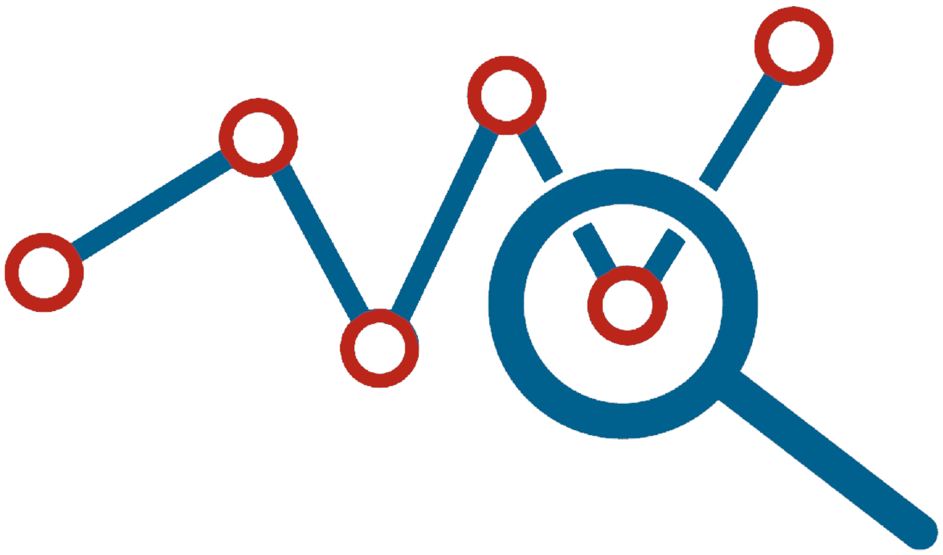
DISSERTATION

TIMEFRAME: A NOVEL FRAMEWORK FOR INTERPRETABLE AND PRIVACY-PRESERVING DEEP LEARNING FOR TIME SERIES ANALYSIS



Thesis approved by the
Department of Computer Science
of the RPTU Kaiserslautern-Landau
for the award of the Doctoral Degree

DOCTOR OF ENGINEERING
(DR.-ING.)

to

Dominique Mercier

Date of the viva: 2023-09-20
Dean: Prof. Dr. Christoph Garth

Reviewers:
Prof. Dr. Prof. h.c. Andreas Dengel
Prof. Dr. Sebastian Vollmer

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

DE-386

Dominique Mercier: *TimeFrame: A Novel Framework for Interpretable and Privacy-Preserving Deep Learning for Time Series Analysis*

SUPERVISORS:
Prof. Dr. Prof. h.c. Andreas Dengel
Dr. Sheraz Ahmed

CONTACT INFORMATION:

mercier@rhrk.uni-kl.de

EXECUTIVE SUMMARY

During our daily lives, we are confronted with vast amounts of data, the processing of which can dramatically influence our lives, both positively and negatively. The enormous amount of data (images, texts, tables, and time series), its variety and possible applications are not always obvious. Due to advancements in the internet of things (IoT), there exist billions of sensors that produce time series which can be found everywhere, whether in medicine, the financial sector or the agricultural economy. This incredible amount of time series data has many hidden features which are useful for industry as well as for daily use, e.g. improving the cancer prediction can save real human lives. Recently, several deep learning methods have been proposed for analyzing this time series data. However, due to their black box nature, their applicability is limited in critical sectors like medicine, finance, and communication. In addition, it is now a compulsion as per artificial intelligence (AI) Act and per General Data Protection Regulation (GDPR) to protect any sensitive data and provide explanations in safety-critical domains. To enable use of DNNs in a broader domain scope, this thesis presents a framework for privacy-preserved and interpretable time series analysis. *TimeFrame* consists of four main components, namely, post-hoc interpretability, intrinsic interpretability, direct privacy, and indirect privacy.

Interpretability is indispensable to avoid damaging people or the infrastructure. In the past years, the development mostly focused on image data, which prevented the full potential of DNNs in time series processing from being exploited. To overcome this limitation, *TimeFrame* introduces five (*Time to Focus*, *TSViz*, *TimeREISE*, *TSInsight*, *Data Lens*) novel post-hoc and two (*PatchX*, *P2ExNet*) novel intrinsic interpretability components. *TimeFrame* addresses multiple perspectives such as attribution, compression, visualization, influence, prototyping, and hierarchical splitting. Compared to existing methods, the components show better explanations, robustness, and scalability. Another crucial factor is the privacy when dealing with sensitive data and deep learning. In this context, *TimeFrame* introduces two (*PPML*, *PPML x XAI*) components for direct and one (*From Private to Public*) component for indirect privacy. These components benchmark privacy approaches, their effect on interpretability, and the synthetic generation of data to overcome privacy concerns.

TimeFrame offers a large set of interpretability and privacy components that can be combined and consider numerous different aspects. Furthermore, the novel approaches have shown to consistently outperform twenty existing state-of-the-art methods across up to 20 different datasets. To guarantee the fairness, various metrics were used including performance change, Sensitivity, Infidelity, Continuity, runtime, model dependency, compression rate, and others. This broad set of metrics makes it possible to provide guidelines for a more appropriate use of existing state-of-the-art approaches as well as the novel components included in *TimeFrame*.

ACKNOWLEDGMENTS

I would like to thank everyone who supported me during this Ph.D. thesis. Their support helped me a lot to write this thesis.

Supervisors

Firstly, I would like to thank Prof. Dr. Prof.h.c. Andreas Dengel for the opportunity to write this thesis. He gave me feedback during various Ph.D. colloquiums and provided me with hints towards interesting directions. Furthermore, I would like to thank Dr. Sheraz Ahmed for his continuous support and feedback. We had many fruitful discussions that led to several papers.

DFKI

I would like to thank my colleagues, the DFKI members and the students who worked together with me to create this thesis. My special thanks to Shoaib Siddiqui and Adriano Lucieri who actively supported me during this thesis. Both supported me with their ideas and worked together with me to solve various research questions. Next, I would like to thank Tahseen Raza Rizvi for the discussions and collaborations not only limited to the time series domain but also in the domain of natural language. Furthermore, thanks to the students that explored interesting topics and research questions during the seminars and projects I supervised.

Further Thanks

Further thanks to the people who supported me with administrative stuff. Here I would like to thank Brigitte Selzer, Thomas Kieninger, Stefan Agne, Nicolai Schwindt and Markus Junker.

— Thank you all, Dominique

LIST OF PUBLICATIONS AS PART OF THIS THESIS

Parts of the research and material (including figures, tables, algorithms and text passages) in this thesis have already been published in:

[1] D. Mercier, S. A. Siddiqui, M. Munir, A. Dengel, and S. Ahmed. "Tsviz: Demystification of deep learning models for time-series analysis." In: *IEEE Access* 7 (2019), pp. 67027–67040.

[2] D. Mercier, S. A. Siddiqui, A. Dengel, and S. Ahmed. "Interpreting deep models through the lens of data." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[3] D. Mercier, A. Dengel, and S. Ahmed. "P2exnet: Patch-based prototype explanation network." In: *International Conference on Neural Information Processing*. Springer. 2020, pp. 318–330.

[4] D. Mercier, S. A. Siddiqui, A. Dengel, and S. Ahmed. "TSInsight: A Local-Global Attribution Framework for Interpretability in Time Series Data." In: *Sensors* 21.21 (2021), p. 7373.

[5] D. Mercier, A. Dengel, and S. Ahmed. "PatchX: Explaining Deep Models by Intelligible Pattern Patches for Time-series Classification." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.

[6] D. Mercier, A. Lucieri, M. Munir, A. Dengel, and A. Sheraz. "Evaluating Privacy-Preserving Machine Learning in Critical Infrastructures: A Case Study on Time-Series Classification." In: *IEEE Transactions on Industrial Informatics* (2021).

[7] D. Mercier., J. Bhatt., A. Dengel., and S. Ahmed. "Time to Focus: A Comprehensive Benchmark using Time Series Attribution Methods." In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,* INSTICC. SciTePress, 2022, pp. 562–573. DOI: 10.5220/0010904400003116.

[8] D. Mercier, A. Dengel, and S. Ahmed. "TimeREISE: Time Series Randomized Evolving Input Sample Explanation." In: *Sensors* 22.11 (2022). DOI: 10.3390/s22114084.

[9] D. Mercier, S. Saifullah, A. Lucieri, A. Dengel, and S. Ahmed. "Privacy Meets Explainability: A Comprehensive Impact Benchmark." In: *arXiv preprint arXiv:2211.04110* (2022).

[10] D. Mercier, A. Lucieri, M. Munir, A. Dengel, and S. Ahmed. "PPML-TSA: A modular privacy-preserving time series classification framework." In: *Software Impacts* (2022), p. 100286.

[11] D. Mercier, A. Dengel, S. Ahmed, et al. "From Private to Public: Benchmarking GANs in the Context of Private Time Series Classification." In: *arXiv preprint arXiv:2303.15916v2* (2023).

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AI	Artificial Intelligence
AOPC	Area Over the Perturbation Curve
API	Application Programming Interface
AUC	Area Under the Curve
CDD	Critical Difference Diagram
CNN	Convolutional Neural Network
DL	Deep Learning
DP	Differential Privacy
DPFL	Differnetial Private Federated Learning
DTW	Dynamic Time Warping
FCN	Fully Connected Network
FDN	Fully Dense Network
FE	Federated Ensemble
FGSM	Fast Gradient Sign Method
FID	Frechet Inception Distance
FL	Federated Learning
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
HE	Homomorphic Encryption
IS	Inception Score
LSTM	Long Short Term Memory
ML	Machine Learning
MSE	Mean Squared Error
PPML	Privacy-Preserving Machine Learning
ReLU	Rectified Linear Unit
RDP	Renyi Differential Privacy
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMC	Secure Multi-party Computation
SVM	Support Vector Machine
TEE	Trusted Execution Environment
TS	Time Series
TSA	Time Series Analysis

VAE	Variational AutoEncoder
WAE	Wasserstein Variational Autoencoder
WGAN	Wasserstein Generative Adversarial Model
XAI	Explainable Artificial Intelligence

URI PREFIXES (CURIES)

Syn. Anomaly Detection Dataset:	https://bit.ly/2UNk0Lo
UEA and UCR Datasets:	https://www.timeseriesclassification.com/
UCI Datasets:	https://archive.ics.uci.edu/ml/index.php
GitHub Source Codes:	https://github.com/DominiqueMercier/
TSViz Dashboard:	https://tsviz.kl.dfki.de/

Part I

PREAMBLE

1 INTRODUCTION

1.1 MOTIVATION

Time Series (TS) occurs in all areas of our lives and can have crucial impact e.g. monitoring the health, finances or other critical domains. The amount of time series that is created every day presents us with the unfeasible task of analyzing them. This importance of time series data is emphasized by the field of data mining, which also accentuates the immense amount of time series and their relevance [32, 43, 154]. Although the number of publicly available time series data [27, 132], and approaches [8] are increasing continuously, the main focus of the research community is on image data. One reason for this trend is that, in theory, any ordered data can be considered as a time series problem [23, 71] leading to the impressive amount of available time series data.

TS are everywhere and have a crucial impact on our lives.

Due to the large amounts of data, the field of artificial intelligence has shown a growing interest in both research and industry [4]. Nowadays, deep learning models are at the forefront of technology in a range of different domains including image classification [67], object detection [48], speech recognition [24], text recognition [13], translation systems [150] and image captioning [61]. Not only the outstanding scaleability of these methods but also their superior performance led to the fact that they are very prominent in almost all application fields. While the use of neural networks solved the scalability problems and additionally made it possible to let the networks themselves decide which features are relevant.

AI is used in almost every domain and produces outstanding performances across various tasks.

However, this attention led to the discovery of crucial limitations and weaknesses when dealing with artificial intelligence. Resource management, efficiency, data security, and interpretability have become increasingly important. While in numerous instances it is enough to get the correct prediction from a model, there are also areas where an explanation is indispensable. Especially in safety-critical, medical and economic areas, it is unthinkable to accept a decision without any explanation as stated by Bibal et al. [12].

Deep learning methods lack interpretability, which is crucial in domains like healthcare.

Especially, the lack of interpretable decisions [82, 125] has introduced significant limitations in domains like business, finance, natural disaster management, healthcare, self-driving cars, industry 4.0 and counter-terrorism where reasons for reaching a particular decision are as important as the prediction [64]. According to Perc et al. [108], these limitations mainly originate from the social and the juristic domain. Relatively quickly, it has become clear that Explainable Artificial Intelligence (XAI) is a crucial topic due to legal restrictions. According to Karliuk [60] it was legally stipulated that neuronal networks, for example, may not be used in all areas of life as their interpretability and ethical problems are not solved yet. It was found that several aspects are relevant for the application of neural networks in the economy [109] including data protection, efficiency, and interpretability. All those aspects play a pivotal role in the deployment of neural networks. Specifically, the interpretability is necessary to take responsibility [12].

Explainable artificial intelligence enables the use of deep learning in various domains and makes it possible to improve state-of-the-art.

While the issue of interpretability has been addressed early on in other modalities, only a few approaches exist for Time Series Analysis (TSA) [122]. More important, according to Zhang and Zhu [160], the majority of these approaches are based on image analysis since the visual criteria and concepts are more intuitive for humans. Concepts such as color or the meaning of different objects are well-defined and explainable. Similarly, the sequence of words and their meaning are clearly defined. However, this is not the case for time series, the interaction between the channels, the length of the time series and the abstract values makes it almost impossible to understand all this data. These circumstances require explanatory methods specifically adapted or developed in the context of time series.

Interpretability methods require a lot of attention to be evaluated and, depending on the data, their complexity increases drastically.

In addition, these methods need to be evaluated cautiously as in the past it was shown that some of the existing interpretability methods produce different or wrong explanations. E.g. in the image domain, they serve as edge detectors highlighting sharp color changes within the data. Adebayo et al. showed for the image domain, existing interpretability methods work independent of the network parameters [3]. Similar findings have been discovered by Tomsett et al. [144]. This makes it even more important to carefully evaluate such methods on time series data to validate their correctness. A wrong explanation can be even worse than no explanation, as it provides the intuition that something was understood, although it is not the case.

Another indispensable aspect of the application of neural networks on time series is the security of data. Privacy treats and the challenge of protecting sensitive data has been of high interest and led to comprehensive work highlighting both existing treats and possible solutions [118]. Besides the attacks during training or after the release of the model, Al-Rubaie and Chang showed that it is insecure to process private data using neural networks. While on the one hand, interpretability is important according to the European regulations [114], it is also necessary to protect the private data. E.g. from a legal point of view, it is impossible to use patient data without protection. To enable the use of neural networks in safety-critical areas, it is therefore necessary to protect them from unintentional interference.

Privacy is one of the most important aspects when dealing and requires additional efforts.

As a model always stores information about the data it has been trained on, it is possible to retrieve sensitive data without even accessing the data directly. Some attack methods allow reconstructing or deanonymize data using only an already trained model. To solve this problem, privacy-preserving methods are necessary. Despite the variety of developed approaches, reviews of Privacy-Preserving Machine Learning (PPML) methods usually focus on the image domain [58, 140] while the specific applicability of methods on time series data is usually left unattended. Liu et al. provided a broad summary of different existing attack and defense methods, which further highlights how important that aspect is [79]. Although there exist methods that provide a secure way to process the data, it is not straightforward to apply these privacy-preserving methods as they can significantly lower the performance of the models.

More importantly, these methods have implications for interpretability techniques. However, it is necessary for a network to have both interpretability and privacy-protecting aspects so that it can be used in all domains in which time series exist. Exactly this is the goal of this work, both interpretability and privacy-preserving methods and their interaction have been analyzed. In particular, the area of time series analysis has been considered, as this area has usually receives less attention, although the relevance and immense availability of time series data offer an inexhaustible amount of information. Different interpretability methods have been evaluated and a set of novel approaches has been presented. In the PPML domain, the applicability to time series and the compatibility with interpretability methods has been evaluated. Both areas present solutions that provide explainable and privacy-protecting networks for time series analysis. The alternative solutions are packed together to a framework that provides interpretable and private processing methods for time series to enable the broader use of neural networks.

It is crucial to proper evaluate interpretability methods and privacy approaches used in the time series domain.

1.2 RESEARCH QUESTIONS & GOALS

Based on the previously mentioned aspects, the following research questions and goals have been extracted and addressed in this work:

1. **Question:** Would it be possible to retroactively incorporate an interpretability component to pre-existing high performing time series models? Would it be possible to improve state-of-the-art post-hoc interpretability methods for time series analysis and measure their performance appropriately?
   Goal: Most of the existing methods are developed and evaluated on image and text modalities. To understand the current state of existing interpretability approaches, it is important to thoroughly evaluate them on a large set of time series datasets and with different metrics. Especially, the choice of the metric can have a crucial impact on the performance of the interpretability method. The search and comparison of evaluation metrics is therefore one important part when judging the methods properly that does not result in biases towards a group of methods. Throughout a benchmark of the sate-of-the-art methods, novel approaches were developed. These post-hoc approaches are aligned with the time series domain and experience of the user to provide better explanations.

2. **Question:** Would it be possible to design deep neural networks in a way that they are interpretable by design? How well do they perform compared to neural networks that are not modified?
   Goal: Almost no work has been done concerning neural networks for time series that are interpretable by design. Therefore, it is highly important that approaches that cover intrinsic interpretability are designed for time series tasks. Two very prominent aspects that are considered in this work are prototypes and patches. For both, novel methods for time series models were designed to fill the lack with such methods for the time series domain.

3. **Question:** What is the impact of privacy-preserving methods for time series analysis concerning their performance and limitations? What are the trade-offs, and does it have requirements on the dataset or model used for the analysis?
   Goal: An evaluation of the most prominent privacy-preserving methods is required to understand their applicability in the context of time series.

This requires a definition of relevant metrics that covers aspects such as resource management and performance. Furthermore, it is important to provide an evaluation of the limitations in model architecture and evaluate the feasibility of privacy approaches. The benchmarking of privacy requires three important aspects: Performance drop, runtime increase, and limitations.

4. **Question:** How does privacy affect interpretability when neural networks are used in the context of time series? Are there limitations, and would it be possible to have both private and interpretable networks? Does the privacy change the explanation and lead to a misunderstanding?
   Goal: An extensive study was conducted to analyze the impact of privacy on interpretability. Furthermore, an evaluation was conducted to show if generated data can resolve the privacy constrains. Therefore, it is possible to train the model on synthetic data to remove the limitations of privacy. Using such a model enables its use without privacy approaches on top of the model or during the training process.

1.3 CONTRIBUTIONS

This thesis made several contributions both in the context of interpretable time series analysis and privacy preserved. Below, the different benchmarks and novel approaches are briefly explained. Detailed information about the technical aspects and their evaluation can be found in the corresponding chapters.

1.3.1 *Post-hoc Interpretability*

The comprehensive benchmark of existing widely used attribution methods as post-hoc interpretability methods is the first contribution of this work. Furthermore, after understanding the limitations, a novel better performing attribution method was proposed. Ultimately, a visualization framework was developed to align the explanation of attribution methods with the user experience. Especially, for the visualization, it is important to show compressed representations, which led to a further contribution to how to compress the data to only cover relevant features. With that, the question of data relevance was solved in a benchmark of influence functions.

1.3.1.1 *Contribution 1: Time to Focus*

The use of attribution methods in the time series domain is very common, however, there is no real benchmark on how they perform on time series datasets. Therefore, a comprehensive benchmark that uncovers the performance of twelve state-of-the-art attribution methods and eight metrics across five datasets was conducted. E.g. it is known that some attribution methods perform well independent of the model or that some metrics highly favor the results towards a given group of metrics. Concerning appropriate metrics usage, the thesis does not only cover an evaluation of the approaches, but further provides guidelines on when to use which approach and provides detailed information about possible applications for the different methods. The resulting guidelines can be used to apply attribution methods more properly and explore their limitations to come up with novel approaches that align better to the time series modalities.

1.3.1.2 *Contribution 2: TimeREISE*

TimeREISE is a novel attribution method that outperforms five existing state-of-the-art approaches on 17 time series datasets, as benchmarks have clearly highlighted the need for an approach that considers time series characteristics, such as the dependency between features. While in the image domain this dependency is fixed, it varies for time series datasets e.g. RGB channels are well-defined however the correlation between different sensors in the time series domain changes for each dataset. Furthermore, locality is a characteristic that is not given in the time series domain. Events that happen much earlier can have effects on late events. *TimeREISE* covers all the characteristics, i.e., locality, granularity, smoothness, complexity, and runtime.

1.3.1.3 *Contribution 3: TSViz*

TSViz is a visualization framework that can be applied to different network structures and attribution methods. Its need originates from the complex representation that time series data can have. In the image domain it is straightforward to visualize heatmaps, however, this does not work in the time series domain when the number of channels and time steps is not limited. Furthermore, it is important to consider the level of experience of the user. To overcome this, *TSViz* covers a service that analyzed a time series network concerning the attribution and further provides information about the network structure, similar filters within the network, importance of the input signal and network filters. All this information is aligned to the level of experience of the user to oppose only the information that is suitable for the end-user. Ultimately, the approach enables experienced users to tune the model, leading to a validation of their understanding and a better performing model using the insights and conclusions drawn thanks to the proposed approach.

1.3.1.4 *Contribution 4: TSInsight*

TSInsight is a novel framework that provides an input compression based on the network's needs. Specifically, the input is compressed in regions that are not used by the classifier to predict the class, whereas the data points used for the prediction are preserved. A good visualization is not only aligned to the user level, but further focuses only on important parts. The network defines which data is needed for the inference, and the remaining data is suppressed in a way that does not bias the performance of the model. To achieve this compression, a novel loss was proposed that is used to train an autoencoder to create compressed input for the trained classifier model. The evaluation has shown that this approach significantly outperforms the existing methods across eight datasets and preserves a high performance if the compressed input is used instead of the original input.

1.3.1.5 *Contribution 5: Data Lens*

Data lens provides an opportunity to identify samples which are favorable and those which are causing negative effects on the model. A detailed evaluation is made in this context, where three influence functions were compared. The goal of these functions is to detect which samples are related to each other, e.g. are harmful or helpful for the prediction process. To do so, mathematical estimations are used which do not require the retraining of the model. The benchmark across three datasets has shown that simple methods like the loss perform on par with the complex influence functions concerning misclassified samples and their harmful effect on the model.

1.3.2 *Intrinsic Interpretability*

Intrinsic methods are the counterpart to post-hoc methods, as they are directly integrated into the model architecture. Two very prominent approaches are the use of prototypes and patches. Both techniques are used in the image domain. However, they cannot be transferred to the time series domain without any adjustments. The contributions of this section include two novel approaches for the time series domain that have shown to outperform existing implementations of prototype and patch-based approaches.

1.3.2.1 *Contribution 6: PatchX*

PatchX is a novel two-step patch-based approach. Based on the idea that prototypes are specific for classes, a novel approach was proposed that divides the sample into patches and learns how class-specific a patch is. The individual patches are then classified, and moreover, an overall prediction is computed. This goes hand in hand with the principle of divide and conquer. A large problem is divided into smaller ones that are more interpretable. In addition, a visualization was proposed to show the parts of the sample that align to a specific class. Therefore, it is possible to have a better understanding which parts are relevant for which class. *PatchX* was evaluated on five datasets and compared to two state-of-the-art approaches.

1.3.2.2 *Contribution 7: P2ExNet*

P2ExNet is a novel approach that provides prototypes of the input signal to explain the decision. These are then used to explain the decision of the network based on similarities. The use of prototypes originates from the human behavior to compare new things to already seen things. Comparing this to a simple sample, the principle is to learn e.g. the prototype of wheels and other parts of a car and then explain the classification with these prototypes. Therefore, a novel architecture as well as a training process have been proposed. The *P2ExNet* has shown to learn understandable prototypes and outperforms two existing prototype or patch-based approaches. *P2ExNet* was evaluated on eight datasets.

1.3.3 *Direct Privacy Preservation*

Direct privacy preservation requires a privacy mechanism to protect sensitive data against an attacker. There exist several approaches that incorporate this during the training of a model. However, these approaches are not benchmarked on time series data. Furthermore, their impact on interpretability methods is not evaluated. Both aspects are addressed by the below listed contributions.

1.3.3.1 *Contribution 8: PPML*

This thesis provides an in-depth analysis of the use of preserving privacy methods (federated learning, differential privacy, and encryption) and their impact on the model performance. The aim was to uncover the model of deep learning-based models in the presence of privacy constraints. The application of privacy mechanisms such as differential privacy, federated learning and encryption is very common in deep learning. These approaches introduce noise, average values or encrypt values to protect the model during the training and inference. While hey are intensively benchmarked in the image domain, this was not the case for the time series domain. The contribution in this field is a benchmark of their applicability and on time series models. The benchmark across 16 datasets provides insights in the performance drops, runtime increases, hyperparameter tuning to achieve good results and the combination of these approaches. One highlight is the combination of differential privacy and federated learning to increase the privacy further.

1.3.3.2 *Contribution 9: PPML x XAI*

The contribution of this part is a comprehensive evaluation of attribution methods when using private models. Privacy can change the outcome of the attribution methods, which results in biases and misunderstanding. Throughout the benchmark across five datasets, the impact of the privacy has been shown. The results have indicated that the interpretability significantly suffers when differential privacy is used.

1.3.4 *Indirect Privacy Preservation*

A solution for privacy constrains can be the use of synthetic data. One contribution was mare in this topic field to highlight how data generation can be used to overcome privacy constrains and apply interpretability.

1.3.4.1 *Contribution 10: From private to public*

Generative models are models that generate data that looks similar to the original data. It is possible to use these to create data from sensitive data and use the generated data without constrains. Therefore, it is required that the data generated does not expose sensitive information. As part of this thesis, a benchmark across nine datasets has been made to train a generative model in a private manner. The generated data can then be used as a classifier without the need of additional privacy approaches and therefore does not harm the interpretability. The results indicate that the aligned approach works well across different time series datasets.

1.4 OVERVIEW

The rest of this work is structured as following. Part I with Chapter 2 which covers background relevant to understanding the thesis. Chapter 3 is divided into a section that covers related work from the interpretability domain (Section 3.1) and a section for the privacy domain (Section 3.2). After that, Chapter 4 provides details about *TimeFrame* the framework built on top of the proposed methods and existing approaches.

Part II presents a set of benchmarks and components related to post-hoc interpretability. Chapter 5 provides a comprehensive benchmark of existing state-of-the-art attribution methods. Chapter 6 introduces *TimeREISE* a novel attribution methods, Chapter 7 proposes *TSViz* as a visualization and optimization framework for time series networks, Chapter 8 proposes *TSInsight* as a compression framework, and Chapter 9 benchmarks influence functions.

Next, Part III addresses the intrinsic interpretability perspective. Two methods inspired by the image domain are presented in this context to enable the use of intrinsic interpretable time series analysis. In Chapter 10 a divide and conquer approach, and in Chapter 11 a prototype-based approach, are presented.

Part IV focuses on direct privacy and evaluates the impact of these approaches on attribution methods. Therefore, Chapter 12 presents a comprehensive benchmark of direct privacy methods, addressing their advantages and drawbacks. In addition, the interaction between privacy and interpretability is analyzed in Chapter 13.

To complete the privacy perspective, Part V evaluates the use of indirect privacy. Specifically, a generative approach applied to time series data is evaluated in Chapter 14.

Finally, Part VI discusses and summarizes this work. Chapter 15 discusses the components and provides a conclusion of the results achieved during this work. Furthermore, Chapter 16 provides some ideas about possible limitations and extensions to overcome those.

2 BACKGROUND

This work addresses the topic of interpretable and privacy-protecting neural networks for the analysis of time series. To answer the research questions and provide solutions in the best possible manner, it is necessary to have background knowledge in the following four areas: artificial intelligence, interpretability, privacy protection, and time series analysis. The following explains the necessary topics and lists the relevant terms.

2.1 PERFORMANCE & COMPUTATIONAL ASPECTS OF AI

Artificial Intelligence (AI) has become increasingly important lately. This increase in relevance is primarily due to numerous possible areas of application and outstanding results [99]. Although Deep Learning (DL) was founded in 1955, it has not been used extensively due to the traditional approaches and the data scarcity. In the modern world, DL is one of the most present and actively researched areas due to the changes in data availability, understanding, and computer hardware. 'Big data', the availability of large amounts of data, was one important step toward the era of DL. Schermann [126] described big data as an exponential growing field concerning the available data and therefore requires fast and scalable processing.

As AI offers great variability concerning the processing of data, it was evident that this research field will gain more attention over time. Generally, classification approaches can be divided into two groups. The first group includes traditional Machine Learning (ML) methods. These are often referred to as white-box approaches. They include support vector machines, decision trees, and multi-layer perceptrons. All these approaches try to separate the data based on different criteria. Especially the decision trees are very explainable. The same holds for the support vector machine, where a separation line divides the data. For more information about these approaches, the reader is referred to Das et al. [26]. However, for most of these approaches, it is necessary to define features in a pre-processing as good as possible. This leads to the question of which features should be used. While this can be easily done with tabular data, it becomes more challenging with time series. Labrinidis and Jagadish stated in their work that most of the data is not in a suitable format and requires an expensive processing to extract the relevant data [70]. Especially, with the amount of available data that is not in a format ready to analyze, this imposes the question of better solutions that can handle the amount and quality of data in a more appropriate way.

One possibility to avoid this problem is the use of representation-based learning techniques such as neural networks, which learn a feature representation independently. Bengio et al. [11] stated that representation learning is a new perspective and offers the great opportunity to process the data faster without knowledge about it. Especially, in domains that are complex such as signal processing, the extraction of features is important but challenging at the same time. The released formulation of representation-based learning to not require extracted features makes it possible to process such data in a more appropriate way. Another advantage of these methods, especially neural networks, is that they scale very well with large amounts of data. While on small datasets the traditional machine learning shows a superior performance, this is not the case for large amounts of data [50, 148]. Gonzalez-Carvajal and Garrido-Merchan [50] showed that for text classification, the traditional metrics and approaches perform well on small data but are significantly outperformed by deep learning approaches when the data gets larger. Wang et al. compared traditional learning approaches such as Support Vector Machine (SVM) against neural networks and stated that the latter are superior concerning the performance and scaleability. These networks have produced outstanding results in different areas such as the image analysis, text and time series processing. One example of their rise has been shown by Chen et al. [19] in the medical domain. Specifically, in large-scale setups, neural networks have proven to profit from the amount of data and reach superior performances due to the opportunity to train them in a distributed manner. The benefits of the distributed approach are mentioned in the work of Dean et al. [28]. Furthermore, the distributed learning is well suited concerning the increasing availability of data and the development of faster and more efficient graphics processors, leading to more complex networks.

2.2 CHALLENGES FOR DEEP LEARNING IN THE REAL-WORLD

Even though deep learning is outstanding in terms of performance and computational aspects, its applicability in real-world use cases is limited by different challenges. Neural networks benefit enormously from large amounts of data, but these are not always available and even when they are, the data is not always complete. As mentioned by Raghunathan [112] this problem is very present in the healthcare domain. Therefore, it may be necessary to use machine learning approaches to repair the data, e.g. using neural networks as proposed by Gosh and Kristensson [47]. In contrast to the neural network approach, Sudkamp and Hammell [135] presented a rule-based method to complete the data using interpolation. However, it can also happen that the necessary data sets are not available, which means that traditional machine learning approaches retain their importance or data must be generated.

Another important aspect is the user's trust in the software. Lim et al. [77] stated in their paper that building trust is crucial for the success and applicability of deep learning in the medical domain. Concerning the trust, traditional approaches have an advantage over deep neural networks, as they are explainable by design. Depending on the domain, the trust of the user in the decision plays a secondary role or is necessary. In safety-critical areas, the trust of the user is necessary to clarify the question of liability. Besides the user trust, Bibal et al. [12] mentioned that there are legal restrictions concerning the interpretability related to the responsibility and accountability when of neural networks are considered. In safety-critical areas that require an explanation, neural networks can fulfill an assisting task. However, the necessary trust needs to be established using interpretability methods.

Another source of trust is related to the secure processing of data, which plays a major role, especially in the medical field [114]. It is therefore of utmost importance that the data is not passed on to third parties. However, since modern data is huge, it may well be the case that the neural networks are not trained directly at the respective institution. In these cases, the secure transmission of the data gets important as described by Gochhayat et al. [49] and the problem of data leakage through the model parameter needs to be addressed.

2.3 EXPLAINABLE ARTIFICIAL INTELLIGENCE

XAI attempts to explain algorithms in such a manner that their areas of application can be enlarged. One reason for this additional effort is the problematic from a legal and user perspective to making decisions without having a rationale for them. Bibal et al. [12] mentioned that depending on the domain, the regulations are stricter. For example, when public authorities are involved, the legal restrictions require much more interpretability compared to private firms. The same holds for fully automated systems that do not have a human in the loop. According to the authors, the interpretability requirements can be divided into two categories. The first category covers cases without public authorities involved and is mainly based on the General Data Protection Regulation (GDPR) [114]. Therefore, it is required to provide information about the involved logic, decision criteria, and parameters. For use cases that involve public authorities, an administrative and adjudicative decision explanation is required. The category without public authorities requires explaining the motivation behind a decision, and the second category involving public authorities requires being able to explain all factual and legal grounds of the decision. For detailed information on the legal restrictions, the reader is referred to Bibal et al. [12]. Even though the regulations are more recent, researchers have been working on finding regularities and explanations for neural networks since 1990. E.g. Fagen et al. [33] proposed a system in 1980 for a medical task that involved the need for explanations.

To achieve accepted solutions for neural networks, different perspectives and interpretability approaches evolved. The methodologies reach from post-hoc analysis to intrinsic algorithms that directly incorporate interpretability into the reasoning process. A good overview of such methods is provided by Dovsilovic et al. [31]. However, an optimal solution has not yet been found and the evaluation of the individual approaches is not standardized.

Previously mentioned papers also provide evidence that most of the approaches are used and developed in image processing [127]. Their use in time series processing is largely unexplored or questionable and only rarely evaluated. One work that addressed this problem for a subset of interpretability methods is the paper written by Schlegel et al. [127]. Schlegel et al. provided a high-level analysis of attribution methods for time series analysis. Their findings have shown that some methods are not well suited for the characteristics of time series. However, more detailed analysis is needed to better understand the effects of modality, especially since the results indicate that methods are not consistent across model architectures.

Besides the heatmaps, which depend on the gradient and perturbation, another approach is the identification of concepts within the data. Especially in the image domain, the identification of concepts has achieved more attention in the last years. E.g. Yeh et al. [156] recently proposed a new concept-based explanation method. As shown in their paper, the benefit of concept-based explanation compared to the traditional heatmaps is that the concepts describe well-defined shapes or objects. This makes it possible to provide very sharp explanations in the image domain. The downside is that these concepts need to be defined and largely depend on the domain. This limits the applicability of concept-based approaches to domains that provide a meaningful definition of the used concepts. Precisely speaking, the use of concepts in areas outside of image analysis is difficult, which also illustrates that interpretability methods are closely related to their domain.

One other direction is prototype-based explanation, in which the similarity to prototypes provides evidence for or against a class. Li et al. [74] proposed a network architecture based on prototypes and similarity. The prototype-based approaches mainly profit from their intrinsic explainable design. Similar to human reasoning, the comparison of new data to old known data based on similar measurements helps to generate trust. However, the field of prototype-based explanation, has to deal with the performance drop as the introduced decision-making process shows limitations compared to deep learning approaches that only use post-hoc explanations. On the other hand, these approaches achieve a higher level of interpretability when ranked according to the legal restriction levels mentioned by Bibal et al. [12]. One major benefit is that prototypes can be defined for the time series domain, making it possible to adopt this kind of interpretability approach.

Although many approaches and perspectives are present in different domains, there is one major remaining concern. The question of how good an explanation actually is. The difference of perspectives addressed by the methods is a major obstacle, but also the individual application scenario affects the comparability. E.g. Warnecke et al. compared attribution based across several modalities [149] using different measurements. However, while it is straightforward to compare methods within the same category of interpretability method, this gets more complicated when a comparison between categories is needed. Furthermore, some interpretability methods are favored by some metrics as they share the same underlying concept, and therefore a benchmark based on the combination of a method and the metric is not fair. One example is Infidelity and Sensitivity proposed by Yeh et al. [155]. Both metrics are based on perturbations, making it easier for perturbation-based approaches to achieve better results. Another hurdle is the interpretability usage itself, as it is difficult to compare a post-hoc and an intrinsic method. For example, it is possible to compare attribution methods, but intrinsic approaches are less so. Furthermore, the absence of annotated data limits the interpretability of the influence in the prediction and subjective evaluation criteria. This is especially problematic in time series analysis, as subjective evaluation can end up in cherry-picking the best examples or those that strengthen a specific explanation.

Finally, terms such as explainability and interpretability are often not clearly differentiated. Therefore, it is required to define the terminology, as Palacio et al. [106] have done. In their paper, they defined a theoretical framework and the terms to enable a better comparison of interpretability methods. This is especially helpful when the interpretability methods are quite different in terms of their category or application filed. Furthermore, the terms interpretability and explainability are actively interchanged in most existing work. One crucial point mentioned by Palacio et al. is that the model explanation is provided and then interpreted. That means that interpretability is the second step that further depends on the knowledge of the observer, whereas the explanation is not affected by the human.

2.4 PRIVACY-PRESERVING ARTIFICIAL INTELLIGENCE

The main goal of the privacy-preserving research area is to protect models from unwanted attackers. This is not necessary in cases where the data is not sensitive, but it is essential otherwise. In addition to the legal aspects mentioned in [114], it is also necessary to establish trust in the process, since a loss of data can be expensive for a user.

With the rise of neural networks, new attack methods have been discovered. For example, it is possible to reconstruct parts of the data set, which in the case of sensitive data leads to dramatic consequences. Rahman et al. [113] presented the membership attack as one such approach that manipulates the model. They highlighted to which extent it is possible to extract information from an unprotected network. Their analysis indicated that a large amount of information can be reconstructed, and privacy protection is an important field. A second approach is the model inversion attack. Fredrikson [41] showed the capabilities of this approach. The goal of this attack is to recreate the model based on specific queries. It is important to mention that a model always includes data of the dataset, as the parameters are trained using backpropagation. Therefore, stealing a model also provides sensitive information about the dataset and enables the reconstruction of the data. For further attacking mechanism, the reader is referred to the work of Liu et al. [79]. In their paper, they described the different attacking mechanisms. However, it is important to mention that attacks can happen during both the training and the testing stage. Another comprehensive survey paper was written by Mireshghallah et al. [94]. Similar to the previously mentioned work, the authors elaborated the perspectives of attacks. They further shaded light on the direct information exposure related to the data center and clouds involved in the process.

To limit the indirect information expose, it is mandatory to introduce privacy assuring mechanisms, as presented by Zhu et al. [163]. An important aspect raised in their paper are the properties that need to be fulfilled to come up with a successful safe approach. However, the application of a privacy-preserving techniques may lead to stability and performance issues which needs to be addressed. On the other hand, a certain performance loss is expected as the privacy-preserving methods mostly use noise or reduce the dataset size with a distributed learning to achieve the privacy. Both approaches, intuitively, contribute negative to the stability of the network.

One of the most successful approaches is the use of differential privacy. Abadi et al. [1] showed that this technique for deep learning can provide private models that do not expose the data during the inference or other stages. Throughout the training of a differential private model, the gradients that flow through the model are clipped and noise is introduced. This noise produces a variance or uncertainty which leads to a model that cannot expose the precise training data. However, as this approach introduces noise to the training process, it reduces the performance of the model.

Another approach to get rid of the noise problem is the use of data from different sources. Hao et al. [53] presented the successful application of federated learning with a twofold effect. First, because of the different data sources and possible variances, the model can achieve a better generalization and second, due to the aggregation of the gradients the data-specific information gets lost. The latter one results in privacy, as it is not possible to reverse the aggregation process to get information about the individual data.

A third approach that can potentially be combined with previously mentioned techniques is encryption. Fang et al. [34] presented the use of homomorphic encryption in a federated setup. The benefit of the encryption is that no noise is required to achieve a private behavior as encrypted data is saved. However, this comes at the cost of computation resources. Fully encrypted networks and encrypted training suffer from large overhead and are in some cases unfeasible.

2.5 DELIMITATION FROM OTHER MODALITIES

To understand the need of this work, it is important to consider the difference between the modalities. Most of the above-mentioned work is aligned to work with image or textual data. However, this does not mean that these approaches work on time series. Below, some data characteristics are given and discussed, providing information about the extent to which it is possible to transfer approaches from one domain to another.

2.5.1 *Image Domain*

As most of the work is designed to work with image data, it is pivotal to understand whether this work can be applied to time series or not. In the case of attribution methods, such as those evaluated by Nielsen et al. [101] for image data, it is possible to use most of them on the time series. However, when the number of channels or time steps increases, it gets problematic to use heatmaps. In the image domain except for domain-specific images, the number of channels is fixed to either one or three. Additionally, the interaction between the channels is defined as they always have the information related to the color encoding. This is a major difference compared to the time series, where the channels mainly relate to sensors. These sensors can have different meanings and maybe interact with each other. Ultimately, their number is not limited, and it is important to show only relevant information. Another problem is the definition of concepts within the time series domain. Concepts, as used by Yeh et al. [156], they are obviously defined for the image domain but not for the time series domain. This is mainly because a human is as opposed to a lot of visual data. A third difference is the impact of noise. Noisy explanations in the image domain are less problematic compared to the time series ones. The reason originates from the concepts defined in the image domain. Intuitively, a human can handle noisy attribution maps on images better compared to the time series domain. The last characteristic that is not shared between the image and time series data is localization. While it is common that objects close to each other are related in the image domain, this is not true for the time series. In the time series domain, it can be the case that values in the early time steps do not affect their neighbors, but a relation to time steps far later exists.

2.5.2 *Natural Language*

The second modality is the textual domain, for which the heatmap creation works fine. E.g. LIME proposed by Ribeiro et al. [115] is an attribution approach that uses a surrogate model to highlight important parts in the data. An adaption of this approach is available and usable in the time series domain. Concerning the concept definition, the natural language domain offers a defined set of rules and the individual terms can be categorized. For example, it is possible to categorize them by their sentiment. To do so, embeddings are trained as described by Li and Yang [76]. Embeddings define the relation between the different terms and therefore help to cluster or group similar meanings. This makes it possible to extract concepts for the natural language domain which are understandable to a human. Furthermore, noise in the natural language domain is similar to the image domain and less problematic, as the meaning of each token is known by a human. Finally, in case of the localization, the natural language domain shares a similar behavior with the time series to some extent.

3 RELATED WORK

This chapter provides detailed insights on the existing state of research, both in the context of interpretable and privacy preserved time series analysis. Section 3.1 explains the different perspectives and existing methods that can be used in this field to address the interpretability of artificial intelligence approaches. Section 3.2 provides the related work concerning privacy methods, their advantages and limitations. In the remainder of this paper, these related works are assumed to be known.

3.1 STATE-OF-THE-ART INTERPRETABILITY METHODS

Interpretable approaches can be classified into two main categories [25], which are post-hoc and intrinsic. Post-hoc techniques aim to interpret the decisions and actions of a model without modifying the model itself, hence the term 'post-hoc'. On the other hand, intrinsic techniques focus on designing models in a way that allows for insights into their decision-making process at the design time. There are additional characteristics to further divide both categories, such as their scope, and methodology. The scope addresses whether the method points out a single instance or provides explanations based on the complete dataset. While a global behavior may sound better, the instance-based explanations are directly related to the sample and usually provide more details. Finally, the methodology can be divided into the perturbation- and gradient-based methods. An overview of the categories is given in Figure 3.1.

It has to be mentioned that among the existing work there is no superior approach, as each method faces limitations regarding the quality, subjectivity [78], the audience, and the domain usage. Since this paper only discusses approaches applicable to time series, the reader is referred to the individual original papers or the review by Rojat et al. [117] for further details.

3.1.1 *Post-hoc*

The largest category of post-hoc covers attribution methods. These techniques are mainly instance-based and produce maps that highlight the parts relevant to the decision. Over time, these methods became more complex to overcome some of their limitations and can be divided into gradient- or perturbation-based. Gradient-based methods need access to the internal parameters of the models, so that gradient propagation can be used to interpret the model's decision. In contrast to that, perturbation-based approaches are less model dependent and do not need access to the gradients. Perturbation-based methods slightly change the input to create an importance ranking. However, that is one of their weaknesses, as the fidelity of these models is lower. In contrast to that, the gradient-based

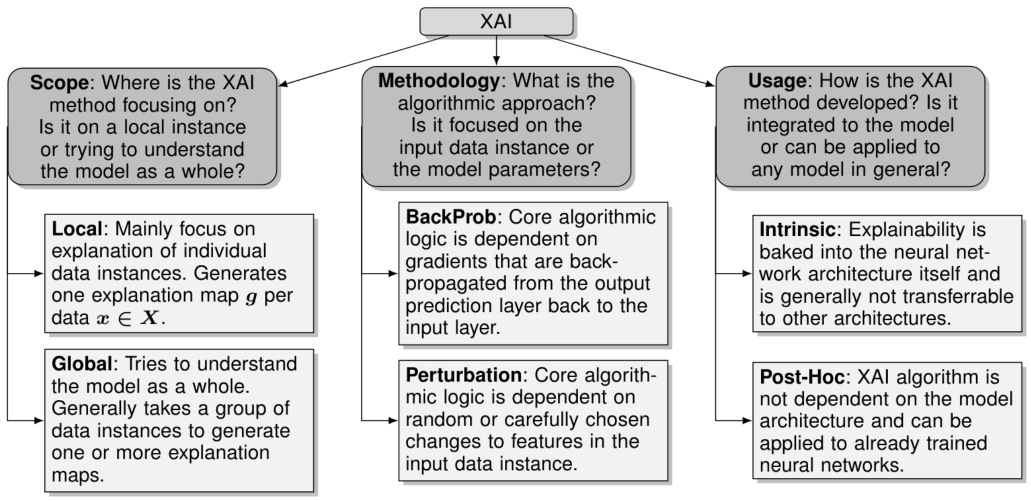

XAI
Scope: Where is the XAI method focusing on? Is it on a local instance or trying to understand the model as a whole?
Methodology: What is the algorithmic approach? Is it focused on the input data instance or the model parameters?
Usage: How is the XAI method developed? Is it integrated to the model or can be applied to any model in general?
Local: Mainly focus on explanation of individual data instances. Generates one explanation map g per data $x \in X$.
Global: Tries to understand the model as a whole. Generally takes a group of data instances to generate one or more explanation maps.
BackProb: Core algorithmic logic is dependent on gradients that are back-propagated from the output prediction layer back to the input layer.
Perturbation: Core algorithmic logic is dependent on random or carefully chosen changes to features in the input data instance.
Intrinsic: Explainability is baked into the neural network architecture itself and is generally not transferrable to other architectures.
Post-Hoc: XAI algorithm is not dependent on the model architecture and can be applied to already trained neural networks.

Figure 3.1: Shows the different interpretability perspectives and the characteristics. Reprinted from [25].

approaches require specific model architectures as they are applied directly to the model parameters to generate an explanation. Besides the attribution methods, influence functions and compression mechanisms can be used as post-hoc methods to analyze the data and prediction. They explain the input data in a way that provides insights into the dataset. Below are the advantages and disadvantages of different post-hoc methods.

Simonyan et al. [133] proposed the *Saliency* as an interpretability method based on gradients in 2013. This method uses the backward pass to compute the gradients of a trained network. The gradients provide the information in which direction the parameter needs to be adjusted to increase the confidence of the model for a certain input. While the gradients were mainly used during the training process, Simonyan showed that they can be used to generate a map that explains the model decision. One advantage of their methods is that there is no bias from a perturbation, and therefore the produced heatmap only includes the actual behavior of the network. However, the map is likely to be noisy as it is based on the gradients, making it difficult to understand the map depending on the dataset. Although *Saliency* is known to not produce the best results, it is often used as a baseline approach to highlight the feasibility as it does not require any hyperparameter.

In 2014 *Guided-backpropagation* was introduced by Springenberg et al. [134]. This method is works to some extent similar to the *Saliency*, however, the way the backward pass is handled differs. In this approach, the values below zero are not propagated and are set to zero. This removes negatively influencing features from the heatmap and focuses only on those that contribute positively to the attribution and the prediction. Focusing only on positively influencing data produces a less dense representation of the attribution map. However, it also provides a false impression that the negative values might be of relevance for the prediction.

As an improvement of *Saliency*, *Input X Gradient* was proposed by Shrikuma et al. [131]. To improve the *Saliency* the authors proposed to multiply the gradients with the input. The main motivation behind their approach origins from the noisy attribution maps created by the *Saliency*. Using the input and gradients in conjunction ensures that the noise of the gradients gets reduced as it is multiplied by the input. Therefore, a feature that is not really present in the input but has a high gradient will be less attributed compared to one with the same gradient but a higher input value. One problem with this method is that the gradient function considers one small step and might look entirely different after a few steps. However, this drawback applies to all gradient-based methods.

Another prominent method to produce heatmaps is the *Integrated Gradients* approach proposed by Sundararajan et al. [136]. This method is based on the gradients but requires multiple calls. The idea is to compute a path from a so-called baseline to the actual sample using the integral. Therefore, a single or multiple baselines can be used. The authors suggested that the baseline in the imaging modality can be an image containing only zeros. In the time series domain, the same baseline can be used if the time series is normalized with a mean of zero. According to the authors, the main improvements of their methods are that the implementation invariance and sensitivity are given. They defined the implementation invariance such that two networks that produce the same outputs should have the same attribution maps independent of their implementation. Sensitivity states that when a feature changes output, the feature should have a value greater than zero. In addition, the subtraction of the baseline gradient serves as a smoothing algorithm for the attribution maps. However, it is crucial to understand the baseline of the data, as this may result in an entirely different explanation.

Ribeiro et al. [115] proposed *LIME* in the year 2016. This approach uses a surrogate model to compute the impact of perturbations in the input and their effect on the output. Based on neighborhood samples, the surrogate model learns to mimic the behavior of the original model. This makes it possible to produce smoother attribution maps. However, the drawback is that the surrogate model as it is interpretable and less complex might not be able to cover all the decisions of the black-box classifier. In addition, it requires a certain number of samples to be trained on, and depending on this amount, its quality might be worse. Although this trade-off between fidelity and interpretability is problematic, the results and applicability of *LIME* are good. Especially, as it is applicable to other models that do not use a backward pass, such as support vector machines and decision trees.

The idea of *Shapley Values* originates from the game theory, but is a well-known concept for the interpretability of neural networks. Castro et al. [15] published their work to efficiently compute *Shapley Values*. The computation of these values is based on the idea of perturbing the values and adding them sequentially to the baseline to inspect the impact on the output. Another variant is to add the features independently to get better information about their individual impact on the prediction. This procedure can also be extended using masks to not add a single feature as its impact might not be well represented or insignificant. However, the mask definition and the number of randomly performed perturbations used in this process can distort the attribution. Ultimately, computing every possible combination of features is not feasible as the runtime to do so would be unfeasible depending on the data.

Lundberg and Lee [80] proposed an approach to estimate the previously mentioned *Shapley Values* in 2017. While the *Shapley Values* provided valuable results, Lundenberg et al. correctly identified that it is required to compute them more efficiently to enable a larger number of samples and produce more stable attribution maps. Therefore, they used *LIME* [115] as a surrogate model and set the loss function and weighting kernel accordingly. In most cases, the surrogate model, as it is a white-box model, is much smaller than the model it describes. Based, on the correctness of the *LIME* model, the predictions are equivalent to the original model, and therefore it is possible to query this model instead of the original model for the shapely values.

Another approach to efficiently compute the *Shapley Values* was mentioned in a paper written by Lundenberg et al. [80]. The idea was to compute the values using *DeepLift* proposed by Shrikumar et al. [131]. Therefore, the attributions are computed based on the *DeepLift* algorithm and averaged across the baselines to estimate the *Shapley Values*. *DeepLift* decomposes the contribution of each feature concerning the output. Therefore, it uses a reference activation of each neuron and compares the actual value to the reference to compute a difference, which then highlights the impact. While the motivation of this approach is the same, this version is applied directly to the model and does not involve a surrogate model that potentially can lead to wrong attributions. However, this version has more limitations concerning the architecture, as not all layers are suitable.

Feature Ablation in contrast to the *Shapley Values* is a more straightforward approach that requires less computation power. Zeiler and Fergus [159] proposed this approach to compute the impact on the output when a feature is removed. While it is possible to sequentially remove a single data point, it is often beneficial to use a feature mask and remove multiple features at the same time because of possible feature interactions. To remove the features, *Feature Ablation* replaces the value with a defined baseline value. Usually, the quality of the attribution depends on the correct baseline. Furthermore, as it uses forward passes, a sufficient but feasible number of these is required.

Feature Occlusion [159] is very similar to the *Feature Ablation* as it replaces the values using a similar principle. The difference is that while the *Feature Ablation* removes individual features, the *Feature Occlusion* uses a sliding window to replace every contiguous region. This results in smoother maps as the impact of the adjacent data points is considered. However, this also reduces the fidelity, as adjacent points can produce wrong importance values due to the windowed approach. Usually, the *Feature Ablation* and the *Feature Occlusion* produce very similar maps.

Feature Permutation [159] is the third approach proposed by Zeiler and Fergus. This technique differs from the previous two as it does not remove the features, but perturbs them. This has the advantage that the sample cannot drop out of the distribution, which can be the case for the previous two approaches. However, the approach suffers from the same computational difficulties as the other perturbation-based approaches, as the number of forward passes need to be sufficient. In addition, the correct perturbation is required as otherwise, the attribution might not be correct. A common approach is to perturb the values by a given percentage or add random noise on top of the features.

One major drawback of the *Feature Occlusion* comes from the definition of the window size. In real-world cases, it is not possible to know the perfect window size and features that can interact with each other. To overcome both problems, Petsiuk et al. [110] proposed *RISE*. The authors use a set of masks that define areas to occlude and compute the prediction for the input multiplied by the mask that occludes parts of the data. The prediction scores are then used to serve as weight scores for the attribution, which is achieved by the multiplication of the occlusion masks with the prediction scores. One major difference compared to the *Occlusion* approach is that the set of masks is created randomly on a down scaled input shape. In a second step, these masks are upscaled, resulting in patches instead of individual values that are occluded. Throughout the experiments, the authors have shown that the performance of this approach is superior to the simple occlusion, but it consumes more resources as the number of masks and forward passes increases with the data shape.

Dynamic Masks proposed by Crabbe and Van der Schar [22] is a very recently proposed time series related attribution method. The approach is based on the famous excremental perturbation masks proposed by Fong et al. [39]. The idea is to learn a perturbation map by optimization, and in addition relax the importance of binary values. Although this might come up with lower fidelity, it lowers the cognitive effort to understand an attribution map in the time series context significantly. However, the drawback of their approach is that the number of masks and the target density of the mask needs to be defined. Defining the number of relevant features beforehand might not be suitable in a real-case scenario. Furthermore, the maps show significant differences when the target ratio of important features is not set correctly. Theoretically, it is possible to test different values, but that is computationally costly.

Ko and Liang [66] proposed an approach to weighting dataset samples and their effect on the actual classifier. This approach helps to understand which samples are the reason for a classifier to support or decline a decision. The novelty comes from the fact that using the *Influence Functions* is not required to retrain the model to understand how each sample affected the parameter. To do so, the authors proposed to use the gradients and the hessian product vectors. Furthermore, this approach can be used to debug datasets as it reveals the samples that are mislabeled. However, concerning interpretability, the most important fact is that using the values it is possible to get insights about the concepts or relevant features the model learned as these are represented with higher influence.

Another approach that weights the samples and produces scores to show their impact was proposed by Yeh et al. [157]. The *Representer Point* was presented as an improvement of the *Influence Functions* approach based on the preactivation prediction. The preactivation is decomposed into a linear combination that captures the impact of each data point. Thus, as mentioned previously, it is possible to debug and explain the decisions of the network. Although this is done post-hoc, it differs from the post-hoc method in the way the explanation is given and mainly addresses the debugging of the model and dataset.

The compression of the input to see its effect is rarely studied, although it comes with several advantages. Palacio et al. [105] proposed an approach to see what the network requires to predict the input. Their approach first trains an autoencoder based on the dataset used to train of the classifier, and then attaches it to the trained classifier. While the classifier is not modified, the autoencoder is fine-tuned in a way that it compresses the data as much as possible and only reconstructs the data required for the classifier to achieve the best possible prediction. One of the benefits of this approach is that the compressed input only contains data relevant to the task, which lowers the cognitive effort, in addition, it removes possible confusing parts from the data. However, this method requires training the autoencoder and model, which might be time and resource intense.

3.1.2 *Intrinsic*

Intrinsic methods are designed in a way that accepts a trade-off in accuracy, but therefore provides white-box decisions compared to the black-box predictions of traditional neural networks. One commonly used intrinsic approach is to create prototypes, as this technique is closely related to human thinking. The comparison between new and already known prototypes is intuitive and efficient. In the second category of approaches, the divide and conquer principle is applied: To explain the overall prediction, the sample is divided into smaller parts, resulting in a better understanding of individual influence.

Using prototypes to understand the decision process is a principle that is related to human thinking. Especially, in the image domain, it is straightforward to define prototypes for objects that share some properties. The resulting explanation is intuitive for a human and provides trust. Li et al. [74] proposed such a network architecture. They used a prototype layer to learn artificial prototypes, which are then used to compute the similarity between the input and themselves. To learn a proper prototype, only the latent representation of the input is given to the prototype layer. The prototypes are then learned to use the same latent dimension. A decoder is used to transform them back to the input dimension for inspection. Finally, a head using a fully connected layer is used to produce the final output based on the prototypes. Throughout their analysis, Li et al. have shown that the accuracy drops only by a few percent, but the model is interpretable without the need for a post-hoc interpretability method. However, it has to be mentioned that their approach requires hyperparameter tuning of the different loss terms to ensure interpretability. Another prototype-based approach was presented by Gee et al. [45]. They adopted the idea of the previously mentioned approach to the time series. This enabled them to produce global prototypes for sequences. In addition to domain changes, they included a so-called diversity score that ensures that the learned prototypes differ, as otherwise the prototypes could look similar due to dataset biases such as the distribution.

The work of Chen et al. [17] extended the idea of using prototypes. In their work, the prototypes were not learned as global representatives, but served as prototypes for specific parts of the input. This makes it possible to cover concepts that are shared across the classes and further improves the interpretability as a prototype consists of a single patch of the data instead of the whole data. Another benefit of this prototype approach is that it learns the concepts without the need for additional annotations. In their work, they have shown that the network architecture only shows minor changes in accuracy. As their approach does not use an autoencoder network to project back the prototypes, they push the prototypes towards existing samples in the dataset and use the most similar one as a representative patch for the prototype. Another work that demonstrated that the processing of data with patches can be an efficient way to produce interpretable results was written by Hou et al. [55]. The idea behind the patchwise approach in their case originated from the large data, as the images are huge and processing them requires large resources. Furthermore, the features that affect the prediction are local, and using a divide and conquer strategy helps to locate them. To achieve this, the authors suggested splitting the data into patches and training a network on these patches in a way that the network learns to classify the patches and does not consider patches that are class-independent. In a second step, they use a classifier to compute the overall label of the input.

3.2 STATE-OF-THE-ART PRIVACY-PRESERVING METHODS

Privacy-Preserving Machine Learning (PPML) aims to preserve data privacy, but comes with some drawbacks in terms of XAI [103] and computational costs. The field of privacy-preserving methods covers plenty of different perspectives that address several security aspects. However, the below-mentioned work is limited to those that address the model of security and does not cover all privacy factors mentioned by Liu et al. [79] such as a secure connection and storage. Therefore, it is relevant to identify the correct defense mechanism for every attack to protect the data [20]. According to Mireshghallah et al. [94], the privacy can be divided into direct threats related to the previously mentioned data connection and storage security and the indirect threats. Indirect threats are those that are inferred based on the model and can be avoided with privacy-preserving models. Direct treats are not related to the deep learning. For further information not covered in this work, the reader is referred to the work of Zhang et al. [161] which presents a wide variety of existing privacy methods or Tanuwidjaja et al. [141]. As this work focuses on the model security, direct attack aspects are not included.

3.2.1 *Attack Mechanisms*

According to Mireshghallah et al. [94], it is possible to identify up to five different attack categories during the inference stage. However, it is worth noting that it is also possible to attack the model during training through data injection, corruption, or modification. Below, the different indirect attack perspectives are listed and shown in Figure 3.2.

The membership attack tries to infer data based on multiple queries. Shokri et al. [130] proposed this attack mechanism. The target is to get information on whether a specific sample is in the original training data or not. This can be achieved with queries through the trained model. Therefore, the attack trains a model that identifies the membership of samples based on the confidence of the target model. Furthermore, it is possible to combine this attack with a model inversion to not only get information about the membership, but further mimic the model.

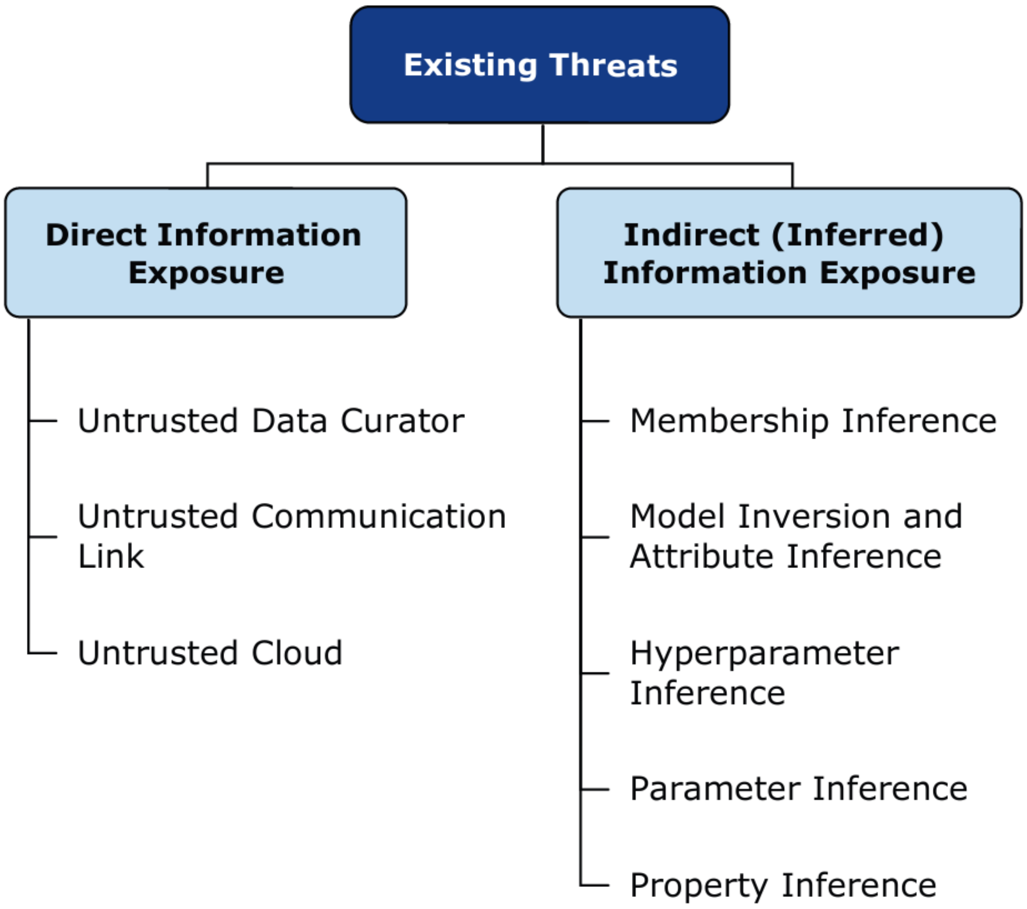

Existing Threats
Direct Information Exposure
Indirect (Inferred) Information Exposure
Untrusted Data Curator
Untrusted Communication Link
Untrusted Cloud
Membership Inference
Model Inversion and Attribute Inference
Hyperparameter Inference
Parameter Inference
Property Inference

Figure 3.2: Shows the different attacking mechanisms applicable to deep learning models. Reprinted from [94].

Another approach is the model inversion attack proposed by Fredrikson et al. [42]. The goal of this attack is to create a model that is similar to the target model. To do so, the previously mentioned membership attack can be used to identify candidate samples used to train the new model. Using statistical measurements, it is possible to weight different attributes and samples to get a model that is close to the target model. While this attack is best suited for black-box attacks, it is also possible to attack white-box models. Furthermore, Fredrikson et al. [41] explained the process to reconstruct the samples with the model inversion attack.

Going one step further, it is possible to reconstruct the model architecture using attacking mechanisms. Yan et al. [152] proposed a mechanism to infer the model architecture. To do so, they use different mathematical formulations and query the target model. Based on the confidence scores they obtain from the model, the proposed approach can precisely, determine the used hyperparameters such as channel, the filter number, and filter size.

Carlini et al. [14] proposed an approach to extract dataset patterns as they are memorized by the network and not protected by default. Furthermore, the authors defined mathematical formulas to measure the degree of memorization and data exposure.

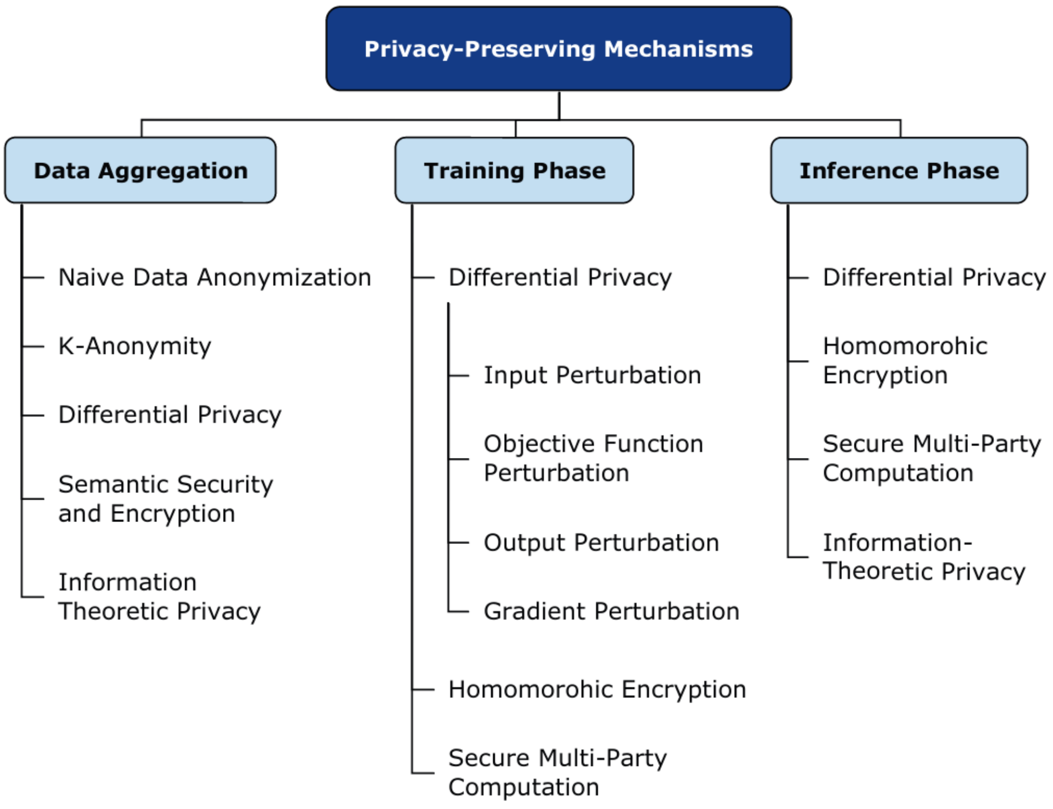


Figure 3.3: Shows the different defense mechanisms applicable to deep learning models. Reprinted from [94].

3.2.2 *Defense Mechanisms*

To prevent attackers from stealing information, it is required to protect the model, as it includes valuable information about the data. In general, as stated by Mireshghallah et al. [94] it is possible to differ between privacy-preserving and privacy-enhancing mechanisms. The first group tries to prevent the model from memorizing and exposing data. The second group can be used to enhance the privacy further. E.g. splitting the data and computing the average over the data removes some properties that are sample specific. Privacy-enhancing methods are not mandatory designed for privacy. Within the privacy-preserving approaches it can be differentiated between those that aggregate data, training and inference protection. The different categories of defense mechanisms are shown in Figure 3.3.

Among the data aggregation mechanisms, the anonymization [30] is one of the most commonly used approaches. While it is intuitive to anonymize data by deleting or changing attributes, there are advanced approaches such as the *K-anonymity* proposed by Sweeney et al. [137]. This approach ensures that the sample is similar to at least k-1 neighbors concerning a given attribute.

Another method is the Differential Privacy (DP) which in addition provides training and inference protection. DP is one of the most used and widespread solutions to tackle privacy restrictions in the context of deep learning. In 2016, Abadi et al. published their work on this topic [1]. The authors proposed an efficient way to compute the gradients of a network using differential privacy. The proposed differential Stochastic Gradient Descent (SGD) clips the gradients during the training to introduce privacy. After the gradient clipping, noise is added to the gradients to further improve the privacy. This procedure ensures that less private information is leaked during the training process. Other DP methods aim at perturbing the input [44], output or optimization objective [16] of the model.

Furthermore, Abadi et al. [1] mentioned the epsilon privacy as a budget for the training process to track the information leaked through the network. In addition, Renyi Differential Privacy (RDP) was proposed by Zhu and Wang [164] as a privacy budget calculation for advanced sampling strategies. Renyi Differential Privacy (RDP) precisely computes the privacy for the Poisson sampling without replacement and is computationally efficient.

Another very famous approach among the differential private methods is *PATE* proposed by Papernot et al. [107]. *PATE* describes a framework with a teacher and a student network. A private teacher network is trained and then predicts queries on public data. The public data and teacher predictions are then used to train a student network. The idea is that the teacher can have the information about the private data, whereas the student does not see the private data.

Federated learning initially was invented to enable distributed training involving multiple parties, and therefore it is a privacy enhancing method [153]. McMahan et al. [81] proposed an approach to enable the training using an averaging approach. To set up the federated technique, a copy of the model which is held globally is distributed to each participant. Next, a specific number of iterations using the data is executed locally, and the gradients are forwarded to the global model. However, before the gradients are applied to the global model, they are averaged across the different parties. By design, the averaging introduces privacy, as individual private information gets discarded during the average step if enough participants are available. The authors specifically mentioned that unbalanced datasets, the number of clients, and communication stability play an important role in success.

Homomorphic Encryption (HE) is a privacy preserving training and inference algorithm. It can be applied to the data and the model depending on the use case. In 2014 Yi et al. [158] proposed HE. This kind of encryption enables specific mathematical operations to work on the encrypted data. For example, it is theoretically possible to compute gradients using the encrypted values. However, since homomorphic encryption involves a high computational cost, various strategies have been developed to apply the encryption. Aono et al. [6] proposed an approach that encrypts the weights for the update on the local model in a federated scenario and uses encrypted weights on the global model to preserve privacy. This approach shows a successful combination of encryption and federated learning. According to Naehrig et al. [100], although recently many approaches including encryption evolved, it is still questionable if the application is feasible in real case scenarios due to the disadvantages.

Secure Multi-party Computation (SMC) is a hardware software solution used to create a safe computation during the training and inference stage. One of the possible solutions to achieve this is the *SecureNN* framework proposed by Wagh et al. [147] which splits into secret parts that are uploaded to different servers. All the splits need to be available to train the model. For detailed information about the hardware setup and the corresponding protocols, the reader is referred to the paper of Wagh et al. [147].

For completeness, Trusted Execution Environments (TEEs) proposed by Sabt et al. [120] are mentioned. TEE describes a specific hardware solution to ensure a secure data processing and is part of the privacy enhancing category.

3.2.3 *Synthetic Data Generation*

In contrast to the previously mentioned privacy approaches that are applied directly to the data to perform the target task, the data generation offers new perspectives. Initially, the idea is not correlated to the private domain. E.g. Pu et al. [111] proposed one of the most promising network architectures to generate data and the corresponding labels, namely the Variational AutoEncoder (VAE). VAEs are a modified version of the traditional autoencoders to generate data and preserve specific properties of the latent space. The motivation behind such approaches is that data scarcity can be reduced if synthetic data that looks similar to real data can be produced.

Besides the autoencoder-based approach, the so-called Generative Adversarial Network (GAN) was proposed by Goodfellow [51] in 2014 and has proven to be one of the best approaches to generate data. The proposed network architecture consists of a generator and a discriminator. The generator produces samples based on a vector within the latent space, and the discriminator tries to identify whether a presented sample is synthetic or not. During the optimization, both parties try to fool each other. Both the autoencoders and generative models can produce data and over the years many modifications of these networks originated, e.g. the Wasserstein Variational Autoencoder (WAE) proposed by Tolstikhin et al. [143]. This modification uses a different loss that results in a better representation of the latent space and the samples projected into that space.

Based on the generative models, Xie et al. [151] proposed a differential private version of the Generative Adversarial Network (GAN) that uses the Differential Privacy (DP) algorithm. The resulting generator can then be used to generate data that can be used without the privacy restrictions, making it possible to apply existing deep learning structures to the synthetic data and transfer the results to the private data. Recently, Chen et al. [18] further investigated in the same direction and came up with an approach that protects only the generator. They found that this leads to better results as the discriminator gets stronger. In addition, removing the privacy constraints from the discriminator does not lower the privacy, as this part of the network is not shared with anyone and is discarded after the training is finished.

4 TIMEFRAME: INTERPRETABLE AND PRIVACY-PRESERVING DEEP LEARNING

This chapter provides detailed insights about *TimeFrame*, the framework proposed in this thesis. First, the need for the system is addressed in Section 4.1 and next, the main parts of the system consist of are introduced, including their importance for the actual use of the proposed framework. Detailed information about the components can be found in Part II to Part V

4.1 NEED OF THE SYSTEM

GDPR [114] and AI act defines and regulates the development and deployment of AI systems in the EU. This includes guidelines for ethical and trustworthy AI. In addition, it requires having human oversight in the development and use of these systems by transparency and privacy. On the other hand, deep neural networks have recently achieved super human performance in several areas. However, these models are not conforming to AI act and GDPR and neither transparent nor privacy preserved. This becomes even more critical to the models which are developed to deal with time series data. The regulation defines several rules that need to be fulfilled by a system to allow the use in certain domains. This regulation is one of the major challenges that needs to be addressed as it is crucial to extend the use of neural networks which perform well on a variety of data. With the fulfillment of the GDPR, the use of neural networks in safety-critical domains such as healthcare and finance can be permitted. As it is already known that neural networks are one of the top competitors when it comes to performance, this can have a huge impact e.g. assisting doctors can improve the health system. As a step towards such an improvement, the proposed framework addresses the rules included in the GDPR and aligns them with the processing of time series using deep neural networks. Although there already exist several methods that are applied in research to different classification problems, there is no real framework that offers a comprehensive set of tools for time series analysis. Furthermore, as mentioned in previous chapters, most of the research focused on the image domain, which makes it even more challenging to come up with a solution that addresses the constraints raised by the GDPR to use deep learning in the time series domain.

Besides the capabilities of the system to combine interpretability and privacy, it also offers the possibility to understand the interaction of those aspects. This aspect was not evaluated so far for time series, but it is crucial to have a framework that makes it possible to identify the biases that occur if the two properties are combined. It is possible to explore the interaction of the two contrary goals, or to use components that isolate the two properties in a way that no interaction biases the results. Ultimately, this provides a solution to use private but explainable classifiers for the time series analysis. However, to come up with such a flexible

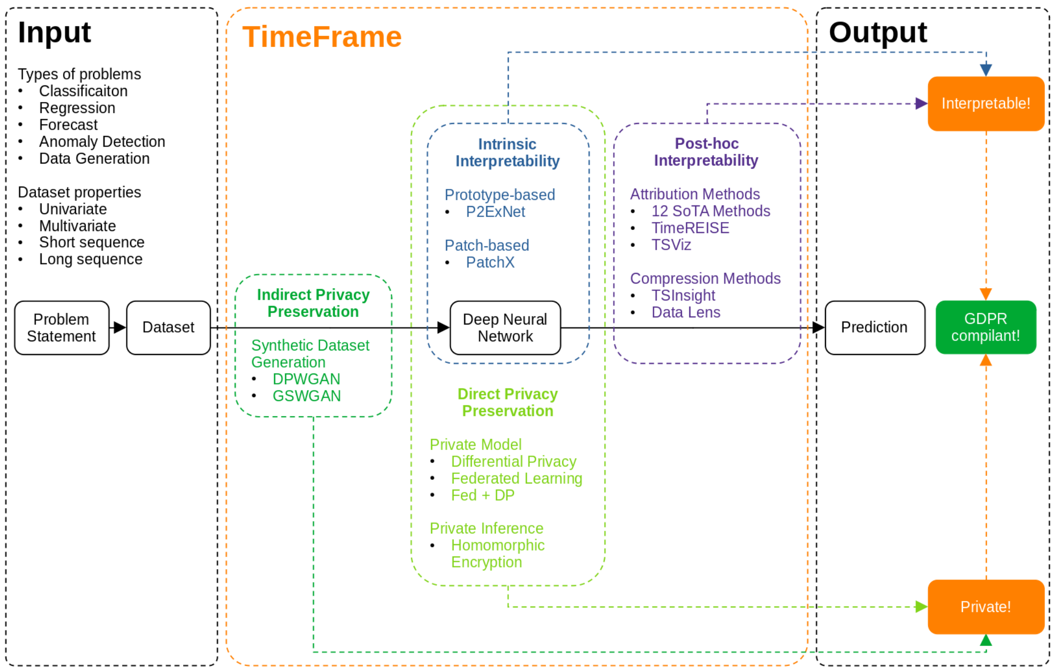


Figure 4.1: Shows the TimeFrame framework and its components. The framework offers a large set of existing and novel interpretability and privacy methods that can be used to create a GDPR conformed neural network.

framework that can be used with different architecture, and data, and yield good performance while still conforming the GDPR rules requires a separation of the challenges into fundamental parts which are explained in Section 4.2

4.2 COMPONENTS OF THE PROPOSED FRAMEWORK

The whole system can be divided into two parts. These parts are interpretability and privacy. A system that is used in the real-world requires both of them, however, their goals are contrary. Interpretability tries to reveal as much as possible, while privacy tries to hide everything. The following subsections cover each of the parts individually, and a third subsection covers their interaction and how they are combined to come up as one system that is capable of dealing with sensitive data.

4.2.1 *Interpretability Components*

The interpretability component is one of the main parts, as it is crucial to understand the decisions made by a network. Furthermore, as per GDPR and AI act guidelines, accountability is an important part. However, it is not possible to hold accountable without interpretability. The framework provides a wide variety of components which can be used to comply with deep neural network-based time series analysis systems with the regulations. To enable existing highly performing models to comply with the regulations, the framework is well-equipped with several post-hoc methods that work on a trained neural network. Concerning the post-hoc methods, there are 12 state-of-the-art methods and four novel components. The benefit of those is that they do not depend on the network architecture, making them very versatile. Two different perspectives are addressed. First, the relevance of the input features and their contribution to the output can be visualized. Second, the suppression of input data reduces the amount of data seen by the network and relevant to the task. Similarly, the framework also offers two novel intrinsic methods, which can be used to design interpretable time series analysis systems. These intrinsic methods offer a better explanation according to the GDPR as the reasoning itself is explainable, but their drawback is the architecture dependence. The details of these components are discussed in Part II and Part III. One of them is based on prototypes and the other method uses the divide and conquer principle. The reasoning using prototypes is closely related to human thinking, as humans try to extract concepts and compare them to already known ones. Similarly, if a human faces a complex problem, one approach is to break it down into smaller subtasks which are done by the second component. This results in a hierarchical solution that is easier to understand.

4.2.2 *Privacy Components*

In addition, the framework offers a set of privacy preserving methods as even if a network is interpretable, it is not possible to use it in a safety-critical environment as long as it is not privacy-preserving. Deep neural networks by default expose a lot of data, as mentioned in the previous part. It is possible to mimic the network, reconstruct input data, and poison those networks. None of these behaviors is acceptable if sensitive data is processed. However, introducing privacy always lowers the performance, which might be problematic in some cases. The proposed framework, therefore, has two categories of privacy that are addressed. First, the direct privacy in which the privacy mechanisms are applied to the classification network, and second the indirect privacy in which the sensitive data is never exposed to the classification network. The first component consists of four different direct privacy methods to modify the training and inference process. However, using the direct privacy approaches changes the explanation of the model. To overcome this problem, it is possible to apply indirect privacy. The second component offers two methods to create a private generative model. The data created using this model can then be used without any restriction. This has several benefits, such as a larger amount of data, no privacy on the classifier, and more generalized models.

4.2.3 *Interaction of Components*

In Figure 4.1 the framework is visualized. Based on a problem and the corresponding data as a basis, the two main aspects are interpretability and privacy. Depending upon the nature of the problem, and the state of the system, the framework provides a wide range of options to select the most suitable combination. For each subcategory, components are provided that can be selected. E.g. It is possible to build an interpretable system using *TimeREISE* as the interpretability method. If in addition the *GSWGAN* [18] as a privacy component is used, then the system results in a classifier that approaches the interpretability and privacy constraints of the GDPR.

It is also possible to use multiple methods to understand their agreement or disagreement, which can be of interest. Therefore, the proposed methodologies within the post-hoc category can be used in parallel or in conjunction, depending on the need. Combining post-hoc and intrinsic methods is possible, although it does not increase the interpretability, as the intrinsic methods are explainable by design. The same holds for the privacy component, while it is possible to combine federated learning and differential privacy within the direct privacy category, it does not make sense to apply a direct approach to the classifier of the indirect privacy category. Ultimately, the performance is important although, it has no impact on GDPR compliance. However, it is an important aspect when the components are selected, e.g. post-hoc methods do not lower the performance whereas intrinsic methods do. Intuitively, all privacy methods lower the performance, and depending on the data, the selection of a good combination of methods is crucial.

Part II

POST-HOC INTERPRETABILITY

Post-hoc interpretability is a widely used category of interpretability methods. However, most times the approaches have been evaluated in the image domain and their performance in the time series domain is untested. This makes it difficult to use them in a real-world context, as it is unknown whether their explanation is correct and interpretable. To address these issues, several different aspects were evaluated.

First, a comprehensive benchmark for attribution methods is presented. As one of the first attribution benchmarks in this domain, it shows the characteristics of the existing well-known attribution maps and further highlights the limitations and possible adoptions. Based on this evaluation, two novel techniques are proposed. *TimeREISE* and *TSViz* are two novel approaches that produce attribution maps for the time series domain. While *TimeREISE* covers the perturbation-based perspective, *TSViz* uses a gradient-based approach. In addition, for *TSViz* a complete service including a front-end was developed.

Second, *TsInsight* as a novel compression method was developed. The compression-based approach reduces the input data to the minimum required by the network. The core idea of this approach is to discard the cognitive intense parts that are not used by the classifier and expose only the parts relevant for the network in a shape similar to the original input.

Third, a benchmark for influence functions was conducted. The analysis highlights to which extent it is possible to infer information about the dataset and model biases. Therefore, the impact of each data sample on the prediction is evaluated.

5 TIME TO FOCUS: BENCHMARKING STATE-OF-THE-ART ATTRIBUTION APPROACHES

Post-hoc interpretability is very rarely evaluated in the context of time series. This section addresses the lack of comprehensive benchmarks for the group of attribution methods. The experiments are centered around aspects such as the applicability and effectiveness of attribution methods in time series analysis. Furthermore, they cover the strengths and weaknesses of these methods. Specifically, a runtime analysis highlights the limitations for real-time use cases.

The presented experiments involve gradient-based and perturbation-based attribution methods. A detailed analysis indicated that perturbation-based approaches are superior concerning the Sensitivity and occlusion game. These methods tend to produce explanations with higher Continuity. Contrarily, the gradient-based techniques are superb in runtime and Infidelity. In addition, a validation of the parameter dependence of the methods, feasible application domains, and individual characteristics is attached. The findings accentuate that choosing the best suited attribution method strongly depends on use case.

5.1 DATASETS

For the experiments, a subset of the datasets from UEA & UCR [9] repositories was used. The selected datasets cover different aspects such as a variance in the number of channels, sequence length, classes, and task. The tasks include point anomaly and sequence anomaly classification. Furthermore, the datasets cover traditional sequence classification not related to atypical behavior. These datasets correspond to different critical domains that require interpretability and privacy. In addition, to the UEA & UCR datasets, the point anomaly dataset proposed by Mercier et al. [91] was included. This dataset is unique compared to the others because a perturbation on single points can change the complete prediction. Table 5.1 lists the different datasets used to evaluate the attribution methods.

5.2 EXPERIMENT & RESULTS

In this subsection, different aspects of the above-mentioned methods are presented. The attribution techniques were not optimized to ensure fairness between approaches because fine-tuning requires assumptions about the data set. The experiments cover the following aspects: Impact on the accuracy, Infidelity [155], Sensitivity [155], runtime, the correlation between the methods,

This chapter is an adapted version of the work presented in: D. Mercier. et al. "Time to Focus: A Comprehensive Benchmark using Time Series Attribution Methods." In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, INSTICC. SciTePress, 2022, pp. 562–573. DOI: 10.5220/0010904400003116.

Table 5.1: Shows the used UEA & UCR Datasets related to critical infrastructures.

Domain & Dataset	Train	Test	Steps	Channels	Classes
Communications					
UWaveGestureLibraryAll	896	3,582	945	1	8
Critical manufacturing					
Anomaly [91]	35,000	15,000	50	3	2
FordA	3,601	1,320	500	1	2
Public health					
ECG5000	500	4,500	140	1	5
FaceDetection	5,890	3,524	62	144	2
Telecommunications					
CharacterTrajectories	1,422	1,436	182	3	20

Table 5.2: Shows the AlexNet [67] architecture used in this paper. Dropout layers are excluded from the table. The padding of every layer was set to 'same'. The variables 'c', 'w', and 'r' depend on the input channels, width, and the number of classes of the used dataset.

Name	Type	In	Out	Size	Stride
conv_1	Conv, ReLu, Batch	c	96	11	4
pool_1	MaxPool	96	96	3	2
conv_2	Conv, ReLu, Batch	96	256	5	1
pool_2	MaxPool	256	256	3	2
conv_3	Conv, Relu, Batch	256	384	3	1
conv_4	Conv, Relu, Batch	384	384	1	1
conv_5	Conv, Relu, Batch	384	256	1	1
pool_2	MaxPool	256	256	3	2
dense_1	Dense, ReLu	w * 256	4,096		
dense_2	Dense, ReLu	4,096	4,096		
dense_3	Dense	4,096	r		

and impact of label and model parameter randomization. In existing work such as [3, 56, 101] these measurements are judged as significant.

In general, all experiments were executed for the previously mentioned datasets. However, identical results were removed due to the low number of insights they provide to the reader. The pre-processing of the data covered a standardization to achieve a mean of zero and a standard deviation. AlexNet [67] was modified to work with one dimensional data and trained using an SGD optimizer and a learning rate of 0.01 to evaluate the different attribution techniques. In Table 5.2 the network structure of the AlexNet is shown. All networks were trained for a maximum of 100 epochs, and the learning rate was halved after a plateau.

Table 5.3: Evaluation of the test data using the original split provided by the datasets. Subset covers the performance of on 100 samples that were used for the remaining experiments due to the computational limitations. Values show the weighted F1 scores and provide evidence that the sets are similar.

Dataset	**Test Set [%]**	**100 Samples Set [%]**
Anomaly	98.01	94.64
CharacterTrajectories	99.30	100.00
ECG5000	93.52	89.07
FaceDetection	59.56	70.97
FordA	92.04	94.00
UWaveGestureLibraryAll	93.18	98.02

In the particular case of label permutation, the labels of the training data were randomized.

Due to the immense computational effort, a set of 100 test samples was selected to evaluate all methods. In Table 5.3 the weighted F1 scores are shown. The differences between the original data and the subsets are less than 5%. Only the *FaceDetection* dataset shows a difference of 19%.

5.2.1 *Impact on the Accuracy*

To evaluate the performance of the attribution methods, the drop in accuracy under the addition and occlusion of the data points was inspected. To occlude the data, the points were set to zero, as this is the mean of the data corresponding to the baseline. Respectively, the start point was zero when adding points stepwise. Figure 5.1 shows that most of the methods were able to correctly identify the most influential data points. Intuitively, data points that had a higher impact on accuracy should be ranked better. The top row shows the accuracy increase, adding the most significant points stepwise. The bottom row shows the behavior of adding the insignificant data points first. Ultimately, reading each plot starting from 100 to 0 percent results in excluding the least important ones for the top row and the most important ones for the bottom row. The experiments highlight that for most datasets, namely *Anomaly*, *CharacterTrajectories*, *ECG5000*, and *UWaveGestureLibraryAll*, a few data points were enough to recover the accuracy. Surprisingly, adding unimportant data points using *LIME*, *Saliency*, and *Dynamask* resulted in higher accuracy values for the *ECG5000*, *FordA*, and *UWaveGestureLibraryAll* datasets. *Saliency* has shown to suffer from the noisy backpropagation, while the drawbacks of *LIME* and *Dynamask* are their hyperparameters.

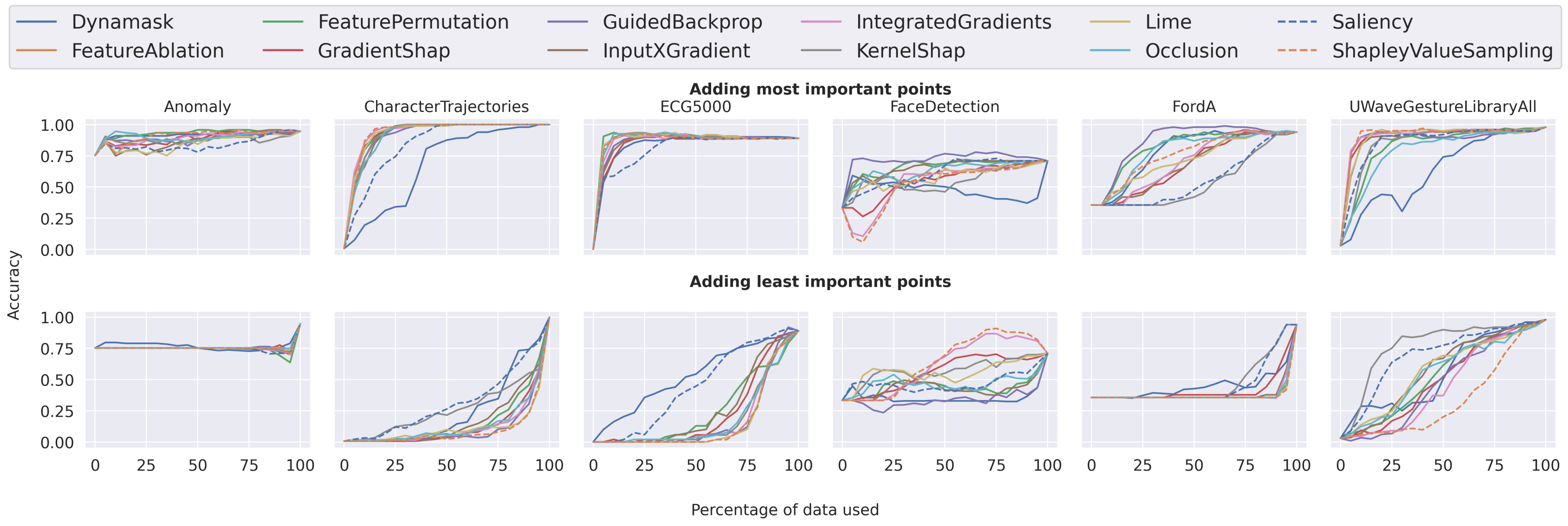


Figure 5.1: Shows the impact when adding points to the baseline signal based on the attribution scores (Top: most important points, bottom: least important points). The values show the weighted F1 scores.

5.2.2 *Prediction Agreement*

In addition to the accuracy drops, the agreement with the original data was computed. Therefore, in Table 5.4 the percentage of data required to produce a similar prediction as with the original sample is shown. To do so, data points were included stepwise based on their importance. Initially, all data samples started with zeros. In every step, the next most important data point was added. The results indicate that the required data for an agreement of 90% of the predictions was in most cases reached with far less than 50% of the data. The results further show that the perturbation-based approaches overall performed better. In addition, the required amount of data highly differed based on the dataset. Intuitively, *Dynamask* did not perform well on this task, as it provides only a binary decision on whether a feature is significant or not. Besides *Dynamask*, *Saliency* and *KernelShap* showed a worse performance too. On the other side, the *Feature Ablation*, *Feature Permutation*, *Guided-backpropagation*, and *ShapleyValueSampling* approaches showed superior performance to the other methods using the data suggested being important by those methods resulting in a much earlier agreement of the prediction.

Table 5.4: Evaluation of how many data points are required to reach a specific agreement between the original and modified input. All numbers are in percentage. Lower values are better as fewer data was needed to restore the ground truth predictions. The numbers in each cell show the percentage of data points added to the baseline to achieve the required agreement concerning the prediction. Perturbation-based approaches show a significantly better performance.

Method	**Anomaly**			**CharacterTraj**			**ECG5000**			**FaceDetection**			**FordA**			**UWaveGesture**		
Req. Agreement in [%]	90	95	100	90	95	100	90	95	100	90	95	100	90	95	100	90	95	100
Gradient-based																		
GradientShap [80]	**1**	44	97	**15**	18	32	15	20	75	60	71	98	69	77	**96**	**12**	38	100
GuidedBackprop. [134]	**1**	76	98	17	27	45	13	**14**	83	**2**	**2**	**5**	**33**	61	98	**11**	**16**	100
InputXGradient [131]	**1**	51	92	16	21	**29**	18	24	**42**	26	36	55	69	81	98	**12**	38	100
IntegratedGradients [136]	**1**	**3**	99	**12**	**15**	31	11	18	**38**	63	81	97	70	79	98	**12**	39	100
Saliency [133]	**1**	76	97	34	41	48	32	37	75	48	51	54	88	93	100	20	53	100
Perturbation-based																		
Dynamask [22]	**1**	5	100	55	72	92	18	31	100	100	100	100	50	71	100	61	74	**98**
FeatureAblation [159]	**1**	**2**	**48**	**15**	20	**28**	**6**	**9**	60	**25**	**30**	**35**	**44**	**52**	**82**	26	55	**99**
FeaturePermutation [38]	**1**	**2**	**48**	**15**	20	**28**	**6**	**9**	60	**25**	**30**	**35**	**44**	**52**	**82**	26	55	**99**
Occlusion [159]	**1**	3	83	19	20	**29**	9	15	**46**	**16**	47	87	43	**55**	**96**	33	68	100
Miscellaneous																		
KernelShap [80]	**1**	58	100	**15**	22	43	**8**	15	84	70	84	99	90	94	98	16	34	100
LIME [115]	**1**	90	100	**15**	**17**	49	**8**	17	75	49	52	81	79	86	99	13	**17**	100
ShapleyValueSampling [96]	**1**	30	**51**	**12**	**13**	30	10	18	71	68	90	93	65	79	97	**9**	**15**	100

Table 5.5: Computed values show the average Infidelity (lower is better) over the 100 sample subsets. Results show differences between the different methods when applied to time series data. No category shows a superior performance, although the gradient-based approaches were slightly better.

Method	Anomaly	CharacterTraj.	ECG5000	FaceDetection	FordA	UWaveGesture
Gradient-based						
GradientShap	2.3803	**1.1408**	**0.7897**	0.0014	1.3734	11.4717
GuidedBackprop	2.4057	1.1665	0.8060	0.0014	1.3782	11.6886
InputXGradient	**2.3056**	**1.1475**	0.8135	0.0014	1.3854	11.5830
IntegratedGradients	2.3594	1.2064	0.8260	**0.0013**	**1.3537**	**11.3763**
Saliency	2.3788	**1.0921**	0.8174	0.0014	**1.3636**	11.7546
Perturbation-based						
Dynamask	2.4382	1.2650	0.8271	**0.0013**	1.3806	11.6034
FeatureAblation	2.3859	1.1513	0.8459	0.0014	1.3869	11.5511
FeaturePermutation	2.4015	1.1654	**0.7949**	0.0014	1.3991	11.5112
Occlusion	**2.3430**	1.2078	0.8107	0.0014	1.3752	**11.3569**
Miscellaneous						
KernelShap	2.4115	1.1802	0.8288	0.0014	1.3785	11.6568
LIME	2.4259	1.1584	**0.8040**	0.0014	**1.3732**	11.6323
ShapleyValueSampling	**2.3352**	1.1671	0.8153	0.0014	1.3745	**11.4625**

5.2.3 *Infidelity & Sensitivity*

The Infidelity [155] provides information about the change concerning the predictor function when perturbations to the input are applied. The metric derives from the completeness property of well-known attribution methods and is used to evaluate the quality of an attribution method. Table 5.5 shows the Infidelity mean error using 100 perturbed samples for each approach. A lower Infidelity value corresponds to a better attribution. The results indicate that the tested methods do differ by a large margin of less than 7.2% on average, and in addition, the Infidelity values strongly depend on the dataset. Neither the gradient-based approaches nor the perturbation-based were superior. The mean increase between the worst performing and the best method was 7.2%. During the experiments, the highest increase was measured for the *CharacterTrajectories* dataset (15.8%) and the lowest for the *FordA* dataset (3.4%).

Further, the Sensitivity [155] of the methods for a single sample was compared. Computationally, the Sensitivity is much more expensive but provides a good idea about the change in the attribution when the input is perturbed. Using the Sensitivity, the robustness against the methods concerning noise was evaluated. Table 5.6 shows the Sensitivity for all methods. The results show that *Dynamask* had a Sensitivity of zero, as *Dynamask* intentionally forces the importance values to be either one or zero. Although this is a benefit concerning the Sensitivity, it results in a drawback when ranking the features as shown in the accuracy drop experiment. In addition, perturbation-based approaches have shown 30.9% better results on average concerning their Sensitivity across all datasets. The most significant difference between the attribution methods (42.1%) occurred using the *FordA* dataset, while the lowest scores (26.1%) were observed using the *CharacterTrajectories* dataset. Besides, the impressive performance of *Dynamask*,

Table 5.6: Computed values show the Sensitivity (lower is better) of a sample. Results show larger values for *LIME* and *Shap*-based approaches. Overall, the performance of the perturbation-based techniques was superior.

Method	**Anomaly**	**CharacterTraj.**	**ECG5000**	**FaceDetection**	**FordA**	**UWaveGesture**
Gradient-based						
GradientShap	0.9364	0.6610	0.9149	0.9764	1.0369	1.0347
GuidedBackprop	0.1324	0.1531	0.0562	0.1339	**0.0398**	0.2057
InputXGradient	0.1890	0.1017	0.0709	0.0952	0.0924	0.1927
IntegratedGradients	0.1166	0.1144	0.0458	**0.0419**	0.0906	0.2086
Saliency	0.1902	0.1126	0.1841	0.0995	0.0762	0.2220
Perturbation-based						
Dynamask	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
FeatureAblation	**0.0414**	**0.0360**	**0.0350**	0.0581	0.0463	**0.0444**
FeaturePermutation	**0.0414**	**0.0360**	**0.0350**	0.0581	0.0463	**0.0444**
Occlusion	0.0645	**0.0167**	**0.0305**	**0.0506**	**0.0254**	**0.0352**
Miscellaneous						
KernelShap	1.0908	0.9405	0.2162	0.9248	0.8876	1.0283
LIME	0.8221	0.4986	0.1408	1.5613	0.6974	0.6378
ShapleyValueSampling	0.9132	0.3917	0.1852	0.5938	0.5536	0.3458

the *Occlusion*, *Feature Ablation*, and *Feature Permutation* showed robustness against permutations.

5.2.4 *Runtime*

The runtime and resource consumption are important. Although the availability of resources increases, they are not unlimited. Depending on the throughput of the approach, real-time interpretability can be possible. For mobile devices, the computation capacity is limited, and low resource dependencies are beneficial. To compare the methods in terms of their computational effort, execution time for a single sample of each dataset was computed. Figure 5.2 shows that especially the simple gradient-based methods like *Saliency*, *Integrated Gradients*, and *Input X Gradient* showed a low computation time. On the other side, methods like *KernelShap* and *ShapleyValueSampling* showed increased time consumption. During the analysis, the default values suggested in the corresponding papers of the methods were used. In the case of the *FaceDetection* dataset, the computational overhead of the *Feature Ablation*, *Feature Permutation*, and *Occlusion* increased significantly as they strongly depend on the number of features. The *FaceDetection* dataset needed 41 times longer than the *Anomaly* dataset. Overall, the computation time of the *FaceDeteciton* dataset was four times longer than the aggregated computation of all others. The characteristics of the *FaceDetection* dataset favors methods that are independent of the number of features. The high number of channels and time steps when every data feature gets evaluated separately increases up to an unacceptable point. In addition, it has to be mentioned that only 100 iterations instead of the default 1,000 for each optimization of *Dynamask* were used to lower the computation times.

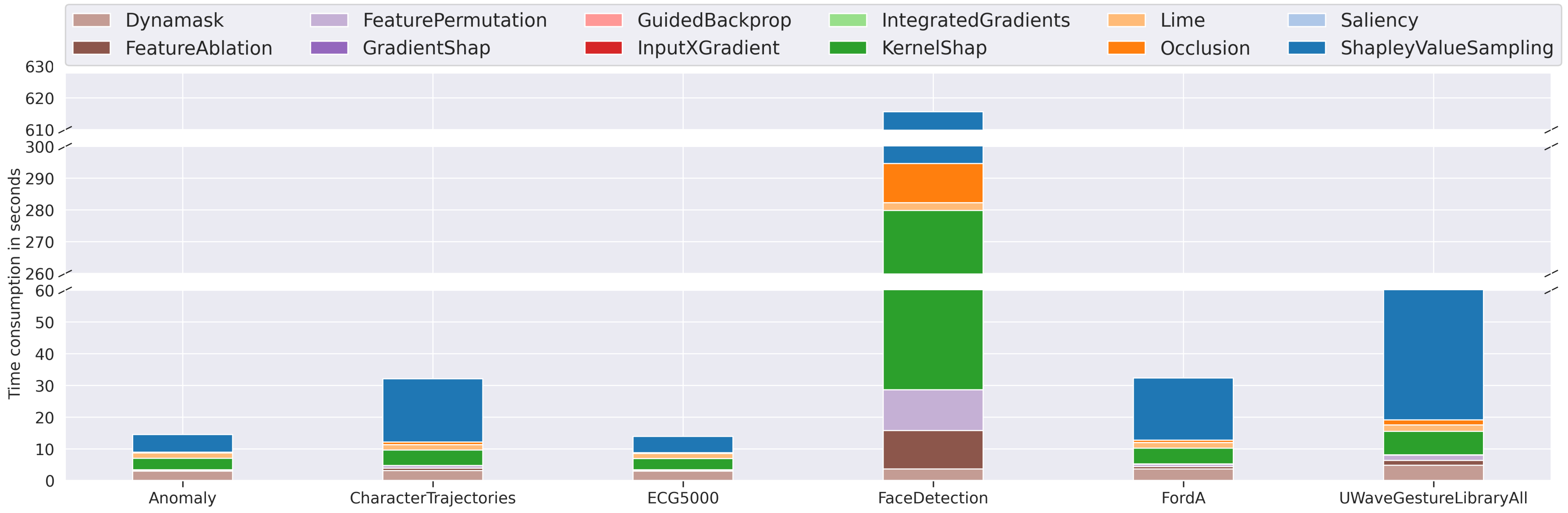
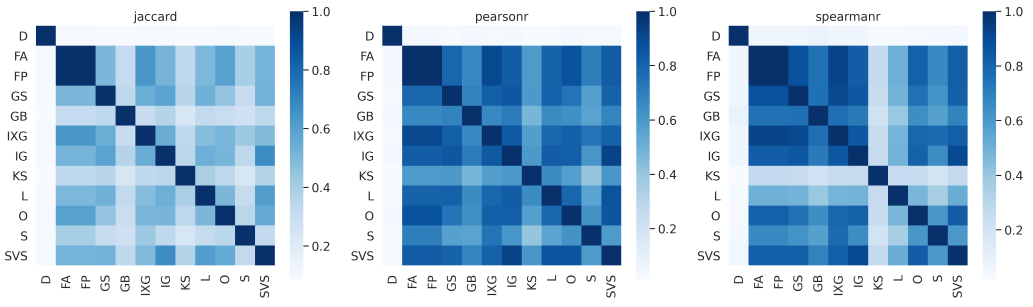
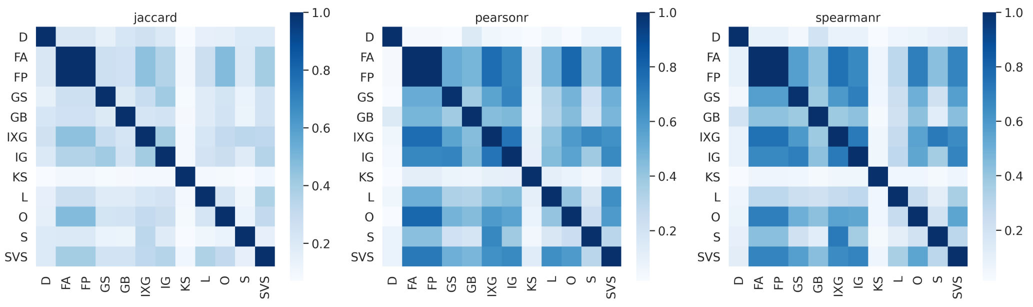


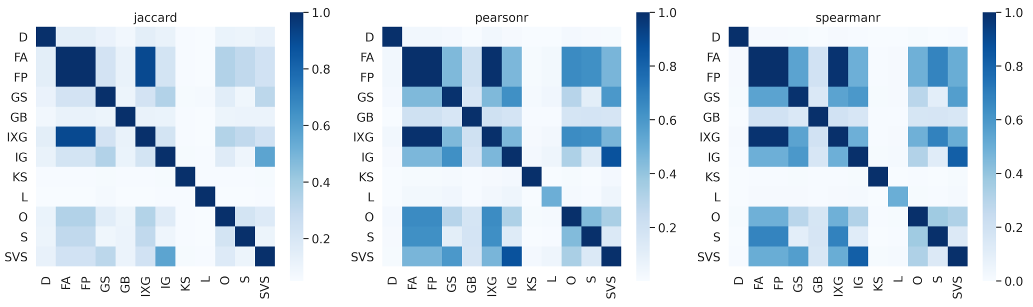
Figure 5.2: Shows the time spent to compute the attribution of a single sample. Note that some bars are not visible due to their fast computation time compared to the other methods, and the time of *Dynamask* is lowered by parameter optimization due to the otherwise unsuitable time consumption. Hardware: Quad-Core Intel Xeon processor, Nvidia GeForce GTX 1080 Ti, and 64 GB memory.



(a) CharacterTrajectories



(b) FordA



(c) FaceDetection

Figure 5.3: Shows the average correlation/similarity of over 100 attributions. The ten percent most important features were selected for the Jaccard Similarity. The method names are shortened using only the capital characters (*Dynamask*, *Feature Ablation*, *Feature Permutation*, *Gradient Shap*, *Guided Backpropagation*, *Input X Gradients*, *Kernel Shap*, *Lime*, *Occlusion*, *Saliency*, *Shapley Value Sampling*). *KernelShap* showed a significantly lower correlation to other methods compared to all others. *Feature Ablation* and *Feature Permutation* showed a high correlation.

The results strongly suggest that this does not change the overall behavior of *Dynamask* but lowered the computational time by a factor of ten. Using the default 1,000 iterations would not be suitable in any case, as the computation time would increase by a factor of ten.

5.2.5 *Attribution Correlation*

Another aspect is the correlation of the different attribution maps. Therefore, different correlation measurements were used, namely the Pearson

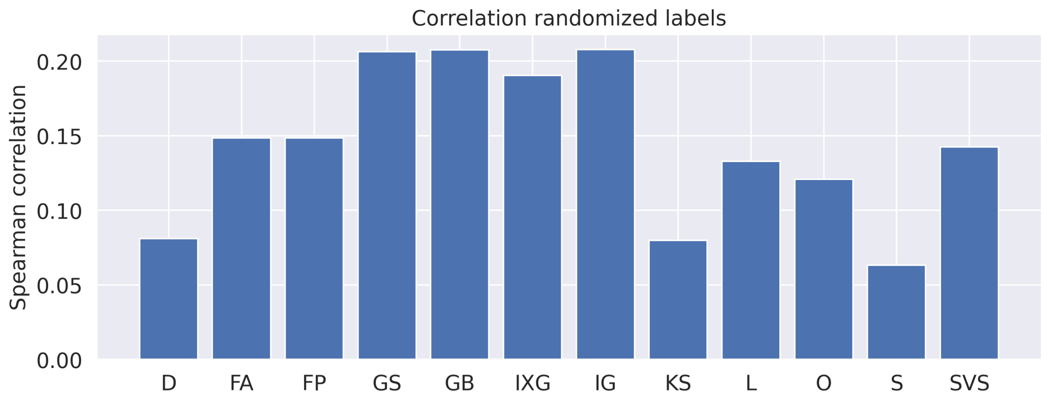


Figure 5.4: Shows the Spearman Correlation (rank correlation) of the attribution methods evaluated on the same model architecture using randomized training labels using the *CharacterTrajectories* dataset. The method names are shortened using only the capital characters (*Dynamask*, *Feature Ablation*, *Feature Permutation*, *Gradient Shap*, *Guided Backpropagation*, *Input X Gradients*, *Kernel Shap*, *Lime*, *Occlusion*, *Saliency*, *Shapley Value Sampling*). *Dynamask*, *KernelShap*, and *Saliency* showed a significantly lower dataset dependence.

Correlation [10], Spearman Correlation [98] and Jaccard Similarity [102]. The Pearson Correlation measures the correlation between two series concerning their values. Spearman Correlation is a ranked measurement that compares the ranks for each of the features. Finally, the Jaccard Similarity is used as a set-based measurement. During this experiment, the similarity of the attributions computed over the 100 test sample subsets was evaluated. Ultimately, only the important points matter concerning a correct attribution. To consider that, percentile subsets of the important features were selected for the Jaccard Similarity to understand the agreement of the methods concerning those features.

Figure 5.3 shows the results for the correlation and similarity. Overall, each metric shows a similar behavior. *Feature Ablation* (FA) and *Feature Permutation* (FP) were very similar. In addition, the *Dynamask* (D) approach and *KernelShap* (KS) were different from any of the others. For *Dynamask* this can be explained with the binary decision whether a feature is significant or not. Intuitively, this should result in a high similarity for the Jaccard Similarity. However, this is not the case as the attribution of *Dynamask* has an internal smoothing which includes less important features to preserve a continuous mask. Furthermore, *LIME* (L) and *KernelShap* (KS) were less similar to the other approaches.

5.2.6 *Dependency on Model Parameter*

Attribution methods should depend on the model parameter and the labels of the data. Therefore, the impact of label permutation and parameter randomization of the model was evaluated. This section only shows the results using the *CharacterTrajectories* dataset, as the results on the other datasets are similar.

The idea of the label permutation is that attribution methods should depend heavily on the labels. High values of correlation emphasize that the attribution depends on the data characteristics rather than the concept learned by the model.

The models were trained similar to the baseline model on the same training data but permuted the labels. This permutation results in a model that does not generalize well but learns to replicate the training set. In addition, this approach does not require the validation dataset. Intuitively, the train accuracies of those models are good. Nevertheless, they fail on the test set. Precisely speaking, these models show no label dependence. Figure 5.4 highlights that the correlation drops to values between 0.05 and 0.2. Based on the overall low correlation, the attribution methods highly depend on the labels rather than dataset characteristics.

In addition to the label permutation, layers of a correctly trained network were systematically randomized to understand the dependency concerning the model parameters. To comprehend the impact of the layers, each layer was randomized independently. Further, the model was randomized starting from the bottom to the top and vice-versa. The results in Figure 5.5 show all three approaches. Interestingly, the correlation of *Guided-backpropagation* stayed high when randomizing the top layers but significantly dropped when randomizing the bottom layers. Randomizing the upper layers, the correlation of *Guided-backpropagation* was close to the original attribution map, whereas the correlation of the other methods dropped by 0.5 or more. That suggests that this method highly depends on the first few layers. In addition, the results show that for all attribution techniques, a single randomized layer was enough to get an attribution that was no longer related to the original map, which is a desired property. The top-to-bottom randomization further shows that except for the *Dynamsk* approach, the correlation continuously got smaller when more layers were randomized. Finally, the bottom-to-top randomization highlights that the first layer of the network was enough to produce attribution maps that were not related to the original.

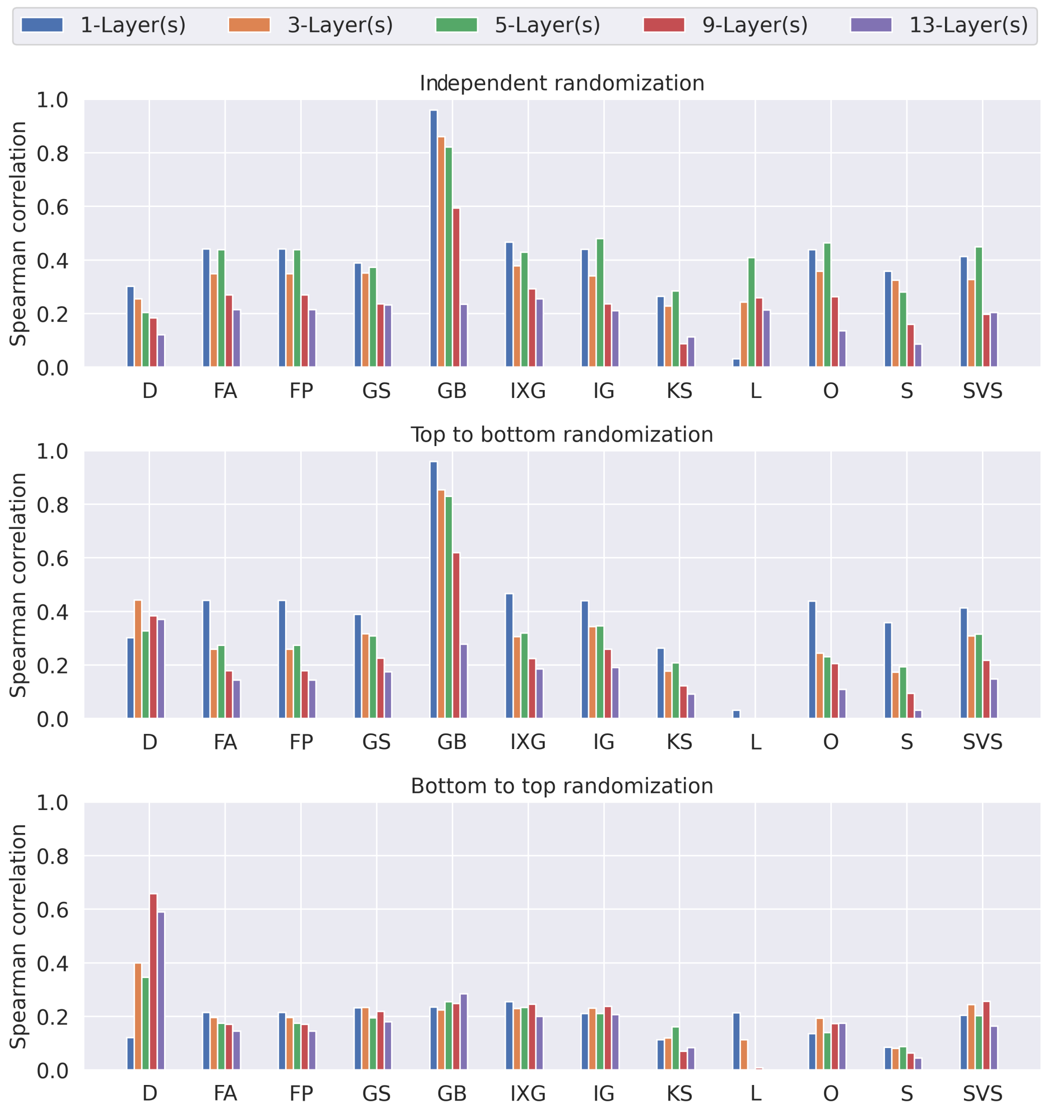


Figure 5.5: Shows the Spearman Correlation of the attribution methods evaluated on the trained model and randomized layer weights using the *CharacterTrajectories* dataset. Weights were either randomized for each layer independently, from top to the bottom layer, or vice versa. Only layers with trainable parameters (convolutional, batch norm, dense) were included when counting the number of randomized layers. The method names are shortened using only the capital characters (*Dynamask*, *Feature Ablation*, *Feature Permutation*, *Gradient Shap*, *Guided Backpropagation*, *Input X Gradients*, *Kernel Shap*, *Lime*, *Occlusion*, *Saliency*, *Shapley Value Sampling*). *Guided-backpropagation* showed significant correlations when only the upper layers were randomized. The correlation of all other methods dropped significantly.

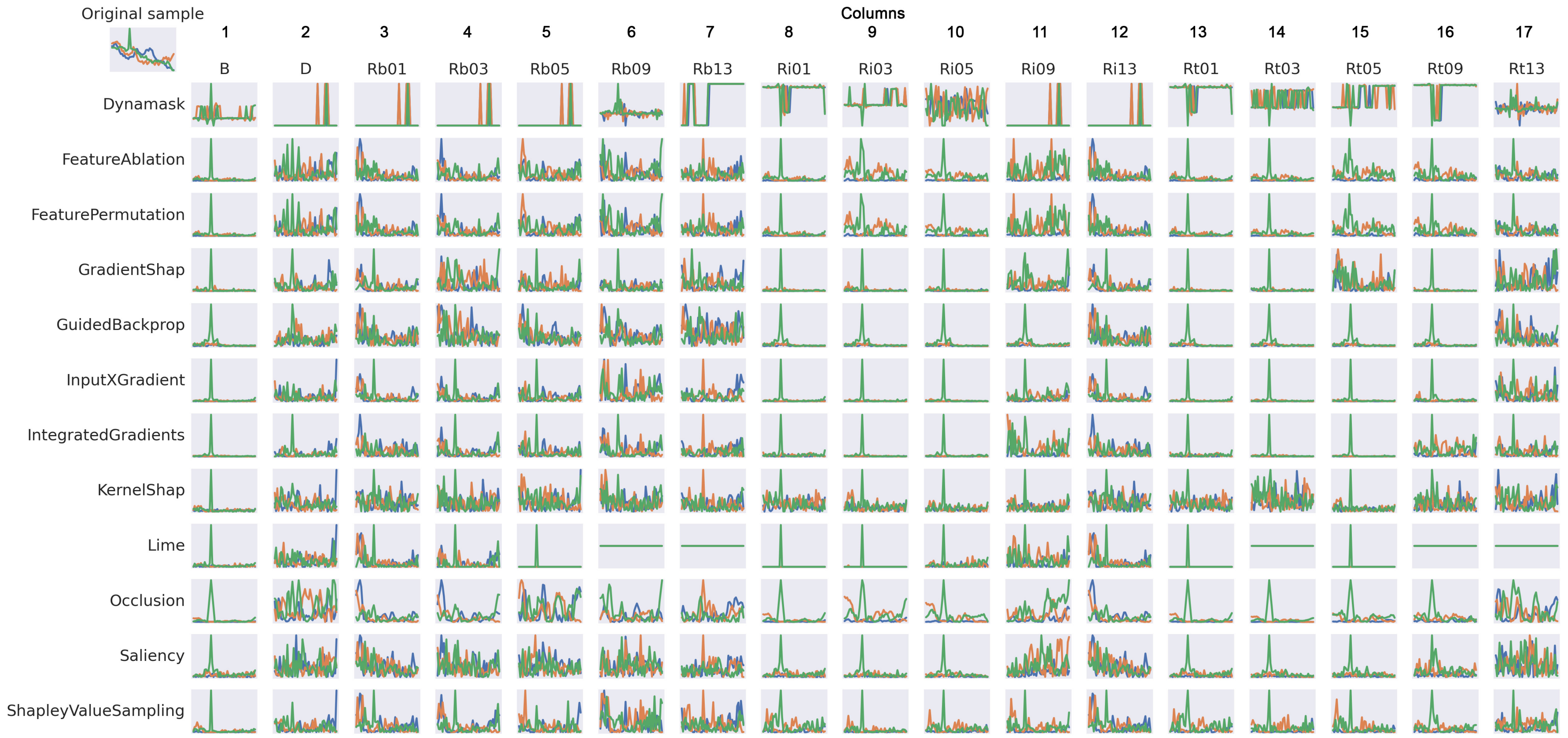


Figure 5.6: Shows all attributions for a selected anomaly sample. The important part is the peak of the sample. ’Ri’, ’Rb’, ’Rt’, ’D’, and ’B’ correspond to the independent, bottom to top, top to bottom randomization, label randomization, and original attribution map. Only convolutional, batch norm, and dense layers are counted. Changing the data labels during training significantly lowered the performance of all approaches except *Integrated Gradients* for the *Anomaly* dataset. Overall, randomizing lower layers resulted in much more noise compared to randomization in the upper layers.

5.2.7 *Visual Attribution Comparison*

Figure 5.6 shows all computed attribution maps for a reference sample. Due to interpretability reasons, an anomalous instance of the anomaly dataset was selected. The example in the top-left corner contains a single anomaly in one channel that is important for the classification. The rest of the figure shows the different attribution maps and the impact of randomization on the methods. The figure shows the robustness towards randomized parameters. In the second column, the *Integrated Gradients* approach was able to find the peak. This column corresponds to a model trained on randomized labels. Therefore, the model used in column two was not generalized and only learned to map the training data. Columns three to seven show a model randomization starting from the bottom layers. The results indicate that some methods still performed well when only one or three layers starting from the bottom were randomized, whereas other attribution methods directly collapsed. Columns eight to twelve show the independent layer randomization. Except for *Dynamask*, the attribution techniques were able to deal with handling the layer randomization in the upper layer of the network quite well, whereas all attribution methods collapsed when the lower layers were randomized. Columns thirteen to seventeen show the randomization starting from the top of the network. Most attribution methods were able to recover from the randomization for a high number of randomized layers. Overall, the randomization of the lower layers changed the attribution much more concerning the noise. Interestingly, changes in the upper layers did not affect the attribution methods that much.

5.2.8 *Continuity*

One aspect that is missed out most times is attribution continuity. In the image domain, the use of superpixels solves this problem. However, in the time series domain, it is not that easy. Most of the attribution methods do not consider groups of values. Table 5.7 shows the evaluation of the Continuity. The Continuity calculates the absolute difference between the attribution value of a point t and $t+1$ for each time step and each channel. Using the mean across a sample provides a value that indicates how continuous the explanation is. Lower values correspond to an explanation that does not contain many switches from important to not relevant features. This metric was computed over the 100 attributed samples for each dataset. The results indicate that the perturbation-based approaches favor continuous explanations. Gradient-based methods overall showed the worst performance. One reason for this is the noisy gradients used to compute the attribution maps.

5.3 DISCUSSION

A summary and discussion in a detailed manner are offered to provide on choosing an attribution method. The different aspects and application scenarios are described below. First, it has to be mentioned that every attribution method

Table 5.7: Computed values show the mean continuity of the attribution maps. Lower values correspond to continuous maps. Continuity was calculated by shifting the attribution map, subtracting if from the original one, taking the absolute values, and computing the mean. Lower values are better. Perturbation-based methods showed to outperform gradient-based regarding the continuity on almost all datasets. Specifically, *Dynamask* and *Occlusion* showed to perform well across all datasets.

Method	Anomaly	CharacterTraj.	ECG5000	FaceDetection	FordA	UWaveGesture
Gradient-based						
GradientShap	0.0947	0.0368	0.0616	0.0613	0.0813	0.0543
GuidedBackprop	0.1201	0.0537	0.0913	0.0957	**0.0801**	**0.0526**
InputXGradient	0.0801	0.0390	0.0508	0.0620	0.0855	0.0537
IntegratedGradients	0.0864	0.0369	0.0609	0.0632	0.0858	0.0508
Saliency	0.1176	0.0748	0.1439	0.1170	0.1229	0.0842
Perturbation-based						
Dynamask	**0.0282**	**0.0014**	**0.0252**	**0.0107**	**0.0159**	**0.0015**
FeatureAblation	0.0784	0.0395	0.0584	0.0624	0.0815	0.0601
FeaturePermutation	0.0784	0.0395	0.0584	0.0624	0.0815	0.0601
Occlusion	**0.0623**	**0.0183**	**0.0419**	**0.0367**	**0.0535**	**0.0284**
Miscellaneous						
KernelShap	0.1423	0.1086	0.0641	0.1671	0.1973	0.1795
LIME	0.1122	0.0496	**0.0498**	**0.0010**	0.0883	0.0928
ShapleyValueSampling	**0.0773**	**0.0365**	0.0505	0.0583	0.0885	0.0713

has shown satisfying results. However, the choice of an attribution method should depend on the required characteristics. The overall results are presented in Table 5.8. The results highlight that choosing an attribution method can be crucial, as mentioned by Vermeire et al. [145].

Starting with the accuracy drop, the evaluation shows to which extent the methods rank the most and least significant features based on the impact on the accuracy. Most of the methods were able to show high-quality results across all datasets. However, specifically, the perturbation-based was able to perform slightly better than the other methods on some datasets. *Saliency* and *Dynamask* showed some weaknesses on datasets, such as the *CharacterTrajectories* and *FordA*. Both methods require further adjustments and knowledge about the data to achieve good results. One example is the ratio of significant points for the *Dynamask* approach to select the correct number of features.

Concerning Infidelity and Sensitivity, every method performed well. The results indicate that gradient-based methods obtained the best Infidelity scores. This was the opposite for the Sensitivity. Especially, *GradientShap*, *Input X Gradient*, and *Saliency* were robust against significant perturbations in the input space (Infidelity). On the other side, the *Dynamask*, *Feature Permutation*, and *Occlusion* have shown good robustness concerning changes in the attribution when small perturbations to the input were applied (Sensitivity). Using attribution methods with low Sensitivity values in cases where adversarial attacks can occur is suggested.

The runtime aspect gets critical when the use case requires near real-time explanations. In addition, the results indicate that the dataset characteristics are relevant. The findings show that approaches based on the sequence length and

Table 5.8: Overall results regarding the different aspects evaluated in this paper. A = Accuracy Impact / Agreement, I = Infidelity, S = Sensitivity, R = Runtime, Ld = Label dependency, Md = Model Parameter Dependency, C = Continuity.

Method	A	I	S	R	Ld	Md	C
Gradient-based							
GradientShap		$\oplus$		$\oplus$			
GuidedBackprop	$\oplus$			$\oplus$		$\ominus$	
InputXGradient		$\oplus$		$\oplus$			
IntegratedGradients		$\oplus$		$\oplus$			
Saliency	$\ominus$	$\oplus$		$\oplus$	$\oplus$		
Perturbation-based							
Dynamask	$\ominus$		$\oplus$	$\ominus$	$\oplus$		$\oplus$
FeatureAblation	$\oplus$		$\oplus$	$\ominus$			
FeaturePermutation	$\oplus$		$\oplus$	$\ominus$			
Occlusion		$\oplus$	$\oplus$	$\ominus$			$\oplus$
Miscellaneous							
KernelShap	$\ominus$			$\ominus$	$\oplus$		
LIME		$\oplus$		$\oplus$		$\oplus$	$\oplus$
ShapleyValueSampling	$\oplus$			$\ominus$			$\oplus$

number of channels suffer from very high runtimes for single samples. However, if the time consumption is not of interest, this aspect is not relevant. Furthermore, gradient-based methods are less dependent on the dataset characteristics and very suitable when time matters. Contrarily, besides *Dynamask* and *LIME*, the perturbation-based approaches suffer from several features. In the case of *LIME*, the number of samples required to populate the space to train the surrogate model increases with a higher number of features. *Dynamask* needs an additional training phase, which requires multiple epochs. Based on the computational times, the use of *ShapelyValueSampling* and *KernelShap* in real-time scenarios is nearly impossible. For completeness, it has to be mentioned that it is possible to tweak hyperparameters.

The label permutation and layer randomization provided insights concerning the role of the model parameters during the attribution computation. Intuitively, all methods have shown a high dependency on the labels of the data. Training a model with randomized targets has shown that the attributions depend on the labels, as they should. Concerning the model parameters, the results show that randomizing any layer results in changes in the attribution maps. Besides *Guided-backpropagation*, attribution maps significantly changed after any modification and specifically *LIME* collapsed completely. This collapse emphasizes that *LIME* directly depends on the model, and *Guided-backpropagation* relies more on data.

Finally, Continuity plays a pivotal role in human understanding, as it is beneficial to have continuous attribution maps. Imagine there is a significant frame with many important but some less important features. It might be superior to mark the whole window as important, although this covers some insignificant features. In the time series domain, the context matters, and continuous attribution maps are easier to understand. The results show that *Dynamask*, *LIME*, *Occlusion*, and *ShapleyValueSampling* were superior in their Continuity. Intuitively, the attribution maps produced by gradient-based techniques look noisy, whereas permutation-based look smoother. *Dynamask* includes a loss term that ensures a smoother attribution map. Also, *LIME* and *ShapleyValueSampling* produce smoother maps. The results suggest using a perturbation-based approach if a human inspection is relevant.

5.4 CONCLUSION

A comprehensive evaluation of a large set of state-of-the-art attribution methods applicable to time series was performed. The results indicate that most attribution methods can identify significant features without prior knowledge about the data. Perturbation-based approaches have shown a slightly superior performance in the data occlusion game. In addition, the results were validated by measuring the agreement of the methods using different correlation and similarity measurements. Except for *Dynamask* and *KernelShap*, the correlation between the attribution methods showed high values. Further experiments were conducted to highlight the high dependence of the attribution methods on the model and the target labels. Only *Guided-backpropagation* has shown lower reliance on the top layers of the network. Concerning Infidelity, the gradient-based attribution methods show a superior performance. The perturbation-based attribution methods are superb concerning Sensitivity and Continuity, which is an important aspect when it comes to human interpretability. Furthermore, the results indicate that the choice of an attribution method depends on the target scenario, and different aspects like runtime, accuracy, Continuity, and noise are indispensable.

6 TIMEREISE: A NOVEL TIME SERIES ATTRIBUTION APPROACH

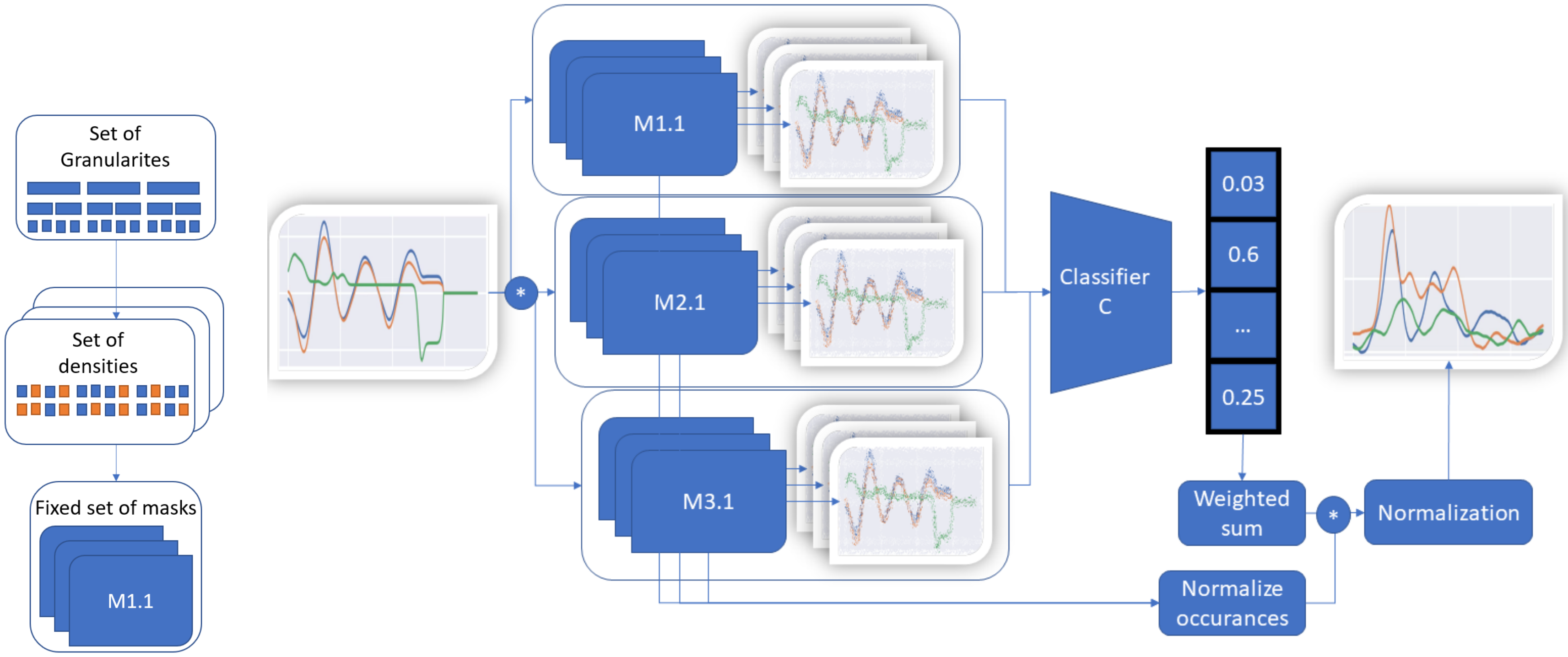
Based on attribution method benchmarks, it is evident that while there exist many different attribution methods, none of these approaches is aligned to the characteristics of time series. Most of these interpretability methods align to the imaging modality intentionally. In this section, *TimeREISE*, a model agnostic attribution method that shows success in the context of time series classification, is introduced. The method applies perturbations to the input and considers different attribution map characteristics, such as the granularity and density of an attribution map. The approach demonstrates superior performance compared to existing methods concerning different well established measurements. Especially, *TimeREISE* scales well with an increasing number of channels and time steps and has no architecture restrictions.

6.1 METHOD

TimeREISE is a novel, a post-hoc interpretability method applicable to any classification network. The approach was inspired by Petsiuk et al. [110]. They presented a random perturbation-based approach for the image domain used as a baseline to build *TimeREISE*. Similar to *RISE* [110] masks are generated, applied to the input, and the output confidence is measured using the classification scores. However, there are several adaptations in the native *RISE* [110] to enhance the approach and successfully apply it to time series data. Besides the simple normalization based on the occurrences of each data point, *TimeREISE* extends this to create masks that evaluate the different channels. Therefore, the masks cover the time and channel dimension. This makes it possible to evaluate different combinations of channel and time steps. The second main addition applied is the summation over different probabilities. *RISE* [110] uses a fixed probability of occluded points to create the masks, resulting in a fixed density. In contrast to that, *TimeREISE* uses masks of different densities and combines them in an additive manner, which removes the assumption of the number of relevant data points. Finally, the granularity is introduced as a parameter to analyze different sizes of patterns.

Figure 6.1 shows the overall workflow of *TimeREISE*. M denotes a set of masks with different window sizes. For example, M1.x can have a smaller window size related to finer granularity. M3.x can have a larger window size for coarse patterns. The same holds for the density. In Figure 6.1a the mask generation is shown.

This chapter is an adapted version of the work presented in: D. Mercier, A. Dengel, and S. Ahmed. "TimeREISE: Time Series Randomized Evolving Input Sample Explanation." In: *Sensors* 22.11 (2022). DOI: 10.3390/s22114084.



(a) Mask creation

(b) Attribution step applied to each individual sample. Uses the existing masks from the mask creation step.

Figure 6.1: Shows the two steps required to use *TimeREISE*. (**a**) shows the generation based on a set of different granularity and densities. The granularity defines the number of slices and the density of the number of perturbed slices within a mask. (**b**) shows a set of masks (M1.1 to M3.x) applied to the input using an exchangeable perturbation function. The masked input is passed to a classifier and the classification score is multiplied (*) by the masks and normalized by the number of feature occurrences.

The process to create the masks can be done once for the dataset as it only depends on the number of time steps, channels and the provided set of granularity and densities. The process shown in Figure 6.1b needs to be executed for each sample.

6.1.1 *Mathematical Formulation*

TimeREISE extends the native mathematical formulation presented by Petsiuk et al. [110] utilizing the different channels. *TimeREISE* generates masks with the shape $s' = (c, t')$ instead of $s'' = (1, t')$ where t' refers to the down sampled time axis and c to the channels. This enhances *TimeREISE* to apply masks that occlude different time steps t' across all channels c within a mask s' instead of using the same time steps t' across all channels c as it is the case for s''. Furthermore, Monte Carlo Sampling is performed across a set of densities P and granularity G to enhance the masks to consider several density values p. Similarly, the use of several granularity values g regularizes the size of the occluded patches. This changes the set of masks as shown in Equation 6.1.

$$M = \{M_0^{p,g}, \dots M_N^{p,g} \mid p \in P \wedge g \in G\} \tag{6.1}$$

Finally, denote S as the weighted sum of the scores produced by the network and the random masks M similar to Petsiuk et al. but normalize each feature as shown in Equation 6.2.

$$S = \sum_{c=0}^{C} \sum_{t=0}^{T} \frac{S_{c,t}}{\sum_{m=0}^{N} M_{c,t,n}} \tag{6.2}$$

6.1.2 *Theoretical Correctness*

Concerning sanity checks mentioned by Adebayo et al. [3], the correctness of the approach is crucial, and it should mainly depend on the learned behavior rather than highlighting dataset-specific features. Adebayo et al. showed that in the image domain, the methods may produce edge detectors. However, this mainly holds for gradient-based methods. *TimeREISE* does not suffer from this, as it only depends on the logits produced by the network prediction. The complete process only depends on the forward pass of the network and has no access to any internal parameters.

6.1.3 *Theoretical Runtime*

For the runtime evaluation, initialization and attribution are considered as two separate processes. Equation 6.3 shows the runtime to create the set of masks for a given set of density probabilities P, granularity G and the number of masks N defined for each combination of p_i and g_i. β is defined as the constant time to create the given map. In addition, P and G are independent of the data shape and therefore do not increase and can be considered as constant factors, leading to a runtime of $\Theta(N)$.

Algorithm 1 Mask generation (Initialization).

```
1: Define: $s$ as input shape, $P$ as a set of probabilities, $G$ as a set of granularities
   for time steps, $N$ number of masks and $M$ as a list of masks.
2: for $p = 1, \dots, P$ do
3:     for $g = 1, \dots, G$ do
4:         for $i = 1, \dots, N$ do
5:             $s' = downscale(s, g)$
6:             $m = uniform(s') < p$
7:             $m = upscale(m, s)$
8:             $m = crop(m, s)$
9:             Append $m$ to $M$
10:        end for
11:    end for
12: end for
13: $S = \frac{S \times M}{N}$
14: $S = \frac{S - \min(S)}{\max(S - \min(S))}$
```

$$t_{init} = P * G * N * \beta \rightarrow t_{init} = \Theta(P * G * N) \rightarrow t_{init} = \Theta(N) \tag{6.3}$$

Equation 6.4 shows the linear runtime of the attribution step. γ is defined as the constant time to apply the perturbation and δ as the constant time the classifier requires to forward pass the sample. Similar to the initialization step, P and G are assumed as constants, resulting in a runtime of $\Theta(N)$.

$$t_{apply} = P * G * N * \gamma * \delta \rightarrow t_{apply} = \Theta(N) \tag{6.4}$$

As the runtime can heavily depend on the implementation and the used hardware, a theoretical analysis offers a much more accurate analysis. It is important to mention that gradient-based methods such as *Guided-backpropagation* are superior concerning their runtime, as they only depend on the backward pass. Perturbation-based methods require multiple forward passes. For *Occlusion* [159] and *Feature Ablation* [159] the number depends on the window size. Using the same window size for a longer sequence results in more forward passes.

As described above, the runtime of *TimeREISE* mainly depends on the number of masks used to compute the attribution map. This number can vary based on the time series, granularity and density. Using hyperparameter tuning, it is possible to find the minimal number of required masks to produce a map that shows only insignificant changes.

6.1.4 *Theoretical Implementation*

The implementation of *TimeREISE* can be divided into two parts, similar to the *RISE* [110] implementation. In the first stage shown in Algorithm 1, a set of masks suited for the input shape gets generated once per dataset. Therefore, every combination of probabilities P and granularity G is generated. The probability

Algorithm 2 Mask application (Attribution).

```
1: Define: $x$ as input, $\theta$ as classifier, $\sigma$ as perturbation function, $S$ as list of scores
   and $N$ as feature occurrences across all masks $M$.
2: for $m = 1, \dots, M$ do
3:     $x_{m_i} = \sigma(x, m_i)$
4:     $y' = \theta(x_{m_i})$
5:     Append $y'$ to $S$
6: end for
7: $S = \frac{S^T \times M}{N}$
8: $S = \frac{S - \min(S)}{\max(S - \min(S))}$
```

refers to a threshold used to determine the density of the mask. Granularity refers to the amount of data considered in a single slice. The down sampling and up sampling is performed along the time axis. Uniform refers to a uniform distribution with the given shape s'. An additional cropping step is performed to preserve the original shape, s.

Algorithm 2 performs the actual attribution. A predefined perturbation method σ is applied to the input x using every mask m_i and is passed to the classifier θ. As default perturbation, the method uses the simple elementwise multiplication of the input x and the mask m_i. This results in a list of scores stored in S. Next, the matrix product of S_T and the masks M is computed, and each point is normalized by the number of occurrences N in the set M. Finally, the map is normalized to values between zero and one.

6.2 DATASETS

Multiple datasets from the well known UEA & UCR repository [9] were used to perform the experiments. The selection of datasets was based on a sufficient number of samples and the dataset modalities such as the number of time steps, channels, and classes. Furthermore, the list of datasets was extended using the *Anomaly* dataset proposed by Mercier et al. [91]. This synthetic dataset served as an interpretable baseline as the point anomalies in this dataset are mathematically defined, and therefore the ground truth attribution is available. Table 6.1 lists the datasets and their characteristics.

6.3 EXPERIMENTS & RESULTS

Following the generic experiment setup is described for the experiments concerning the insertion and deletion, Infidelity, Sensitivity analysis. In addition, the experiments cover a sanity check to validate the correctness of the approach and a runtime analysis to evaluate the dependency on the dataset properties such as channels and time steps.

Table 6.1: Shows the datasets related to critical infrastructures and their different characteristics.

Domain & Dataset	**Train**	**Test**	**Steps**	**Channels**	**Classes**
Critical Manufacturing					
Anomaly (synthetic data) [91]	50,000	10,000	50	3	2
ElectricDevices	8,926	7,711	96	1	7
FordA	3,601	1,320	500	1	2
Food and Agriculture					
Crop	7,200	16,800	46	1	24
Strawberry	613	370	235	1	2
Public Health					
ECG5000	500	4,500	140	1	5
FaceDetection	5,890	3,524	62	144	2
MedicalImages	381	760	99	1	10
NonInvasiveFetalECG	1,800	1,965	750	1	42
PhalangesOutlinesCorrect	1,800	858	80	1	2
Communications					
CharacterTrajectories	1,422	1,436	182	3	20
HandOutlines	1,000	370	2,709	1	2
UWaveGestureLibraryAll	896	3,582	945	1	8
Wafer	1,000	6,164	152	1	2
Transportation Systems					
AsphaltPavementType	1,055	1,056	1,543	1	3
AsphaltRegularity	751	751	4,201	1	2
MelbournePedestrian	1,194	2,439	24	1	10

6.3.1 *Baseline Accuracy*

During the experiments, InceptionTime [35] was used, as it was the state-of-the-art model at that time. The network architecture consists of multiple inception blocks followed by a global average pooling and a fully connected layer. Each inception block consists of multiple convolutional layers and a max-pooling. Furthermore, the network uses residual connections between the input and the different inception blocks. It is based on the Inception-v4 [138] architecture but specifically aligned to the time series domain and has shown to achieve good performances across time series classification datasets while being very robust. Figure 6.2 shows the architecture of IncpetionTime.

The network was trained using a learning rate scheduler to reduce the learning rate on plateaus and early stopping to prevent overfitting. As an optimizer, SGD was applied with an initial learning rate of 0.01 and a maximum of 100 epochs. The experiments indicate that every model converged in less than 100 epochs. As some datasets are huge, and the computation of measures such as the Sensitivity

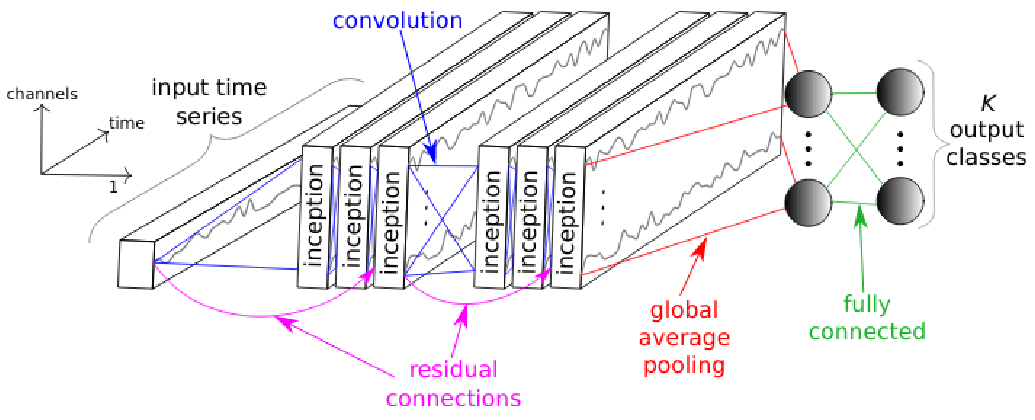


Figure 6.2: Shows the general architecture of InceptionTime proposed by Fawaz et al. [35]. Reprinted from [35].

Table 6.2: Shows the performance of IncpetionTime on the complete and sub-sampled dataset. The results indicate that the subset accuracy was equal to the complete test set accuracy. Numbers are given in percentage.

Dataset	**Test Data**			**100 Samples**		
	Macro F1	**Micro F1**	**Acc.**	**Macro F1**	**Micro F1**	**Acc.**
Anomaly	97.69	98.71	98.72	96.99	97.97	98.00
AsphaltPavementType	91.69	92.44	92.42	89.05	89.91	90.00
AsphaltRegularity	100.00	100.00	100.00	100.00	100.00	100.00
CharacterTrajectories	99.40	99.44	99.44	100.00	100.00	100.00
Crop	71.89	71.89	72.81	70.58	72.28	74.00
ECG5000	56.11	93.52	94.36	60.45	94.12	95.00
ElectricDevices	62.86	69.35	70.56	67.09	76.02	79.00
FaceDetection	66.34	66.34	66.37	67.79	67.90	68.00
FordA	94.92	94.92	94.92	92.94	92.99	93.00
HandOutlines	94.64	95.10	95.14	93.99	94.93	95.00
MedicalImages	72.27	74.61	74.74	70.86	74.79	75.00
MelbournePedestrian	94.22	94.24	94.22	96.35	95.95	96.00
NonInvasiveFetalECG	94.00	94.30	94.25	84.24	92.40	92.00
PhalangesOutlinesCorrect	81.42	82.54	82.75	88.49	88.98	89.00
Strawberry	95.54	95.93	95.95	96.72	96.99	97.00
UWaveGestureLibraryAll	91.65	91.67	91.74	85.25	86.96	87.00
Wafer	99.54	99.82	99.82	100.00	100.00	100.00
Average	86.13	89.11	89.31	85.93	89.54	89.88

is computationally expensive, a set of 100 test samples is used to perform the attribution. In addition, the base accuracy scores for the whole datasets and the subsets are provided in Table 6.2.

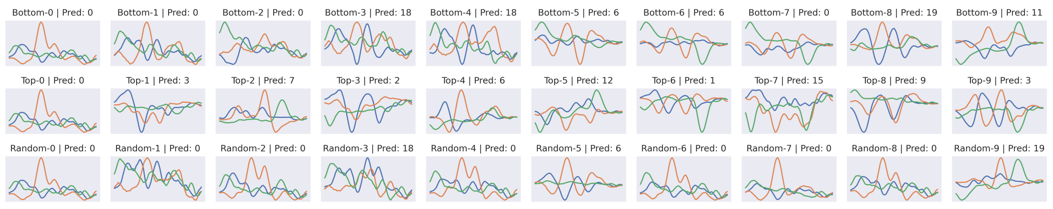


Figure 6.3: Shows the attribution map of a *CharacterTrajectories* sample. Top-X: top-down randomization of X layers, Bottom-X: bottom-up randomization of X layers. Random-X: randomized layer/block X. Top-0 refers to the original attribution map. The results provide evidence that *TimeREISE* method strongly depends on the model parameters.

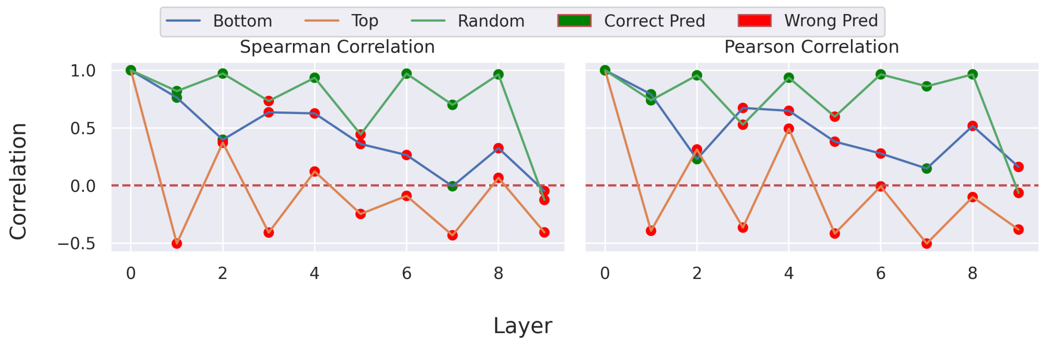


Figure 6.4: Shows the correlation between the maps with randomized layers and the original attribution map. Refers to the attribution maps shown in Figure 6.3. Changes to the network resulted in a lower correlation of the attribution map compared to the original map for *TimeREISE.*

6.3.2 *Sanity Check*

In addition to the theoretical explanation of the correctness, a sanity check was conducted. Therefore, a sample of the *CharacterTrajectories* dataset was used, and the attribution map for different states of the model was computed. In Figure 6.3 the different attribution maps are shown. The first column always shows the original attribution map. Going from left to right increases the number of randomized layers for the top-down and bottom-up approaches. The first row refers to the bottom-up approach in which the layers were sequentially randomized starting from the first convolutional block in the first inception block up to the last dense layer. Respectively, the second row shows the top-down approach where the dense layer was randomized first and the first convolutional block of the first inception block last. The third row covers the independent randomization of a single layer in the case of 'Random-9' as it refers to the last dense layer and single block for the other cases. Across all setups, it is visible that randomizing the layer weights resulted in significant changes in the attribution map.

In Figure 6.4 the Spearman and Pearson Correlation between the original and randomized attribution map is given. The Spearman Correlation was used, as it is a rank-based approach. The color of the individual points shows the correctness of the prediction using the manipulated network. It is visible that

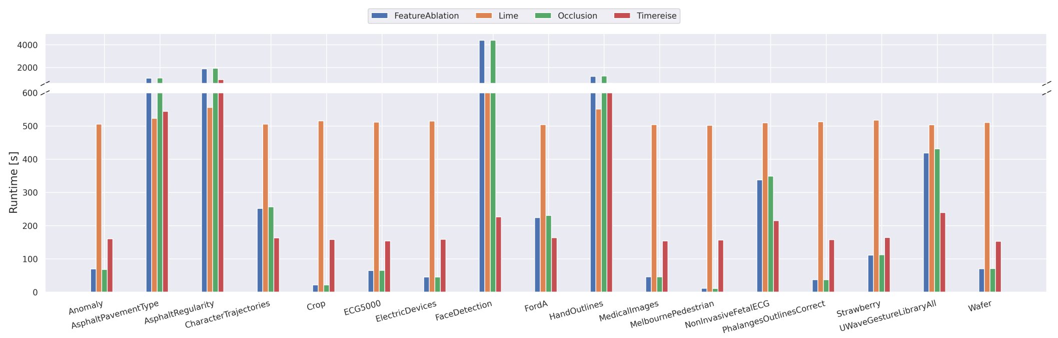


Figure 6.5: Shows the runtime on the different datasets for 100 attribution maps. *TimeREISE* shows a stable runtime across all datasets. For datasets with long sequences, the runtime of the other approaches increases.

the correlation of the attribution maps with the correct prediction is higher than for others. This shows that *TimeREISE* successfully depends on the prediction of the network. Furthermore, the figure shows that for the top-down randomization, the correlation drops by a large value. Similarly, the bottom-up randomization showed an increasing drop in correlation. However, randomizing a single layer or block resulted in higher correlation values, except for the randomization of the last dense layer. This can be explained by the structure of InceptionTime, as it is very robust concerning the randomization of a single block. This is further validated by the correct predictions within this setup.

6.3.3 *Runtime Analysis*

Figure 6.5 shows the runtimes for the experiments executed on 100 samples. The results show that methods that directly depend on a window size, such as *Feature Ablation* [159] and *Occlusion* [159], required much longer processing for the datasets that had numerous features. In particular, the run times for the four data sets with the highest number of channels and time steps illustrate this behavior. To reduce the processing time, it is possible to select a larger window. However, this requires knowledge about the dataset and the size of the pattern within the dataset. *Guided-backpropagation* [134] and *Integrated Gradients* [136] were excluded as they do not depend on such properties of the datasets. *LIME* [115] and *TimeREISE* mainly depend on the number of samples and masks defined for each of the approaches. The experiments show that in both cases, the runtime of both was constant for the parameters that were selected. However, as mentioned, it would be possible to fine-tune the parameters.

In addition to the real datasets, Figure 6.6 shows the increase based on the number of time steps and channels. The runtime of the *Feature Ablation* and *Occlusion* increased based on the time steps and the channels, whereas the processing time of *LIME* and *TimeREISE* only slightly increased as the forward passes requires more time. However, this holds for the gradient-based methods and the backpropagation too. The difference is that the number of backward passes required in those methods is limited, whereas the number of forward passes for *LIME* and *TimeREISE* is higher. It has to be mentioned that the number

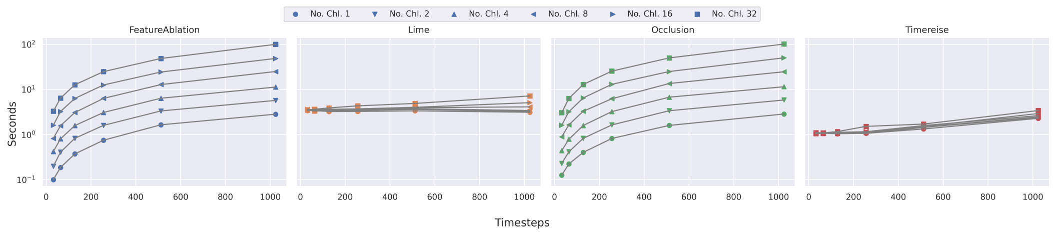
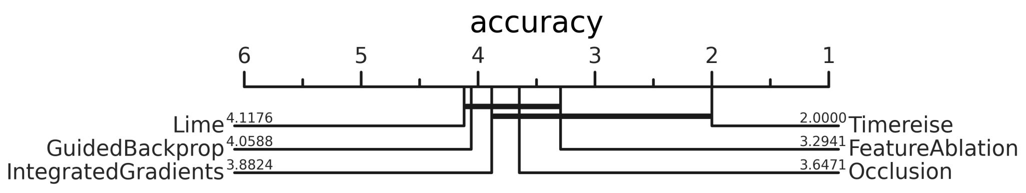
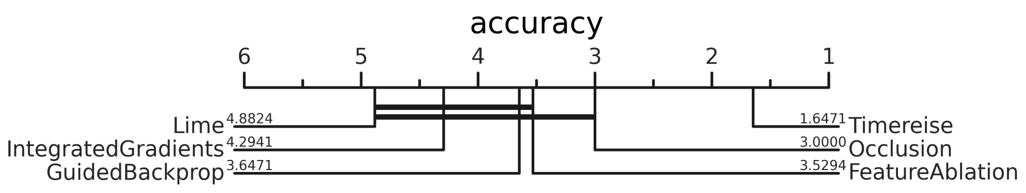


Figure 6.6: Shows the runtime on synthetic data with different number of time steps and channels. *Integrated Gradients* and *Guided-backpropagation* were excluded. The runtime of *Feature Ablation* and *Occlusion* increases dramatically with the longer sequences. *TimeREISE* and *LIME* show a stable runtime.



(a) Deletion of important data points



(b) Insertion of important data points

Figure 6.7: Critical difference diagram showing the average rank of each attribution method across all datasets. The ranking is based on the AUC using accuracy. Perturbation-based approaches achieve better results on the deletion and insertion test. *TimeREISE* shows a superior performance for both tests.

might vary depending on the implementation and hardware. However, they provide insights into the expected behavior when changing the dataset.

6.3.4 *Insertion & Deletion*

Fong and Vedaldi [40] used the insertion and deletion metric to explain the significant values of an attribution method. The intuition behind the metric is that the prediction of a classifier changes if the cause gets removed or added. In the case of the deletion, the points starting with the most important one are removed from the input, and the prediction is computed. Large drops suggest that the feature was significant for the prediction. In the case of the deletion, lower Area Under the Curves (AUCs) suggest that the method is superior in spotting important parts of the input. For the insertion, higher AUCs are superior. Large increases in this setup correspond to adding important data points relevant to the prediction.

Figure 6.7 shows the critical difference diagrams of every attribution method. These were calculated using the AUC scores. In Figure 6.7a *TimeREISE* shows an outstanding performance compared to the other state-of-the-art methods regarding the deletion of significant data. Another important finding is that methods which utilize a window, such as *Feature Ablation* and *Occlusion* showed better performances concerning the deletion compared to methods that directly depend on the gradients, such as *Guided-backpropagation* and *Integrated Gradients*. However, Figure 6.7b highlights that the results were the same for the insertion task. One reason for its outcome is the smoothing applied to approaches that use a defined window. Gradient-based methods provide noisy and spiking attribution maps.

Table 6.3: Sequential deletion of the most important points from the original input signal. Respectively, sequential insertion of the most important points starts with a sample consisting of mean values. Lower AUC scores are better for deletion. Higher AUC scores are better for insertion. AUC was calculated using classification accuracy. *TimeREISE* outperformed any other method. The best results are highlighted in bold.

Dataset	**FeatureAblation [38]**		**GuidedBackprop [136]**		**IntegratedGrad. [134]**		**LIME [115]**		**Occlusion [159]**		**TimeREISE**	
	del	**ins**	**del**	**ins**	**del**	**ins**	**del**	**ins**	**del**	**ins**	**del**	**ins**
Anomaly	0.7731	0.9737	0.7791	0.9597	0.7786	0.9624	0.7783	0.9473	0.7714	0.9739	**0.7631**	**0.9867**
AsphaltPavementType	0.4073	0.8819	0.3930	**0.8944**	**0.3940**	0.8935	0.4622	0.8623	0.4171	0.8726	0.4135	0.8641
AsphaltRegularity	0.5857	0.9954	**0.5785**	**0.9960**	0.5817	0.9964	0.6843	0.9871	0.5901	0.9929	0.5927	0.9833
CharacterTrajectories	0.0856	0.8563	**0.0807**	0.8701	0.1091	0.8580	0.0785	0.8543	0.0878	0.8609	0.0401	**0.8809**
Crop	0.0998	0.3780	0.1402	0.3026	0.1404	0.2652	0.1096	0.3198	0.1583	0.3170	**0.0628**	**0.5065**
ECG5000	0.2104	0.8771	0.1876	0.8782	0.1208	0.8792	0.1294	0.8846	0.1176	0.8796	**0.1015**	**0.9060**
ElectricDevices	0.3086	0.5393	0.3616	0.5718	0.3178	0.5244	0.3338	0.4971	0.3524	0.5914	**0.2726**	**0.6957**
FaceDetection	0.5165	0.6760	0.2462	0.8065	0.5116	0.6660	0.6019	0.6308	0.5281	0.6691	**0.0080**	**0.9968**
FordA	0.4729	0.7816	0.4829	0.8207	0.4793	0.6834	0.4803	0.6731	0.4751	0.8493	**0.3859**	**0.9436**
HandOutlines	0.3125	0.3630	0.3137	0.3289	0.3127	0.3432	0.3153	0.3201	**0.3107**	**0.3911**	0.3485	0.3607
MedicalImages	0.1840	0.5884	0.1588	0.5645	0.1953	0.4518	0.1736	0.5622	0.1569	0.5883	**0.1229**	**0.7125**
MelbournePedestrian	0.1579	0.5967	0.2071	0.5579	0.2733	0.4579	0.1767	0.6013	0.2363	0.4763	**0.0979**	**0.6538**
NonInvasiveFetalECG	0.0424	0.1488	0.0454	0.0654	0.0405	0.0868	0.0462	0.0816	**0.0422**	0.2503	0.0894	**0.4333**
PhalangesOutlinesCorrect	0.4033	0.5072	0.4058	0.4347	0.4056	0.4437	0.4038	0.4288	0.4034	0.5616	**0.2919**	**0.6171**
Strawberry	0.5827	0.7179	0.6141	0.7100	0.6428	0.7179	0.6397	0.7087	0.5958	0.7761	**0.3882**	**0.7909**
UWaveGestureLibraryAll	0.1840	0.4243	0.1353	0.5260	0.1285	0.1452	0.1226	0.1782	0.1743	0.4669	**0.0973**	**0.5379**
Wafer	0.2740	0.7684	0.3441	0.8574	0.2603	0.8061	0.2324	0.8613	0.2642	0.7932	**0.2002**	**0.8976**
Average	0.3295	0.6514	0.3220	0.6556	0.3348	0.5989	0.3393	0.6117	0.3342	0.6653	**0.2516**	**0.7510**

Table 6.4: Lower Infidelity values correspond to better performance. The method names are shortened by taking only the initial character (*Feature Ablation*, *Guided-backpropagation*, *Integrated Gradients*, *LIME*, *Occlusion*, *TimeREISE*). There are only insignificant differences between the methods. The best results are highlighted in bold.

Dataset	**F [38]**	**G [136]**	**I [134]**	**L [115]**	**O [159]**	**T (Ours)**
Anomaly	0.0233	0.0193	**0.0158**	0.0184	0.0222	0.0230
AsphaltPavementType	0.2126	0.2126	0.2126	0.2127	0.2126	**0.2124**
AsphaltRegularity	**0.0045**	0.0046	0.0046	0.0046	**0.0045**	**0.0045**
CharacterTrajectories	0.1399	0.1397	0.1399	0.1399	0.1399	**0.1396**
Crop	0.2967	0.3081	0.3055	**0.2966**	0.3143	0.3032
ECG5000	0.0273	0.0272	0.0257	**0.0210**	0.0236	0.0242
ElectricDevices	18.0869	**18.1047**	18.1130	18.1042	18.0854	18.1070
FaceDetection	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
FordA	0.0118	0.0118	0.0116	0.0116	0.0118	0.0118
HandOutlines	**1.6914**	1.7015	1.6932	1.6928	1.6938	1.6920
MedicalImages	0.2492	0.2492	**0.2472**	0.2486	0.2490	0.2482
MelbournePedestrian	1.2324	1.2833	1.3745	1.1959	1.3319	**1.2301**
NonInvasiveFetalECG	51.7361	51.7288	51.7252	51.7228	51.7413	**51.7072**
PhalangesOutlinesCorrect	0.4394	**0.4285**	0.4360	0.4413	0.4403	0.4405
Strawberry	0.4865	**0.4783**	0.4863	0.4849	0.4811	0.4851
UWaveGestureLibraryAll	4.9995	4.9983	4.9922	4.9996	4.9992	**4.9968**
Wafer	0.0355	0.0356	0.0355	0.0356	0.0355	**0.0352**
Average	4.6867	4.6901	4.6952	**4.6842**	4.6933	4.6859

Table 6.3 shows the different results of the deletion and insertion for every individual dataset. Furthermore, the table provides the average scores achieved by the methods. *TimeREISE* showed a superior behavior in both the average deletion and insertion score. The method achieves the best (lowest) score for 13 datasets and an average of 0.2516. The second-best approach concerning the average AUC score was *Guided-backpropagation* with a score of 0.3220 and two times the best performance. In addition, *TimeREISE* showed the best average score for the insertion, with 14 best scores. *Guided-backpropagation* achieved twice the best score in the insertion task. However, the average score of *TimeREISE* was 0.7510 compared to the second best of 0.6653 for the *Occlusion*.

6.3.5 *Infidelity & Sensitivity*

The Infidelity and Sensitivity proposed by Yeh et al. [155] cover significant and insignificant changes applied to the attribution and the input. The intuition behind Infidelity is that a significant perturbation of the attribution map leads to a change in the prediction. Similarly, the Sensitivity is calculated using an insignificant change in the input sample. The Sensitivity requires recomputing the attribution maps. For both Infidelity and Sensitivity, lower values are better. For Infidelity, 1,000 perturbations were computed for each of the 100 samples and

Table 6.5: Lower Sensitivity values correspond to better performance. The method names are shortened by taking only the initial character (*Feature Ablation*, *Guided-backpropagation*, *Integrated Gradients*, *LIME*, *Occlusion*, *TimeREISE*). Perturbation-based approaches show a superior performance. *TimeREISE* showed the best performance across most of the datasets. The best results are highlighted in bold.

Dataset	**F [38]**	**G [136]**	**I [134]**	**L [115]**	**O [159]**	**T (Ours)**
Anomaly	0.0574	0.0747	0.1470	0.2591	0.0664	**0.0522**
AsphaltPavementType	0.0292	0.2864	0.0358	0.4259	**0.0274**	0.0705
AsphaltRegularity	0.0288	0.2797	0.0567	0.3664	0.0274	**0.0028**
CharacterTrajectories	0.0199	0.0547	0.0705	0.1353	0.0174	**0.0076**
Crop	0.0808	0.1060	0.1702	0.1786	0.1307	**0.0411**
ECG5000	0.0301	0.0772	0.1218	0.1811	0.0248	**0.0111**
ElectricDevices	0.2069	0.2608	0.6129	0.2622	0.1949	**0.1696**
FaceDetection	0.0180	0.0204	0.0136	0.4722	0.0144	**0.0048**
FordA	0.0231	0.0384	0.0708	0.1690	0.0155	**0.0147**
HandOutlines	0.0952	0.1545	0.1203	0.1249	**0.0743**	0.1175
MedicalImages	0.0428	0.0680	0.1483	0.1754	**0.0395**	0.0406
MelbournePedestrian	0.1667	0.1363	0.1684	0.2514	0.2176	**0.0472**
NonInvasiveFetalECG	0.1142	0.1043	0.1543	0.1564	**0.0869**	0.1570
PhalangesOutlinesCorrect	0.0415	0.1442	0.1562	0.1212	**0.0390**	0.0574
Strawberry	0.0486	0.0966	**0.0506**	0.1267	0.0515	0.0698
UWaveGestureLibraryAll	0.0569	0.0535	0.2341	0.1778	**0.0373**	0.0381
Wafer	0.0252	0.0368	0.1299	0.1250	0.0141	**0.0051**
Average	0.0638	0.1172	0.1448	0.2182	0.0635	**0.0533**

computed the averaged Infidelity value. In addition, 10 perturbations for each of the samples were used to compute their Sensitivity.

Starting with Infidelity, the results shown in Table 6.4 emphasized that there is no significant difference between the different methods. Overall, the average scores differed only by 0.011, which is an insignificant difference. Across all datasets, the methods performed similarly, and it is impossible to create a critical difference diagram as the null hypothesis did hold. Interestingly, the Infidelity scores for the *ElectricDevices* and *PhalangesOutlinesCorrect* dataset were much larger compared to the other datasets.

The Sensitivity experiments are shown in Table 6.5. The results of these experiments show a significant difference between the methods. The best result was achieved by *TimeREISE*, with a score of 0.0533. The worst result was achieved by *LIME*, with a score of 0.2182, which was about four times larger than the score of *TimeREISE*. The overall finding was that the perturbation-based approaches are superior in the case of Sensitivity compared to the gradient-based or others. This is the case as the gradient-based methods result in noisy attribution maps, whereas the perturbation-based come up with smoothed maps based on a window of multiple features. This smoothing increases the robustness against minor changes in the input.

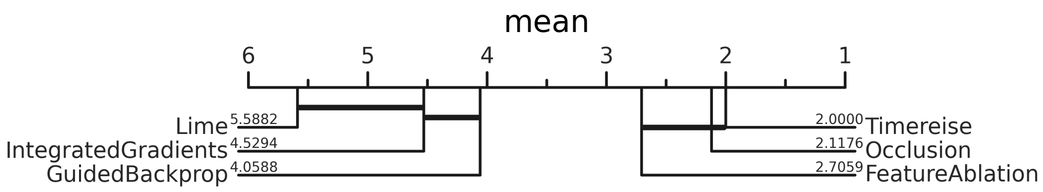


Figure 6.8: Shows critical difference diagram highlighting the average sensitivity rank of each attribution method across all datasets. The ranking is based on the average Sensitivity. Perturbation-based approaches show superior performance. The horizontal bars show the groups that can be concluded based on the rank of the methods. They clearly highlight the high and low performing sets.

In Figure 6.8 the critical difference diagram across all datasets is shown. It highlights the superior performance of the perturbation-based approaches compared to the other approaches. In addition, it highlights that *TimeREISE* was only slightly above the Occlusion method.

6.3.6 *Attribution Continuity*

Furthermore, the Continuity proposed by Abdul et al. [2] was calculated. Continuity is a measurement that bridges correctness and visual interpretability. The Continuity for each feature was calculated as presented in Equation 6.5 and took the mean for the overall evaluation between the methods. Lower values are better regarding the cognitive load, but might conflict with the exact correctness of the feature importance.

$$C = \sum_{c=0}^{C} \sum_{t=0}^{T-1} \mid S_{c,t} - S_{c,t+1} \mid \tag{6.5}$$

Table 6.6 shows the average Continuity of the attribution methods. Similar to the Sensitivity, smaller values are better. Interestingly, the performance of the attribution methods was very similar to the Sensitivity. Again *TimeREISE* showed superior performance with a score of 0.0267 compared to Occlusion as the second-best approach with a score of 0.0565. The reason for the superior performance is the smooth mask design. The masks of *TimeREISE* are created on a down scaled sample, and then they are upscaled using interpolation to the original input size, which results in smoother masks.

Figure 6.9 shows the corresponding critical difference diagram, which looks similar to the diagram for the Sensitivity. This is intuitive that the Sensitivity and the continuity definition have an overlap.

6.3.7 *Visualization*

The visualization presents some interpretable attribution maps. The results highlight that *TimeREISE* produces smoother attribution maps while preserving the similar shape compared to the other attribution methods. *TimeREISE* makes

Table 6.6: Lower continuity values correspond to better performance. The method names are shortened by taking only the initial character (*Feature Ablation*, *Guided-backpropagation*, *Integrated Gradients*, *LIME*, *Occlusion*, *TimeREISE*). *TimeREISE* showed a superior performance across all datasets. The best results are highlighted in bold.

Dataset	**F [38]**	**G [136]**	**I [134]**	**L [115]**	**O [159]**	**T (Ours)**
Anomaly	0.1163	0.1444	0.1309	0.1390	0.0908	**0.0473**
AsphaltPavementType	0.0792	0.0977	0.0770	0.0765	0.0450	**0.0015**
AsphaltRegularity	0.0582	0.0703	0.0485	0.0525	0.0334	**0.0008**
CharacterTrajectories	0.0264	0.0324	0.0368	0.0619	0.0243	**0.0134**
Crop	0.1282	0.1655	0.1952	0.1741	0.0985	**0.0618**
ECG5000	0.0682	0.1000	0.1004	0.0844	0.0505	**0.0296**
ElectricDevices	0.2016	0.1840	0.1984	0.1950	0.0884	**0.0350**
FaceDetection	0.0690	0.0745	0.0613	0.0331	0.0373	**0.0161**
FordA	0.0770	0.0819	0.0959	0.1530	0.0576	**0.0083**
HandOutlines	0.0123	0.0183	0.0258	0.1501	0.0106	**0.0015**
MedicalImages	0.0923	0.1043	0.1259	0.1076	0.0602	**0.0371**
MelbournePedestrian	0.1804	0.1844	0.2217	0.1881	0.1264	**0.1052**
NonInvasiveFetalECG	0.0224	0.0650	0.0753	0.1603	0.0197	**0.0043**
PhalangesOutlinesCorrect	0.1066	0.1187	0.1525	0.1416	0.0715	**0.0496**
Strawberry	0.0720	0.0679	0.0785	0.1447	0.0676	**0.0159**
UWaveGestureLibraryAll	0.0216	0.0557	0.0816	0.1629	0.0226	**0.0038**
Wafer	0.0924	0.0957	0.1418	0.1222	0.0557	**0.0232**
Average	0.0838	0.0977	0.1087	0.1263	0.0565	**0.0267**

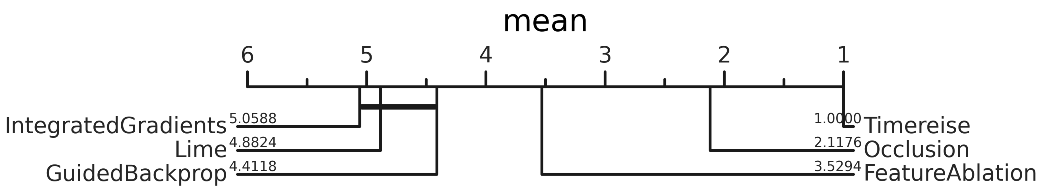
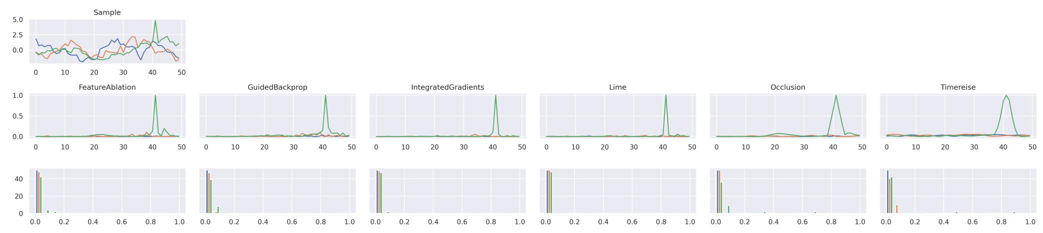


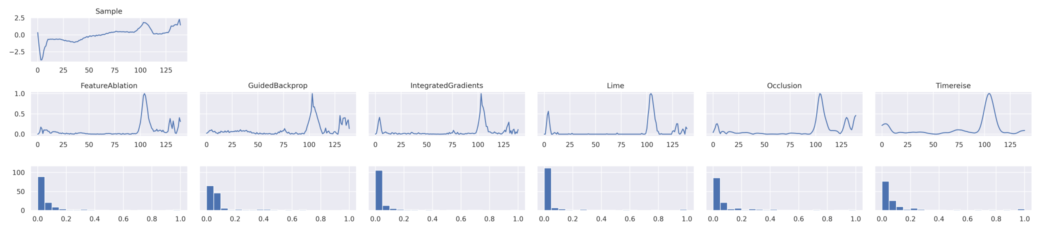
Figure 6.9: Critical difference diagram showing the average continuity rank of each attribution method across all datasets. The ranking is based on the average Continuity. Perturbation-based approaches show superior performance. The horizontal bars show the groups that can be concluded based on the rank of the methods. They clearly highlight the low performing set.

a good compromise between the visual appearance, which is strongly influenced by Continuity and noise, and the correctness of the feature importance values.

In Figure 6.10 an attribution map of every technique is shown. The first Figure 6.10a shows an anomalous sample of the *Anomaly* dataset. The anomaly is represented by the peak in the green signal. All methods successfully identified the peak as the most important part. However, the *Occlusion* and *TimeREISE* highlighted that the neighborhood points of the peak are important. Whereas the intuition first suggests that only the peak should be highlighted, this is not correct,



(a) Anomaly detection dataset sample. Anomalies are defined as point anomalies represented by peaks.



(b) ECG5000 data sample. Classification task that depends on a single channel.

Figure 6.10: Shows the attribution maps for a single sample. First row: original sample, second row: attribution, third row: histogram of the attribution. Low values in the histogram relate to a good separation between relevant and irrelevant features. (**a**) shows an anomalous sample in which the green peak corresponds to the anomalous signal part. Overall, the attributions look similar, except that *TimeREISE* is smoother compared to the other methods.

as changing the neighborhood points will influence the peak. Furthermore, it is visible that the attribution map provided by *TimeREISE* is very smooth compared to the other attributions.

In Figure 6.10b an attribution map for the *ECG5000* dataset is shown. The results of all methods look similar to a certain degree. However, except for *TimeREISE*, the last part of the sequence was identified as features with some importance. In addition, the attribution maps included some noise. Specifically, the first negative peak in the signal was captured by the *Integrated Gradients* and *LIME* to be an important part. This was not the case for the remaining methods, and changing this part or the last part showed only a minor effect on the prediction.

Figure 6.11 shows the results of the attribution applied to an interpretable character trajectory sample. The Figure presents the time series sample and its back transformation to two-dimensional space. Furthermore, the attribution maps given in the second row showed the smoothness of *TimeREISE*. One finding was that the horizontal and vertical movement were rated as more important by most methods, and that the majority of important points occur within the first 100 time steps. Interestingly, *Guided-backpropagation* showed in a surprisingly high relevance for the force. *Feature Ablation* and *Occlusion* showed low importance for both the vertical movement and the pressure.

6.4 DISCUSSION

This subsection discusses the results to gain a better impression of the relevance and possible applications of *TimeREISE*. Furthermore, the advantages and

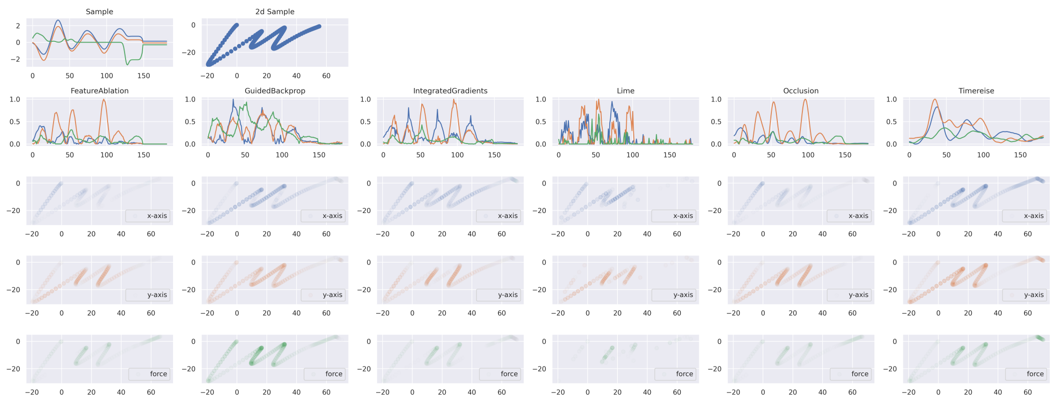


Figure 6.11: First row: time series of the sample 'm' and 2D transformation, second row: attribution, Subsequent rows: importance of horizontal, vertical, force values. *TimeREISE* provides a smooth attribution map and assigns importance to the force channel. Besides *TimeREISE*, only *Guided-backpropagation* highlighted the importance of the force channel.

drawbacks of the existing methods are mentioned. To summarize the experiments, the datasets are grouped below based on their properties described in Table 6.1. Data sets with less than 1,000 training samples are referred to as small data sets. Furthermore, a distinction is made between univariate and multivariate data sets, as well as between binary and multi-class data sets. A final characteristic is the sequence length, which is considered a long sequence if it exceeds 500.

During the sanity checks, *TimeREISE* has shown a strong dependency on the model parameters. This dependency is an indication that the method does not represent dataset characteristics, but visualizes the points considered relevant by the model. The sanity check showed strong robustness to single layer manipulation and a large drop-off in attribution map correlation when layers were manipulated sequentially.

The runtime analysis also showed that *TimeREISE* scales are better compared to other perturbation-based approaches like *Occlusion*. Specifically, the long sequences and multivariate datasets resulted in a dramatic increase in the runtime for perturbation-based methods and *LIME*. It is worth mentioning that the gradient-based methods are always superior to the other methods in terms of runtime, since they require only one backward pass. However, *TimeREISE* has shown superior behavior compared to the remaining methods, specifically when long sequences or high channel numbers occur. For short sequences, the runtime of *TimeREISE* was nearly constant. In addition, it is possible to precisely adapt the runtime of *TimeREISE* to the use case by incorporating knowledge about the data set. One example is the detection of point anomalies, where a low density is sufficient.

The results of the deletion and insertion test provided evidence that the attribution maps are relevant for the prediction. Especially in the deletion test, *TimeREISE* showed excellent results, which directly lead to performance drops when important features were removed. Besides two long sequence datasets, *TimeREISE* has shown superior performance across all dataset categories. In general, the performance of window-based approaches was good in the insertion test, since they do not depend on gradients. The detection of a peak is an example of a use case where the insertion test gives a false picture. Methods that use the gradients showed lower importance at the time steps adjacent to the peak, while this was not the case with *TimeREISE*.

In addition, *TimeREISE* shows excellent results for Infidelity and Sensitivity. *TimeREISE* has shown the best Sensitivity values across all datasets. Concerning Infidelity, *TimeREISE* achieved the second-best overall performance. The best scores were achieved in long sequences. It has to be mentioned that *LIME* was slightly better in the overall Infidelity, however, *TimeREISE* achieved more individual best scores for the datasets. Due to the mask-based design, *TimeREISE* has high robustness in case of insignificant changes in the input. This is also because the attribution masks created by *TimeREISE* have a high Continuity, which contributes to the interpretability. Despite this Continuity, the attribution masks were shown to withstand the sanity check and thus offer a good compromise between explanatory power and correctness. Finally, the visualizations of the different attribution methods shows that *TimeREISE* provides less noisy explanations. *TimeREISE* was able to show the best Continuity values across all datasets.

In summary, *TimeREISE* has proven to be an outstanding method for time series analysis in terms of correctness, runtime and interpretability. In addition, the customization of the hyperparameters allows easy adaptation to different aspects of use. In contrast to existing perturbation-based methods, *TimeREISE* can be applied without knowing the size of the relevant pattern within the data, making it more effective in cases where the solution to the question of which parts are relevant is not known.

6.5 CONCLUSION

The experiments indicated that the proposed attribution method *TimeREISE* can achieve excellent performance concerning most of the evaluated metrics across all selected datasets. The method outperforms other state-of-the-art attribution methods concerning the Continuity, Sensitivity, and insertion and deletion metrics. Specifically, in the deletion test, a superb performance was shown. Furthermore, the experiments demonstrated that the method produces smooth attribution maps that require less effort to interpret. Concerning Infidelity, the approach was on par with the state-of-the-art methods. Further, the theoretical runtime evaluation shows that the method has a better scaling compared to methods that directly depend on the number of features and is applicable to any classifier. Another positive aspect is that the method does not depend on noisy gradients or internal classifier variables. Ultimately, the sanity checks highlight the dependency on the model parameters and the robustness. Concerning the runtime, the method showed superior results compared to the perturbation-based methods when the dataset covers long sequences or multiple channels. For short univariate sequences, the runtime was nearly constant and only slightly above the runtime of the other perturbation-based methods. Compared to the gradient-based methods, the runtime of *TimeREISE* and any other perturbation-based method were inferior.

7 TSVIZ: A NOVEL GRADIENT-BASED VISUALIZATION FRAMEWORK

Using attribution maps to visualize the impact of data points has proven to be successful. However, visualization in the domain of time series is significantly challenging and little or no concentration has been devoted to the development of visualization tools. While it is straightforward to show the impact of the input using an attribution mask the in-depth analysis of a network including each neuron, the interaction between them and possible optimization requires a far more advanced visualization.

In this section, a novel component, namely *TSViz*, for the demystification of convolutional deep learning models for time series analysis is presented. The framework is based on one of the previously analyzed *Saliency* method. To propose a complete system, the following functionalities were implemented and evaluated:

- Input relevance: Shows the importance of the input based on the output.
- Filter importance: Shows the importance of every neuron based on the output.
- Network clustering: Partitions the network based on activation similarity.
- Network pruning: Optimizes the network by removing redundancy or less important neurons.
- Input optimization: Maximizes the output by input modifications to explain the source of variation.
- Impact of noise: Use Fast Gradient Sign Method (FGSM) [52] and iterative FGSM [69]) to understand the impact of adversarial noise.
- Three dimensional visualization: Enables interactive network exploration.

7.1 DATASETS

To evaluate the effectiveness and correctness of the proposed visualization framework, two different time series datasets were used. Following the regression and the classification, datasets and task-specific characteristics are described. In addition, the dataset statistics are shown in Table 7.1.

This chapter is an adapted version of the work presented in: D. Mercier et al. “Tsviz: Demystification of deep learning models for time-series analysis.” In: *IEEE Access* 7 (2019), pp. 67027–67040.

Table 7.1: Used regression and classification dataset.

Dataset	**Train**	**Test**	**Steps**	**Channels**	**Classes**
Internet Traffic	13,900	4,150	50	2	-
Anomaly	50,000	10,000	50	3	2

7.1.1 *Regression*

The goal of the regression task is to predict the next time step based on the previous information. As a dataset to evaluate this, the *Internet Traffic* [21] dataset for time series forecasting was used. In contrast to the classification, this requires some modifications to the network, in particular the loss function needs to be adjusted for the regression case.

7.1.2 *Classification*

For the time series classification task, a novel dataset was proposed. This dataset covers the task of anomaly detection and is referred as *Anomaly* dataset [1].The dataset consists of 60,000 sequences of 50 time steps each, where each time step contains values for pressure, temperature and torque. Significant peaks were randomly introduced in the dataset, and sequences containing such point anomalies were marked as anomalous. The pressure signal does not contain any anomaly and serves as an irrelevant signal concerning its value for an explanation. The dataset is split into 50,000 training and 10,000 test sequences.

7.2 METHOD & EXPERIMENTS

TSViz provides the possibility of interpreting any convolutional network from several dimensions, at different levels of abstraction. This includes the global picture like the types and ordering of different layers present in the network, and their corresponding number of filters, moving to more detailed information like parts of the input that each filter is responding to as well as their importance. Furthermore, filter grouping where filters which are exhibiting similar behavior are clustered together is possible. *TSViz* also uncovers other hidden aspects of the network based on inverse optimization and adversarial examples.

7.2.1 *Backpropagation*

TSViz is based on the principle of backpropagation proposed by Rumelhart et al. [119]. The backpropagation algorithm provides an efficient way to compute the influence of a tensor w.r.t. to another tensor. This capability is leveraged to compute influences for uncovering the deep learning black-box by computing the influence of the input on the current filter, which is the input saliency map.

1 Dataset available at: `https://bit.ly/2UNk0Lo`

Training of a neural network is achieved by reducing the loss function $\mathcal{L} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_+$ ($\mathcal{L} : \mathbb{R}^C \times \mathbb{R}^C \mapsto \mathbb{R}_+$ in case of multi-class classification where C denotes the number of classes). The loss function captures the discrepancy between the network's prediction and the desired output. As the network output is calculated based on the weights and biases of the different neurons involved, these weights and biases are adapted during the learning process to reduce the loss function. This computation of the optimal direction can be obtained by calculating the partial derivatives of the loss function regarding any weight W or bias b as $\partial\mathcal{L}/\partial W$ and $\partial\mathcal{L}/\partial b$.

The learning process can be decomposed into several steps, which include:

1. Feed-forward passes through the network,
2. Backpropagation to the output layer,
3. Backpropagation to the hidden layers,
4. Updating network parameters using an optimizer like SGD

In the forward pass through the network, the output of all the hidden neurons is computed, which is then used for the computation of the final network output and loss. This evaluation is based on the randomly initialized weights. Based on the computed output, the final loss function is evaluated. The networks in deep learning are mostly composed of both convolutional and dense layers. Rectified Linear Unit (ReLU) or some variant of it is commonly used as the non-linearity/activation function. The activation for the dense and convolutional layers is presented in Equation 7.7 and Equation 7.9 respectively. a_j^l denotes the activation of the j^{th} neuron in the l^{th} layer (for dense layers) while a_{ji}^l denotes the activation of the j^{th} neuron in the l^{th} layer at the i^{th} input location (for convolutional layers). k is defined as $\lfloor \mathtt{FilterSize}/2 \rfloor$ for convolutional layers, while $|z_j^{l-1}|$ denotes the number of neurons in the previous layer $l-1$.

$$z_j^l = \sum_{k=1}^{|z_j^{l-1}|} W_{jk}^l a_k^{l-1} + b_j^l \qquad (7.6) \qquad\qquad a_j^l = \max\left(z_j^l, 0\right) \qquad (7.7)$$

$$z_{ji}^l = \sum_{-k}^{k} W_{jk}^l a_{i-k}^{l-1} + b_j^l \qquad (7.8) \qquad\qquad a_{ji}^l = \max\left(z_{ji}^l, 0\right) \qquad (7.9)$$

The error is back propagated to the initial layers, and the gradient regarding the network parameters is computed (weights and biases). The error δ of j^{th} neuron at the output layer L is presented in Equation 7.11.

$$\delta_j^L = \frac{\partial\mathcal{L}}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \qquad (7.10) \qquad\qquad \delta_j^L = \frac{\partial\mathcal{L}}{\partial a_j^L}\max'\left(z_j^L, 0\right) \qquad (7.11)$$

$$\delta_j^l = \left(\left(W_{ij}^l\right)^T \delta_j^{l+1}\right) \odot \max'\left(z_j^l, 0\right) \qquad (7.12) \qquad\qquad \max'(x,0) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \qquad (7.13)$$

Once the gradient of the loss w.r.t. in the output layer is computed, the error is back propagated to all neurons in the hidden layers using Equation 7.12. The gradient of ReLU is one where the value of input x exceeds zero and remains zero otherwise, as mentioned in Equation 7.13. Similarly, the max-pooling layer has a gradient equal to one wherever the maximum quantity occurs and remains zero otherwise. The rate of change of loss $\mathcal{L}$ w.r.t. the bias and weights in the l^{th} layer is given in Equation 7.14 and Equation 7.15 respectively.

$$\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l \quad (7.14) \qquad \frac{\partial \mathcal{L}}{\partial W_{jk}^l} = a_k^{l-1}\delta_j^l \quad (7.15)$$

After computation of the gradients, the network parameters (weights and biases) are updated in the negative gradient direction, as this is the optimal direction for maximum reduction in the loss.

7.2.2 *Influence Computation*

TSViz contributes to the interpretability of deep learning models designed specifically for time series analysis tasks at different levels of abstraction. The first and one of the most intuitive explanations for any model is based on the influence of the input on the final prediction. Consequently, this influence can also be computed for the intermediate states of the network (both in the forward and the backward direction). There are two different influences that can be computed based on any particular filter in the network. The first influence stems from the fact that the input has an impact on the output of the particular filter, while the second influence is of the filter itself on the final output.

7.2.2.1 *Input Influence*

The first influence originates from the input. This information provides important insights regarding the data points in the input that the network is actually responding to for computation of its output. The information regarding the parts of the input which were responsible for a particular prediction is considered a viable explanation in many scenarios, including domains like self-driving cars [63], finance [68] and medical imaging [165].

This value can be obtained by computing the gradient of the current layer l w.r.t. the input layer. The absolute value of the gradient is used, as the magnitude is of relevance irrespective of direction. The input is denoted as a^0, therefore, this influence of the input can be computed using Equation 7.18.

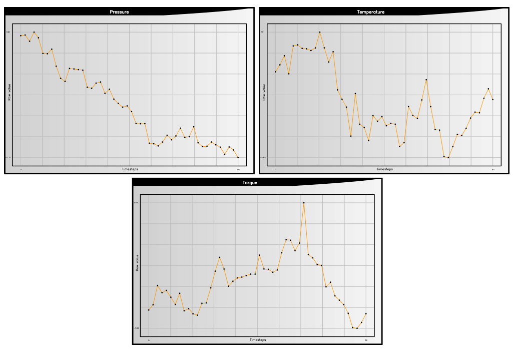
$$\delta_j^l = \frac{\partial a^l}{\partial z_j^l}\frac{\partial z_j^l}{\partial a_j^0} \quad (7.16) \qquad \delta_j^0 = \frac{\partial a^l}{\partial a_j^0} \quad (7.17)$$

$$I_{input}^d = |\delta^0| \quad (7.18)$$

To be able to visualize and compare the saliency of the different filters, the absolute values of the influences are taken using the min-max scaling presented in Equation 7.19.

$$I_{input} = \frac{I_{input} - \min_j I_{input}^j}{\max_j I_{input}^j - \min_j I_{input}^j} \quad (7.19)$$

Figure 7.1a visualizes a sample of an anomalous input in the *Anomaly* dataset. Figure 7.1b equips the raw filter output with the saliency information to provide


(a) Input

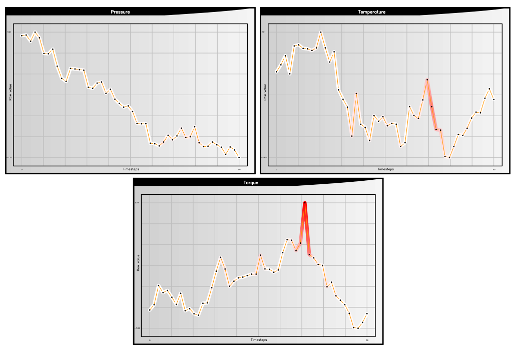

(b) Input with importance

Figure 7.1: A particular anomalous example provided to the network from the *anomaly* dataset. The left figure (a) visualizes the raw input, while the figure on the right (b) equips the raw signal with the saliency information.

a direct interpretation of its utility. It is evident from the figure that the network focused on sudden peaks present in the input to mark the sequence as anomalous. This saliency is computed using Equation 7.18 after applying min-max normalization.

7.2.2.2 *Filter Importance*

Another interesting influence originates from the output, which can be leveraged to compute the filter's influence. This information about filter importance provides hints regarding the filters that were most influential for a particular prediction. Interestingly, many of the filters in the network contribute nothing to a particular prediction. This information is complementary to the information regarding parts of the input that were responsible for a particular prediction.

To obtain this influence, the gradient of the output layer L w.r.t. the current layer l is computed. This provides a pointwise estimate about how each value impacts the output activation a^L. In this case, again, both positive and negative influences are equally important. As the overall influence of a particular filter is of interest. Therefore, pointwise influence estimates can be reduced to a single value by computing the Manhattan or 1-norm of the influence vector. Computing the 1-norm of the influence vector retains the information encapsulated in the vector, by taking the sum of the absolute influences of each of its components, which provides a good estimate regarding the overall importance of the filter. Equation 7.21 provides the mathematical formulation of the influence of layer l on the final output a^L.

$$\delta_j^l = \frac{\partial a^L}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \qquad (7.20)$$

$$I_{\mathrm{output}}^l = \sum_j |\delta_j^l| \qquad (7.21)$$

Figure 7.2a visualizes the third convolutional layer of the network trained on the *Internet Traffic* dataset. Figure 7.2b enhances the filter view by including the

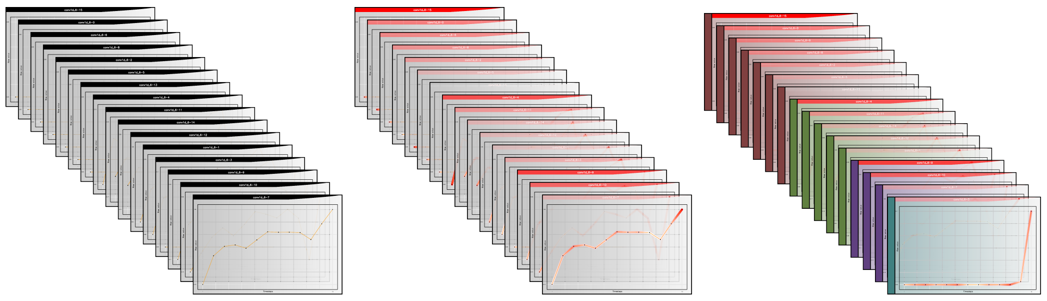

(a) Filters (b) Filters with importance and saliency (c) Filters with importance, saliency and clusters

Figure 7.2: Filters of the third convolutional layer of the network trained on the *Internet Traffic* dataset where the first the raw filters (a) are visualized, followed by additional information regarding the filter importance and saliency information (b), finally adding the information regarding the filter cluster (c).

filter importance information computed using Equation 7.21 after applying the min-max normalization along with the input saliency information.

Proposition 1 (Zero influence). *For extremely confident predictions (with probability of either zero or one) in case of classification, the influence dies off.*

Proof. Let $\mathbf{x} \in \mathbb{R}^D$ be the input and $\hat{\mathbf{y}} \in \mathbb{R}$ ($\hat{\mathbf{y}} \in \mathbb{R}^C$ in case of multi-class classification where C denotes the number of classes) be the prediction by the system. For binary classification, the output probability is usually obtained by the application of sigmoid activation, while in the case of multi-class classification, the output probability is obtained by the application of the softmax activation function. This activation function can be considered as the last layer L in the network. The gradient of the sigmoid and the softmax activation function w.r.t. to its input is presented in Equation 7.22 and Equation 7.23 respectively.

$$\delta^L = \hat{y}(1-\hat{y}) \quad (7.22) \qquad \delta^L_j = \begin{cases} \hat{y_i}(1-\hat{y_i}), & \text{if } i = j \\ \hat{y_i}(-\hat{y_j}), & \text{otherwise} \end{cases} \quad (7.23)$$

In the case of extremely confident predictions, the system makes binary predictions where the probability either goes to zero or one i.e. $\hat{y} \in \{0,1\}$ ($\hat{y} \in \{0,1\}^C$ in case of multi-class classification where C denotes the number of classes). Therefore, during backpropagation, the gradient dies off due to multiplication by zero as highlighted in Equation 7.22 and Equation 7.23 (either the first term or the second term goes to zero due to presence of saturated values). This results in no gradient to previous layers for the computation of the filter influence or saliency for that matter. □

A rudimentary way to overcome the problem of obtaining no influence values (Proposition 1) is to employ temperature augmented softmax in multi-class classification settings by using $T > 1$ as the temperature (Equation 7.24). The temperature T also negatively impacts the gradient, perturbing the actual influence. Therefore, the value of T should be kept close to one to make sure that the overconfident predictions are avoided along with a minor impact on the computed influences.

$$\hat{y}_i = \frac{\exp(z_i^L/T)}{\sum_j \exp(z_j^L/T)} \tag{7.24}$$

A preferable way to avoid these overconfident predictions is to add activity regularization on the final activation where the network penalizes large activation values, thus avoiding extremely confident predictions. This formulation extends to both binary and multi-class classification settings. Therefore, the updated optimization problem can be written as:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x},y)\in\mathcal{X}\times\mathcal{Y}} \mathcal{L}(\phi(\mathbf{x};\mathcal{W}),y) + \lambda\|\mathcal{W}\|_2^2 + \beta\|z^L\|_2^2 \tag{7.25}$$

z^L is denoted as the activation value of the last layer and $\mathcal{W} = \{W^l, b^l\}_{l=1}^L$ encapsulates all the parameters of the network. It is important to note that this formulation requires tuning of yet another hyperparameter, i.e. β to achieve reasonable performance while simultaneously avoiding overconfident predictions.

7.2.3 *Filter Clustering*

Deep networks are great at exploiting redundancy [29], and therefore, it is important to get a measure of the diversity in the network. To capture this diversity, a filter clustering was performed. This clustering phase helps to discover the distinct types of filters present in the network as a notion of the diversity it attained during training. The filters are clustered based on their activation pattern, i.e. filters with similar activating patterns are essentially capturing the same concept. This clustering is also helpful in reducing the information overload for the user in the visualization phase, where only the most salient filters from each cluster can be visualized.

As only the similarity between the activation pattern rather than the actual magnitude and the shifting of the activation pattern (e.g. invariance to the activation at the start or at the end of the peak) is of interest, first the activations of the different filters in a particular layer are aligned. Since the network contains only one dimensional signals (each filter outputs a one dimensional activation vector), therefore, to compute the similarities between the filters, a technique which is very common in the time series analysis community for alignment called as Dynamic Time Warping (DTW) [123] is leveraged.

Each filter is encoded via its activation vector $\mathbf{a} \in \mathbb{R}^d$ where d denotes the dimensionality of the activation. First, the algorithm creates a distance matrix between the every two activation vectors, $\mathbf{a_m} \in \mathbb{R}^d$ and $\mathbf{a_n} \in \mathbb{R}^d$. The distance matrix is called DTW where $\mathrm{DTW}[i,j]$ gives the distance between the activation vectors $\mathbf{a}_m^{1:i}$ and $\mathbf{a}_n^{1:j}$ with the best alignment. The DTW matrix can be effectively computed by a consistent application of Equation. 7.26 where Euclidean distance has been employed as the distance metric $D(i,j)$.

$$\mathrm{DTW}[i,j] = D(\mathbf{a}_m^i, \mathbf{a}_n^j) + \min(\begin{bmatrix} \mathrm{DTW}[i-1,j] \\ \mathrm{DTW}[i,j-1] \\ \mathrm{DTW}[i-1,j-1] \end{bmatrix}) \tag{7.26}$$

$$D(\mathbf{a}_m^i, \mathbf{a}_n^j) = \|\mathbf{a}_m^i - \mathbf{a}_n^j\|_2^2 \tag{7.27}$$

Once the DTW matrix is computed, $\mathrm{DTW}[d,d]$ can be used as a measure of the minimum possible distance to align the two activation vectors where d is the dimensionality of the activation vectors. Therefore, the distance is used to cluster the activation vectors together.

When it comes to clustering, K-Means appears to be the most common choice for any problem. However, K-Means operates with Euclidean distance as the distance metric. Switching to DTW as a distance metric with K-Means results in either unreliable results or even convergence issues. Therefore, a hierarchical (agglomerative) clustering is performed using DTW where clusters which are the closest in terms of distance are combined to yield a new cluster during every iteration of the algorithm until all the data points are combined into one cluster. There are different possible linkage methods that can be employed to compute the distance between the clusters. In this case, the complete linkage is used to compute the distance between clusters. Complete linkage finds the maximum possible distance $d_{CL} \in \mathbb{R}$ between the two clusters G and H using pairs of points i and j from G and H respectively, such that the distance $d_{ij} \in \mathbb{R}$ between the selected points is maximum. This is highlighted in Equation 7.28.

$$d_{CL}(G,H) = \max_{i \in G,\, j \in H} d_{ij} \tag{7.28}$$

The system builds a dense hierarchy of clusters. Since it has a dense hierarchy, the user can navigate the hierarchy, slicing the y-axis at different points to obtain a different number of clusters. The silhouette score for each possible number of clusters is computed, and then the initial number of clusters is selected to be at the point where the maximum silhouette score is obtained. This process is illustrated in Figure 7.3.

Figure 7.2c provides a depiction of equipping the filters with cluster information trained on the *Internet Traffic* dataset. This clustering strategy was also tested on the *Anomaly* dataset and visualized the resulting clusters in a grid view for clarity. This visualization is presented in Figure 7.4. It can be seen that the silhouette method did a reasonable job in selecting the initial number of clusters.

7.2.4 *Inverse Optimization*

An understanding of the input parts deemed most important for a particular prediction is of tremendous value. This information is partially highlighted by computing the input influence. However, this input influence is not very stable. Therefore, the inverse optimization-based framework is used to discover the main sources of variation learned by the network in the input space. The approach starts from a random input and modifies the signal until the desired output is achieved.

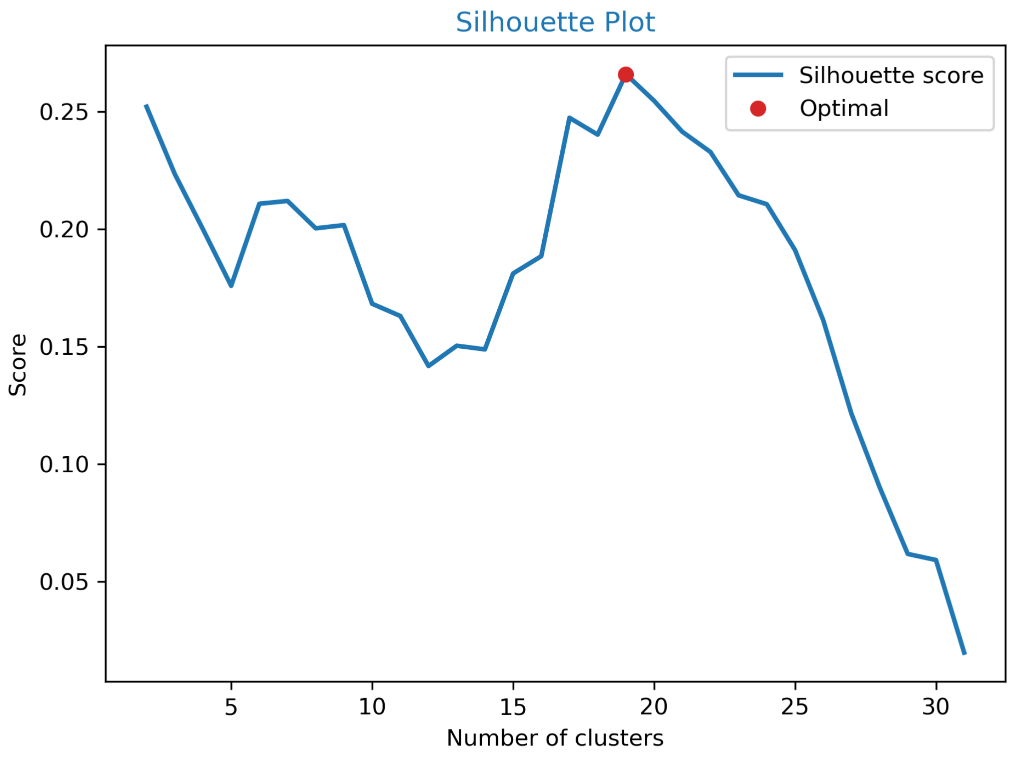


Figure 7.3: Silhouette plot for deciding the initial number of clusters for the second convolutional layer on the *Anomaly* dataset.

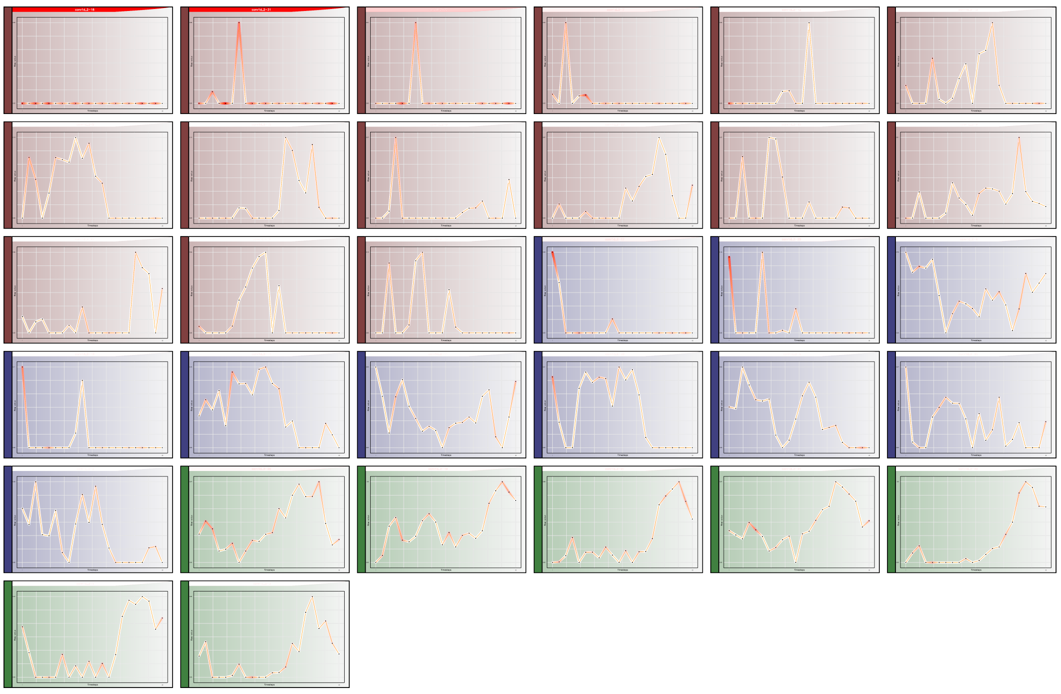
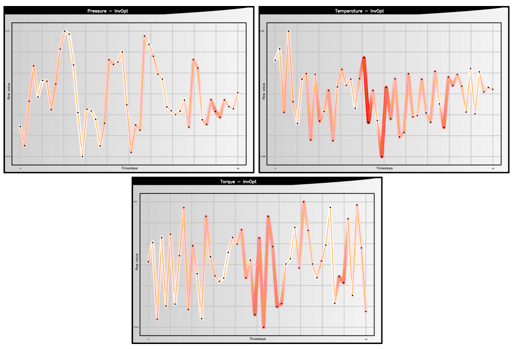


Figure 7.4: Grid view of all the filters equipped with the cluster information (second convolutional layer of the network trained on the *Anomaly* dataset). Different shades represent different clusters.


(a) Start (8.34% conf)

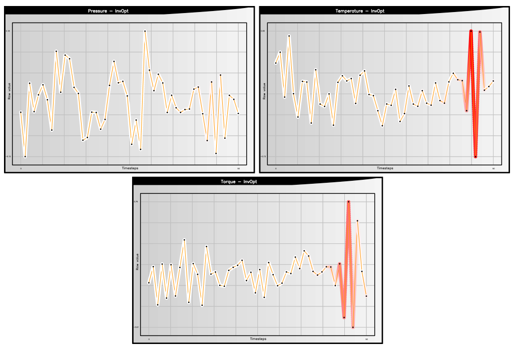

(b) End (99.88% conf)

Figure 7.5: Application of the inverse optimization framework for the classification use case. The left figure (a) demonstrates the random starting position which was classified as non-anomalous with a probability of 91.66% while the figure on the right (b) highlights the final series obtained after optimization which was classified as anomalous with a probability of 99.88%.

During training, the loss is used to minimize the discrepancy between the prediction $\hat{y} \in \mathbb{R}$ and the ground truth $y \in \mathbb{R}$, where both $\mathbf{x} \in \mathbb{R}^D$ and $y \in \mathbb{R}$ are fixed ($\mathbf{y} \in \mathbb{R}^C, \hat{\mathbf{y}} \in \mathbb{R}^C$ in case of multiclass classification where C denotes the number of classes). The optimization problem to discover the optimal parameters of the network $\mathcal{W}^*$ can be written as:

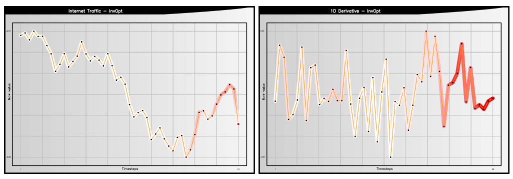
$$\mathcal{W}^* = \underset{\mathcal{W}}{\arg\min} \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x},y) \in \mathcal{X} \times \mathcal{Y}} \mathcal{L}(\phi(\mathbf{x}; \mathcal{W}), y) \tag{7.29}$$

ϕ is defined as the mapping from the input space $\mathbf{x} \in \mathbb{R}^D$ to the label space $\hat{y} \in \mathbb{R}$ ($\hat{\mathbf{y}} \in \mathbb{R}^C$ in case of multiclass classification) parameterized by the network weights $\mathcal{W} = \{W^l, b^l\}_{l=1}^{L}$. Once the network is trained, i.e. it has $\mathcal{W}^*$, the problem can be inverted to discover the input $\hat{\mathbf{x}} \in \mathbb{R}^D$ which produces the same output as a particular time series. This helps in the discovery of the main sources of variation learned by the network. The following optimization problem can be expressed as Equation 7.30. This problem is solved iteratively using gradient descent, as indicated in Equation 7.31.

$$\hat{\mathbf{x}}^* = \underset{\hat{\mathbf{x}} \in \mathbb{R}^D}{\arg\min} \mathcal{L}(\phi(\hat{\mathbf{x}}; \mathcal{W}^*), y) \tag{7.30}$$

$$\hat{\mathbf{x}}^*_{t+1} = \hat{\mathbf{x}}^*_t - \alpha \frac{\partial \mathcal{L}(\phi(\hat{\mathbf{x}}; \mathcal{W}^*), y)}{\partial \hat{\mathbf{x}}^*_t} \tag{7.31}$$

This optimization problem can be efficiently solved again using the backpropagation algorithm. Initially, $\hat{\mathbf{x}}^*_0 \sim \mathcal{N}(0, I)$ is sampled from a standard normal distribution. Since the input is initialized randomly from a normal distribution, the initial time series looks distorted. This initialization is visualized in Figure 7.5a for the classification use case. Upon passing this randomly initialized series to the trained anomaly detection model, it marked the sequence as non-anomalous (8.34% probability for the sequence belonging to the anomalous class). Next, this sequence is optimized as mentioned in Equation 7.31 using gradient descent. The final sequence obtained after optimizing the input for 1,000 iterations ($t = 1,000$) with $\alpha = 0.01$ is visualized in Figure 7.5b. Since the original dataset contained point anomalies, the network introduced point anomalies in


(a) Start (Forecasted value: -0.49923)

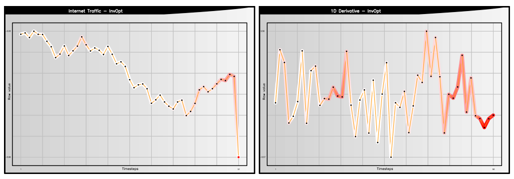

(b) End (Forecasted value: -0.95739)

Figure 7.6: Application of the inverse optimization framework for the regression use case. The left figure (a) demonstrates the random starting position which resulted in a forecasted value of -0.49923 while the figure on the right (b) highlights the final series obtained after optimization which resulted in a forecasted value of -0.95739.

the initial time series to convert the non-anomalous sequence to an anomalous one, with a probability of 99.88%. It is interesting to note that since none point anomalies were introduced in the pressure signal, during inverse optimization, the network also left the pressure signal intact with only minor changes.

Similar tests were performed on the *Internet Traffic* dataset (time series forecasting). The seemingly unimportant channel containing the first-order derivative based on the saliency map was the main factor that the network nudged to obtain the same output as the time series visualized in Figure 7.6a. The inversely optimized series is visualized in Figure 7.6b. The last time step in the original signal (internet traffic) indicates the forecasted value.

7.2.5 *Adversarial Examples*

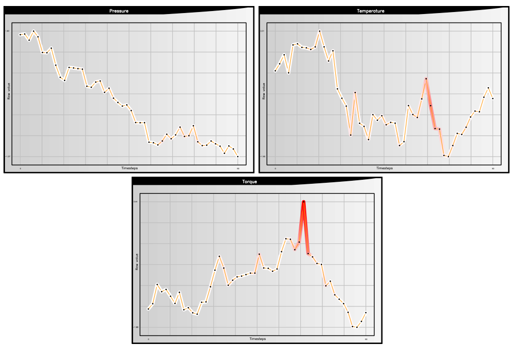
Deep learning models have been discovered to be highly prone to adversarial noise [52]. This problem has been very well-studied in prior literature, specifically for image-based classification networks [59]. The experiments below were conducted using Fast Gradient Sign Method (FGSM) [69] (iterative variant of FGSM [52]) attack on the studied time series.

In situations where the network is not highly susceptible to adversarial noise, this optimization step can help answer a more interesting question, regarding the network's interpretation of parts of the input that with very minor perturbation can bring maximal change to the output. This highlights the network's understanding of the input parts that were mainly responsible for a particular prediction.

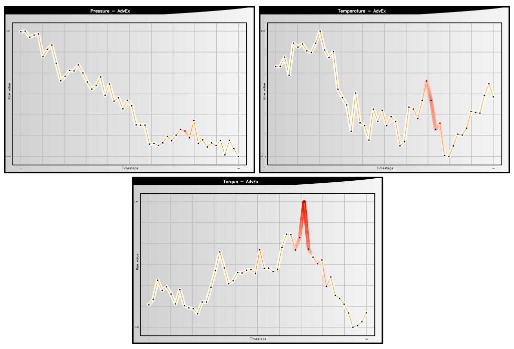
The FGSM attack performs a single step of optimization to obtain an adversarial example. The adversarial example is denoted as $\mathbf{x}^{\mathbf{adv}} \in \mathbb{R}^{D}$. The FGSM optimization problem can be represented as:

$$\mathbf{x}^{\mathbf{adv}} = \mathbf{x} + \epsilon \; \text{sign}\left(\frac{\partial \mathcal{L}(\phi(\mathbf{x}; \mathcal{W}^{*}), y)}{\partial \mathbf{x}}\right) \tag{7.32}$$

The iterative FGSM, instead of solving a single step of optimization, performs iterative refinement of adversarial noise, therefore significantly boosting the chances of producing a successful adversarial example. This optimization problem can be written as:

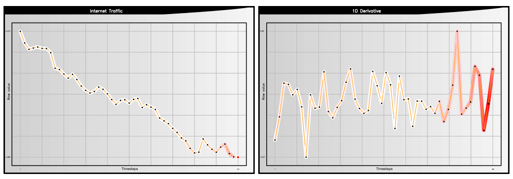


(a) Input (95.76% conf)

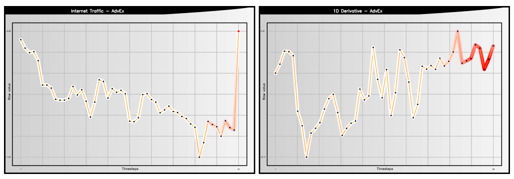


(b) Adversarial output (35.55% conf)

Figure 7.7: Discovered adversarial examples for the classification use case. The left image (a) highlights the input which was classified as anomalous with a probability of 95.76% while the figure on the right (b) indicates the discovered adversarial example which was classified as non-anomalous with a probability of 64.45%.



(a) Input (Forecasted value: -0.97946)



(b) Adversarial output (Forecasted value: -0.27947)

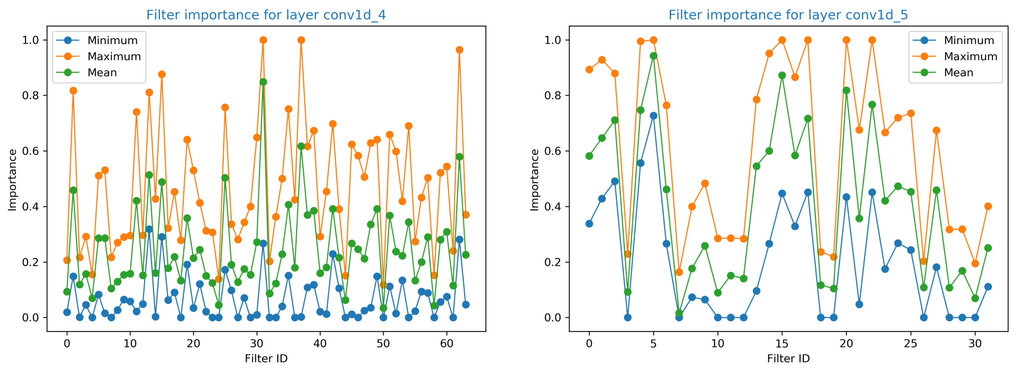
Figure 7.8: Discovered adversarial examples for the regression use case. The left image (a) highlights the input which resulted in a forecasted value of -0.97946 while the figure on the right (b) indicates the discovered adversarial example which resulted in a forecasted value of -0.27947.

$$\mathbf{x}_{t+1}^{\mathbf{adv}} = \text{Clip}_{\mathbf{x},\epsilon}\left\{\mathbf{x}_t^{\mathbf{adv}} + \alpha\ \text{sign}\left(\frac{\partial\mathcal{L}(\phi(\mathbf{x}_t^{\mathbf{adv}};\mathcal{W}^*),y)}{\partial\mathbf{x}_t^{\mathbf{adv}}}\right)\right\} \tag{7.33}$$

$\text{Clip}_{\mathbf{x},\epsilon}$ bounds the magnitude of the perturbation to be within $[-\epsilon,\epsilon]$ from the original example $\mathbf{x}$. The value of the original example $\mathbf{x}$ is used to initialize the initial adversarial example $\mathbf{x}_0^{\mathbf{adv}}$.

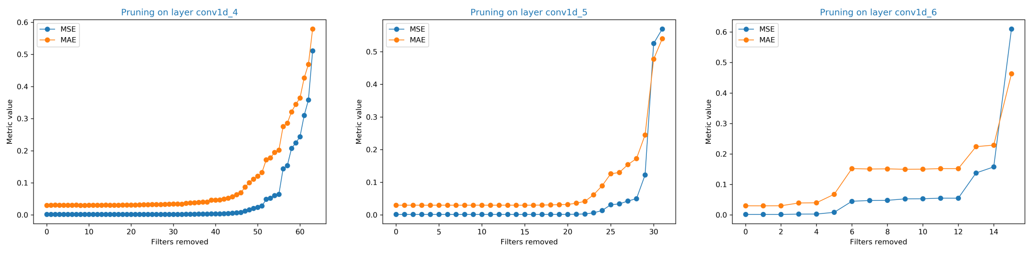
Figure 7.7a visualizes an original anomalous sequence present in the *Anomaly* dataset. The network successfully marked the sequence as anomalous with a probability of 95.76%. Next the iterative FGSM attack is performed using Equation 7.33, with $\alpha = 0.0001$, $\epsilon = 0.1$ and $t = 1,000$. The inverse optimized sequence was predicted to be non-anomalous (35.55% probability of it being an anomalous sequence) which is visualized in Figure 7.7b. Consistent with the understanding of the anomaly present in the network, the network reduced the magnitude of the main peak, which was mainly responsible for the anomalous prediction. It is interesting to note that only a slight reduction was required to flip the label, which is indicative of the network's susceptibility to adversarial noise.

The Adversarial impact on the time series regression task was much more profound. The seemingly non-important first-order turned out to be the main



(a) 1st convolutional layer (b) 2nd convolutional layer

Figure 7.9: Min, mean, and max filter importance computed over the entire dataset for the first and second convolutional layer on the *Internet Traffic* dataset.



(a) 1st convolutional layer (b) 2nd convolutional layer (c) 3rd convolutional layer

Figure 7.10: Test set performance of the network after pruning the specified number of filters from the first, second and the third convolutional layer on the *Internet Traffic* dataset.

reason for the network's vulnerability. The network mainly altered its prediction due to a significant change in the first-order derivative rather than the original signal. Figure 7.8 highlights this case. The same parameters were used, i.e. $\alpha = 0.0001$, $\epsilon = 0.1$ and $t = 1,000$. Again, the last time step in the original signal (internet traffic) indicates the predicted value.

7.2.6 *Network Pruning*

As a sanity check for the utility of the information contained in the computed influences, pruning of the network was performed based on these computed influences. The filters were pruned based on their influence. The filters with the least influence are pruned first, followed by filters with maximum influence. Since the network should be pruned based on this information, it is important to average this influence value over the entire training set. Therefore, the final influence w.r.t. the output can be written as:

$$I^{l}_{\mathrm{output}} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} I^{l}_{\mathrm{output}}(\mathbf{x}) \tag{7.34}$$

Table 7.2: Influence faithfulness test for the *Internet Traffic* dataset. Shows the Mean Squared Error (MSE) and its increase.

Layer	Influence	MSE	Increase (MSE)
Default	-	0.001608	0.000%
1st Conv	Min	0.001609	0.049%
1st Conv	Max	0.005172	221.608%
2nd Conv	Min	0.001608	0.000%
2nd Conv	Max	0.007728	380.553%
3rd Conv	Min	0.001608	0.000%
3rd Conv	Max	0.130290	8002.298%

The minimum, maximum as well as mean importance of every filter of the first two convolutional layers computed over the entire training set of the *Internet Traffic* dataset are visualized in Figure 7.9. Figure 7.10 visualizes the results of pruning based on these influences. The pruning starts with the least influential filter until only one filter is left. It is important to note that the network was fine-tuned for 10 epochs after the pruning step to adjust the network weights to compensate for the missing filter.

Table 7.2 provides results regarding faithfulness of the computed influences where the corresponding most and least influential filter of a particular layer without any fine-tuning were pruned. For the sole purpose of pruning to accelerate inference and reduce model size, the reader is referred to more sophisticated techniques dedicated to pruning relying on second-order gradient information w.r.t. the loss [97, 142].

7.2.7 *Properties*

The three major desirable properties for any interpretability method are faithfulness, stability and explicitness / intelligibility [82]. The following is an analysis of the *TSViz* framework based on these three properties.

7.2.7.1 *Explicitness/Intelligibility*

Explicitness or intelligibility captures the notion of interpretability of the explanations provided by the system. Both the input and the output modalities are well understood by the humans, but the intermediate representations are not. Therefore, these intermediate representations are interpreted in terms of their influence on input and output. Since both the input and the output space are interpretable for humans, this makes the interpretability of the *TSViz* influence tracing algorithm easy. Adversarial examples and inverse optimization also operate in the input space, making them intelligible.

7.2.7.2 *Faithfulness*

Faithfulness captures the notion of the reliability of the computed relevance. True relevance is subjective and can vary from task to task. The influence of noise can be considered relevant in some cases, but might be counterproductive to consider in others [5]. As a sanity check, which is common in literature, filters can be removed from the network to assess their impact on the final performance [82].

For the regression network trained on *Internet Traffic* dataset, the most and the least influential filters from the first convolutional layer were removed to assess their impact on the final loss. As per expectation, pruning the most influential filter had a strong impact on the final performance as compared to pruning the least important filter.

Figure 7.9 provides a depiction of the filter importance (minimum, maximum and mean importance) computed over the entire training set of the *Internet Traffic* dataset. This mean importance was used to remove the most and the least important filter from each of the three convolutional layers on the network. Table 7.2 summarizes the results for the faithfulness experiment. It is evident from the results that removing the most important filter from a layer had a very significant impact on the performance as compared to pruning the least important filter. Since the weights of the corresponding filter are set to zero, therefore, as the layer ascend hierarchy, the impact of pruning a particular filter was more profound (since it had a direct influence on the result). Pruning also reduces the expected value of the output, resulting in a significantly deviated prediction. These results advocate that the computed influence was indeed faithful.

7.2.7.3 *Stability*

Since the first-order gradient is used to trace the influence due to its direct interpretation for humans, this results in unstable explanations due to noise. Interpretability, therefore, sometimes leads to the wrong conclusion regarding the smoothness of the decision boundary, which is not the case in reality [52]. Most interpretability methods suffer from this inherent weakness due to reliance on first-order gradients [5]. Employing second-order methods can resolve the stability issue, but will make it significantly difficult for humans to comprehend the gained knowledge.

7.2.8 *Implementation*

In this work, a novel three-dimensional framework for visualization and demystification of any deep learning model for time series analysis was developed. The user interface communicates with the back-end, which is exposed as a RESTful Application Programming Interface (API). This decouples the model from the visualization aspect. Even though the focus of this work is on time series data, the system is generally applicable to any deep learning model, as it is only dependent on the effective computation of the gradients.

The first view in the visualization presents the user with an overview of the network. This gives the user a chance to get acquainted with the model in question.

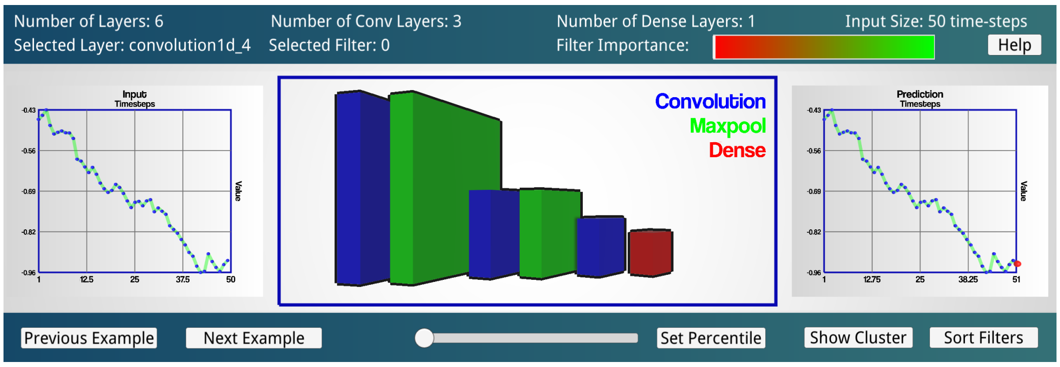


Figure 7.11: Network overview screen for the regression use case

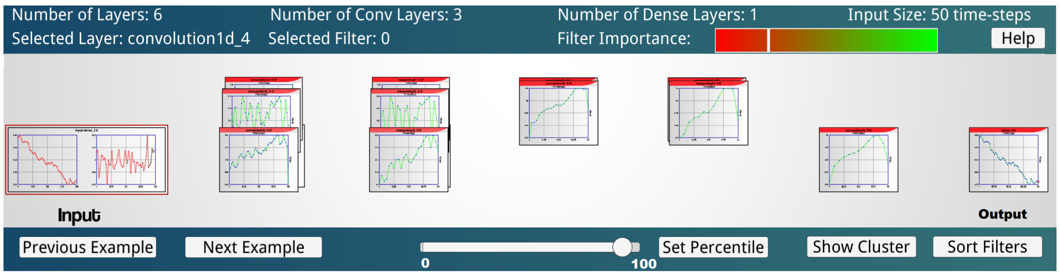


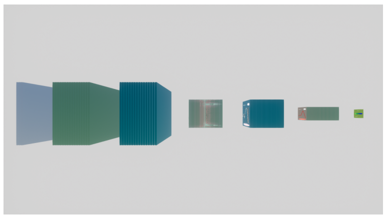
Figure 7.12: Application of the percentile filter on the detailed view (Second level)

A sample visualization of the first screen is presented in Fig. 7.11. The second level provides an overview of the most influential/important filters in each layer, leveraging the influence computation framework. The third view enhances the presented information by clustering the relevant filters together to gain insights regarding the diversity in the network. The second and third views are equally applicable to adversarial examples and inverse optimization outputs, as they only affect the inputs of the model.

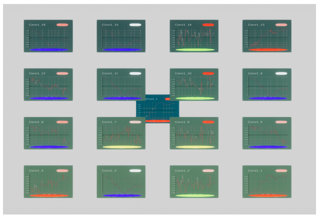
There is usually a high interest in visualizing the most important filters from the network, since they are indicative of the most important parts of the network leveraged for prediction. Therefore, a percentile view was integrated where the user can select the percentile of filters to be viewed based on their importance. This significantly helps in reducing the amount of information presented to the user. Fig. 7.12 provides an example application of the percentile filter onto the second level view of the network in the tool. Another possible way to reduce information overload for the user is to visualize the most salient filters from each cluster.

7.2.8.1 *Virtual Reality*

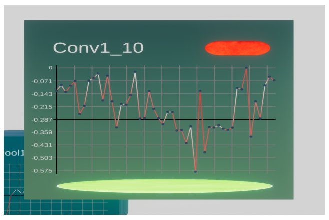
Besides the initial implementation, a virtual reality version that further supports gestures to navigate through the network was developed. Figure 7.13 shows the three different views for the virtual reality framework. The main view in Figure 7.13a shows the complete network in a closed way. The color encodes the layer type and the width, the number of filters. The user can use a specific gesture to open the layer view shown in Figure 7.13b. The layer view shows the different



(a) Main view

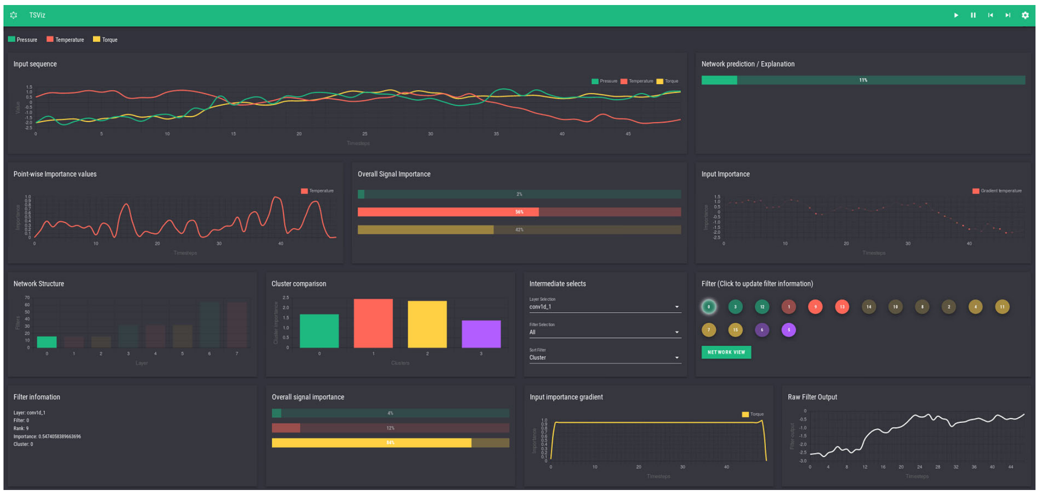


(b) Layer view

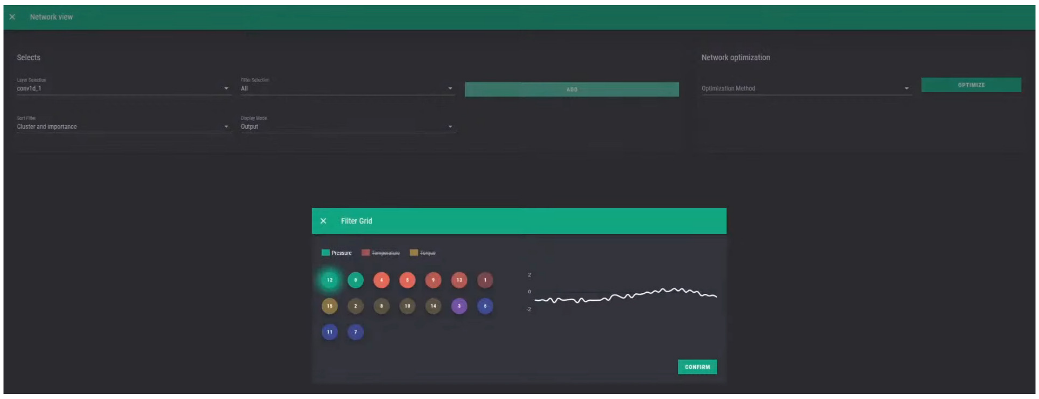


(c) Filter view

Figure 7.13: Virtual reality version of the *TSViz* implementation that supports gestures to move between the filters and layers.


TSViz
Pressure
Temperature
Torque
Input sequence
Network prediction / Explanation
Point-wise Importance values
Overall Signal Importance
Input Importance
Network Structure
Cluster comparison
Intermediate selects
Filter (Click to update filter information)
NETWORK VIEW
Filter infomation
Overall signal importance
Input importance gradient
Raw Filter Output

(a) Main page



(b) Pruning page

Figure 7.14: Web-based dashboard for *TSViz*. Designed in a way that considers user interests and knowledge. From top to down, the experience level required to understand the data increases.

filters, the cluster and the importance of each of those filters. Furthermore, it is possible to focus on a single filter, as shown in Figure 7.13c. This provides the capabilities to have a detailed look at the activation of that filter. Similar to that, it is possible to look at the input and see the input importance.

7.2.8.2 *Dashboard*

Finally, a dashboard version of the framework was created[2]. Figure 7.14 shows the two frames it consists of. The main page shown in Figure 7.14a is divided into three different levels of experience based on the user.

- Level 1 (1st & 2nd row): Shows the input signal and prediction. Furthermore, the input signal importance and the channel importance are shown. This helps to understand which points and input channels were relevant for the prediction. The channel importance is calculated by adding the importance of the points within a channel.
- Level 2 (3rd row): Shows the network structure and the clustering of the filters with their importance. This level provides an initial understanding of the network and can be used to optimize the architecture by pruning.
- Level 3 (4th row): Shows the individual statistics for a selected filter. The user can get information about the input relevance for that individual filter, its activation and its importance and cluster.

It is also possible to switch to the pruning page shown in Figure 7.14b which enables the selection of filters based on their importance. Furthermore, it is possible to select individual filters for each layer and perform a pruning. As soon as the fine-tuning after the pruning step is finished, the dashboard presents updates to the pruned network, and it is possible to interpret and optimize this network further.

7.3 DISCUSSION

The visualization enabled a detailed inspection of the network, which highlighted many different aspects of the network's learning employed in this study.

- Most of the filters in the network were useless, i.e. they contributed nothing to the final prediction for that particular input.
- Many of the filters had very similar activation patterns in the network which were assigned to the same filter cluster. This highlighted the aspect of the lack of diversity in the trained network.
- Despite the improvement in performance with the addition of the first-order derivative of the original signal, most of the filters strongly attended to the original signal as compared to the first-order derivative in the time series forecasting task.
- The network mostly focused on the temperature and torque for detecting the anomalies as no anomalies were introduced in any synthetic anomalies in the pressure signal in the time series classification task.

2 Available at: `https://tsviz.kl.dfki.de/`

Another finding was that there is no perfect way for the interpretability of these models. Therefore, the model is inspected from many different angles to come up with a range of different explanations.

7.4 conclusion

TSViz is a novel framework for interpretability of deep learning-based time series analysis models. The framework enabled an understanding of the model as a parametric function. The different views available within the framework enabled in-depth exploration of the network. These different views include filter importance, filter input saliency map, filter clusters, inverse optimization and adversarial examples. The significance of the computed filter importance was evaluated by pruning filters from the network according to their importance. As there is no one right way to visualize and understand the network, therefore, *TSViz* uncovers all these different aspects to aid human understanding. This exploration will help in the understanding of the network itself, as well as enable new improvements within the architecture with insights gained by uncovering the different aspects of the trained model.

8 TSINSIGHT: A NOVEL TIME SERIES COMPRESSION APPROACH

In this section, the problem of interpretability is approached in a novel way by proposing *TSInsight*, where an autoencoder is attached to the classifier with a sparsity inducing norm on its output and fine-tune it based on the gradients from the classifier and a reconstruction penalty. *TSInsight* learns to preserve features that are important for prediction by the classifier and suppresses those that are irrelevant, i.e. serves as a feature attribution method to boost interpretability. In contrast to most other attribution frameworks, *TSInsight* is capable of generating both instance-based and model-based explanations. The evaluation results show that *TSInsight* naturally achieves output space contraction.

8.1 METHOD

The overview of the proposed methodology is presented in Figure 8.1. As the purpose of *TSInsight* is to explain the predictions of a pretrained model, a vanilla autoencoder is trained on the desired dataset as the first step (indicated as step 1 in the figure). Once the autoencoder is trained, it is attached on top of the pretrained classifier to obtain a combined model. Then the autoencoder is fine-tuned within the combined model using the gradients from the classifier using a specific loss function to highlight the causal/correlated points (indicated as step 2 in the figure). Finally, the attributions are computed from the trained autoencoder (step 3) followed by the sanity check using the suppression test (step 4).

8.1.1 *Pretrained Classifier*

A classifier ($\Phi : \mathcal{X} \mapsto \mathcal{Y}$) is a map from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$. *TSInishgt* requires such a classifier that is trained using standard regularized risk minimization on the given dataset. The training objective can be written as:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x},y)\in\mathcal{X}\times\mathcal{Y}} \mathcal{L}\big(\Phi(\mathbf{x};\mathcal{W}^*), y\big) + \lambda\|\mathcal{W}\|_2^2 \tag{8.35}$$

Φ defines the mapping from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$, while $\mathcal{L}$ corresponds to the classification loss (assumed to be cross-entropy in this work). Furthermore, $\|.\|_p$ represents the L_p norm. Specific instances of L_p norm that are used within this paper are L_1 and L_2 norm. L_1 norm is computed by summing up the absolute values of the given vector ($\|\mathbf{x}\|_1 = \sum_{i=1}^{d} |\mathbf{x}_i|$). Similarly, L_2 norm is computed by taking the square root of the sum of squared values of the given

This chapter is an adapted version of the work presented in: D. Mercier et al. “TSInsight: A Local-Global Attribution Framework for Interpretability in Time Series Data.” In: *Sensors* 21.21 (2021), p. 7373.

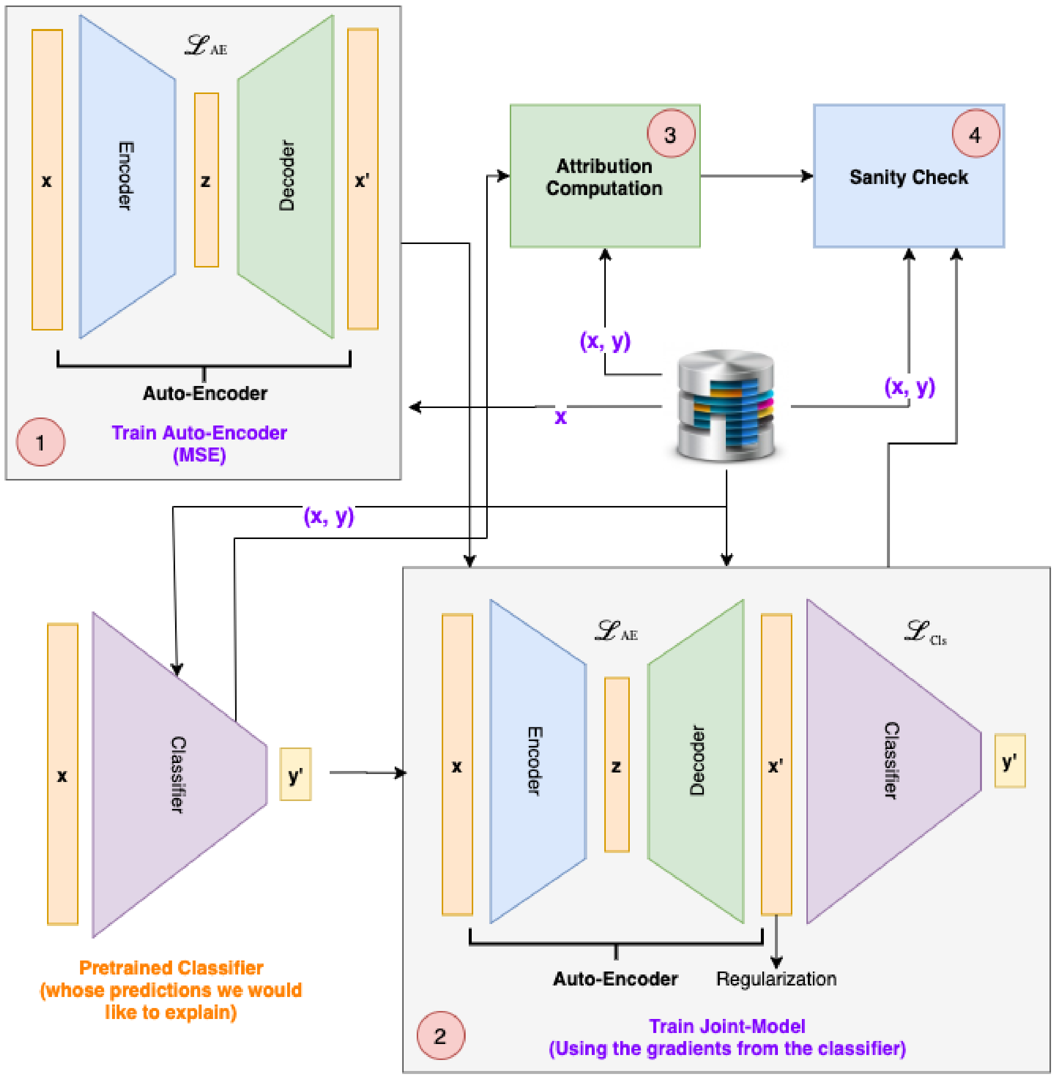


Figure 8.1: Shows the workflow and the performed checks related to the sanity.

vector ($\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{d} \mathbf{x}_i^2}$). The objective also includes a regularization term $\|\mathcal{W}\|_2^2$ with an associated hyperparameter λ to define the relative importance of the classification objective and the simplicity of the hypothesis class. $\mathcal{W}^*$ denotes the final set of parameters obtained after optimization.

8.1.2 *Autoencoder*

An autoencoder ($D \circ E : \mathcal{X} \mapsto \mathcal{X}$) is a neural network where the defined objective is to reconstruct the provided input by embedding it into an arbitrary feature space $\mathcal{F}$, therefore, is a mapping from the input space $\mathcal{X}$ to the input space itself $\mathcal{X}$ after passing it through the feature space $\mathcal{F}$. The autoencoder is usually trained through MSE as the loss function. The optimization problem for an autoencoder is:

$$(\mathcal{W}_E^*, \mathcal{W}_D^*) = \underset{\mathcal{W}_E, \mathcal{W}_D}{\arg\min} \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - D\big(E(\mathbf{x}; \mathcal{W}_E); \mathcal{W}_D\big)\|_2^2 + \lambda\big(\|\mathcal{W}_E\|_2^2 + \|\mathcal{W}_D\|_2^2\big) \quad (8.36)$$

E defines the encoder with parameters $\mathcal{W}_E$ while D defines the decoder with parameters $\mathcal{W}_D$. Similar to the case of the classifier, the autoencoder is trained using regularized risk minimization on a particular dataset.

8.1.3 *Formulation by Palacio et al.*

TSInsight is based on the work of Palacio et al. (2018) [105]. They presented an approach for discovering the preference the network had for the input by attaching the autoencoder on top of the classifier. The autoencoder was fine-tuned using the gradients from the classifier. The new optimization problem for fine-tuning the autoencoder is:

$$(\mathcal{W}_E^{'}, \mathcal{W}_D^{'}) = \underset{\mathcal{W}_E^*, \mathcal{W}_D^*}{\arg\min} \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x},y) \in \mathcal{X} \times \mathcal{Y}} \mathcal{L}\Big(\Phi\Big(D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^*\big); \mathcal{W}^*\Big), y\Big) + \lambda\big(\|\mathcal{W}_E^*\|_2^2 + \|\mathcal{W}_D^*\|_2^2\big) \quad (8.37)$$

$\mathcal{W}_E^*$ and $\mathcal{W}_D^*$ are initialized from the autoencoder weights obtained after solving the optimization problem specified in Equation 8.36 while $\mathcal{W}^*$ is obtained by solving the optimization problem specified in Equation 8.35. This formulation is slightly different from the one proposed by Palacio et al. where they only fine-tuned the decoder part of the autoencoder, the proposed approach updates both the encoder and the decoder as it is a much natural formulation as compared to only fine-tuning the decoder. This complete fine-tuning is significantly more important once advanced formulations are used, since it is beneficial if the network also adapts the encoding to better focus on important features. Fine-tuning only the decoder will change the output without the network learning to compress the signal itself.

8.1.4 *TSInsight: The Proposed Formulation*

In contrast to the findings of Palacio et al. [105] for the image domain, directly optimizing the objective defined in Equation 8.37 for time series yields no interesting insights into the input preferred by the network. Figure 8.2c presents an example from the forest cover dataset. Even though the network was able to reconstruct the anomaly present in the dataset, this resulted in a loss of spatial information. Therefore, instead of optimizing this raw objective, the objective is modified by adding the sparsity inducing norm to the output of the autoencoder. Inducing sparsity on the autoencoder's output forces the network to only reproduce relevant regions of the input to the classifier, since the autoencoder is optimized using the gradients from the classifier. The use of this sparsity

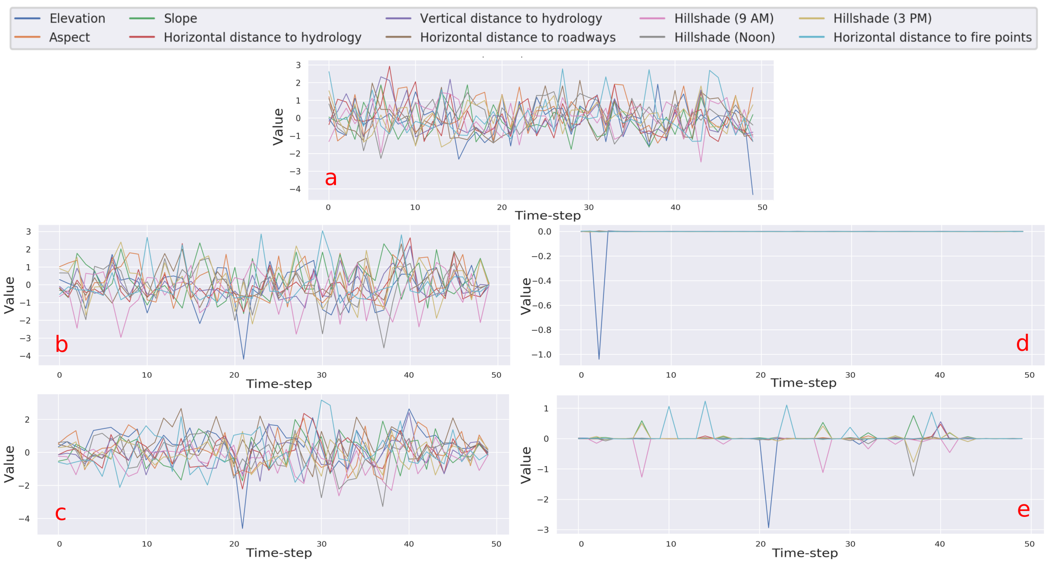


Figure 8.2: Comparison of different autoencoder outputs (a) Original input, (b) Reconstruction from the vanilla autoencoder, (c) Palacio et al. [105] (d) Autoencoder fine-tuned with sparsity, and (e) *TSInsight*.

inducing norm stems from the motivation to obtain the most sparse attribution that retains the prediction. A trivial solution for obtaining attributions is just to predict the whole sequence to be causal/correlated with the prediction if that would have not been the case. However, the attribution obtained in this case would not be useful. Therefore, for human understanding, attributing the prediction to the smallest region possible is important. This has been termed as the complexity of the explanation in the past [115], and the proposed sparsity-based framework focuses on finding the explanation with the least complexity.

However, just optimizing for sparsity introduces misalignment between the reconstruction and the input, as visualized in Figure 8.2d. To ensure alignment between the two sequences, a reconstruction loss was additionally introduced into the final objective. Therefore, the proposed *TSInsight* optimization objective is:

$$
\begin{aligned}
(\mathcal{W}_E', \mathcal{W}_D') = \underset{\mathcal{W}_E^*, \mathcal{W}_D^*}{\arg\min} \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}} \Bigg[\mathcal{L}\bigg(\Phi\Big(D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^* \big); \mathcal{W}^* \Big), y \bigg) \\
+ \gamma \Big(\|\mathbf{x} - D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^*\big)\|_2^2 \Big) + \beta \Big(\|D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^*\big)\|_1 \Big) \Bigg] \\
+ \lambda \big(\|\mathcal{W}_E^*\|_2^2 + \|\mathcal{W}_D^*\|_2^2 \big) \quad (8.38)
\end{aligned}
$$

$\mathcal{L}$ represents the classification loss function which is cross-entropy in this case, Φ denotes the classifier with pretrained weights $\mathcal{W}^*$, while E and D denotes the encoder and decoder respectively with corresponding pretrained weights $\mathcal{W}_E^*$ and $\mathcal{W}_D^*$. Two new hyperparameters, γ and β are introduced in this context. γ controls the autoencoder's focus on reconstruction of the input. β on the other hand, controls the sparsity enforced on the output of the autoencoder. After training the autoencoder with the *TSInsight* objective function, the output is both sparse and aligned with the input, as evident from Figure 8.2e.

The aforementioned hyperparameters play an essential role for *TSInsight* to provide useful insights into the model's behavior. Performing grid search to determine this value is not possible as large values of β results in models which are more interpretable, but inferior in terms of performance, therefore, presenting a trade-off between performance and interpretability which is difficult to quantify. Although during the experiments manual tuning of hyperparameters was superior, this work investigated into the employment of feature importance measures [91, 146] for the automated selection of these hyperparameters (β and γ). The simplest candidate for this importance measure is the saliency:

$$I(\mathbf{x}) = \frac{\partial a^L}{\partial \mathbf{x}} \tag{8.39}$$

L denotes the number of layers in the classifier, and a^L denotes the activations of the last layer in the classifier. This saliency-based importance computation is only based on the classifier. Once the feature importance values are computed, they are scaled in the range of [0, 1] as shown in Equation 8.40 to serve as the corresponding reconstruction weight i.e. γ (Equation 8.41). The inverted importance values then serve as the corresponding sparsity weight, i.e. β as highlighted in Equation 8.42.

$$I(\mathbf{x}) = \frac{I(\mathbf{x}) - \min_j I(\mathbf{x})_j}{\max_j I(\mathbf{x})_j - \min_j I(\mathbf{x})_j} \tag{8.40}$$

$$\gamma^*(\mathbf{x}) = I(\mathbf{x}) \tag{8.41}$$

$$\beta^*(\mathbf{x}) = 1.0 - I(\mathbf{x}) \tag{8.42}$$

Therefore, the objective imposing sparsity on the classifier can be written as:

$$\begin{aligned}\gamma\Big(\|\mathbf{x} - D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^*\big)\|_2^2\Big) + \beta\Big(\|D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^*\big)\|_1\Big) \Rightarrow \\ C \times \|D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^*\big) \odot \beta^*(\mathbf{x})\|_1 \\ + \|\Big(\mathbf{x} - D\big(E(\mathbf{x}; \mathcal{W}_E^*); \mathcal{W}_D^*\big)\Big) \odot \gamma^*(\mathbf{x})\|_2^2\end{aligned} \tag{8.43}$$

$\odot$ corresponds to Hadamard (elementwise) product, evading the need to manually tune the hyperparameters (β and γ). In contrast to the instance-based value of β, the average saliency value is used in the experiments. This ensures that the activations are not sufficiently penalized to significantly impact the performance of the classifier. Due to the low relative magnitude of the sparsity term, it was scaled by a constant factor C. Although a new hyperparameter C has been introduced instead of the two old hyperparameters (β and γ), the value of C can be easily fixed based on the relative contribution of the two terms. $C = 10$ was used in all the experiments.

Table 8.1: Shows the dataset characteristics.

Dataset	Train	Validation	Test	Steps	Channel	Classes
Syn. Anomaly Detection [91]	45,000	5,000	10,000	50	3	2
Electric Devices	6,244	2,682	7,711	50	3	7
Character Trajectories	1,383	606	869	206	3	20
FordA	2,520	1,081	1,320	500	1	2
Forest Cover [139]	107,110	45,906	65,580	50	10	2
ECG Thorax	1,244	556	1,965	750	1	42
WESAD [128]	5,929	846	1,697	700	8	3
UWave Gesture	624	272	3,582	946	1	8

8.2 DATASETS

Several different time series datasets were used in this study. The summary of these datasets is available in Table 8.1. Besides the *Synthetic Anomaly Detection* [91] dataset, all datasets were taken from UEA & UCR repository [9].

The *Forest Cover* dataset [139] has been adapted from the UCI repository[1] for the classification of forest cover type from cartographic variables. The dataset has been transformed into an anomaly detection dataset by selecting only 10 quantitative attributes out of a total of 54. Instances from the second class were considered to be normal, while instances from the fourth class were considered to be anomalous. The ratio of the anomalies to normal data points is 0.9%. Since only two classes were considered, the rest of them were discarded. *WESAD* dataset [128] is a classification dataset introduced by Bosch for person's affective state classification with three different classes, namely, neutral, amusement and stress.

8.3 EXPERIMENTS & RESULTS

This subsection will cover the evaluation to establish the utility of *TSInsight* in comparison to others commonly used attribution techniques.

A commonly used metric to compare model attributions in visual modalities is via the pointing game or suppression test [39]. The suppression test attempts to quantify the quality of the attribution by just preserving parts of the input that are considered to be important by the method. This suppressed input is then passed to the classifier. If the selected points are indeed causal/correlated to the prediction generated by the classifier, no evident effect on the prediction should be observed.

TSInsight was compared against a range of commonly employed attribution techniques. Each attribution method provided an estimate of the features' importance, which was used to suppress the signal. In all the cases, the absolute magnitude of the corresponding feature attribution method was used to preserve the most important input features. For all the methods computing class-specific

1 UCI repository: `https://archive.ics.uci.edu/ml/index.php`

activations maps, the class with the maximum predicted score was used as the target. The descriptions of the methods are provided below:

- None: None refers to the absence of any importance measure.
- Random: Random points from the input are suppressed in this case.
- Input Magnitude: The absolute magnitude of the input is treated to be a proxy for the features' importance.
- Occlusion sensitivity: Over different input channels and positions and mask the corresponding input features with a filter size of three and compute the difference in the confidence score of the predicted class was computed.
- TSInsight: For *TSInsight* the absolute magnitude of the output from the autoencoder of *TSInsight* was treated as features' importance.
- Palacio et al.: Similar to *TSInsight*, the absolute magnitude of the autoencoder's output was used as the features' importance [105].
- Gradient: The absolute value of the raw gradient of the classifier w.r.t. to all the classes was used as the features' importance [68].
- Gradient X Input: The Hadamard (element wise) product between the gradient and the input was computed, and its absolute magnitude was used as the features' importance [136].
- Integrated Gradients: The absolute value of the integrated gradient with 100 discrete steps between the input and the baseline (which was zero in this case) was used as the features' importance [136].
- SmoothGrad: The absolute value of the smoothed gradient was computed by using 100 different random noise vectors sampled from a Gaussian distribution with zero mean, and a variance of $2/(max_j \mathbf{x}_j - min_j \mathbf{x}_j)$ where $\mathbf{x}$ was the input.
- Guided-backpropagation: The absolute value of the gradient provided by *Guided-backpropagation* [134] was used.
- GradCAM: The absolute value of gradient-based class activation map (*GradCAM*) [129] was used as the feature importance measure. Since *GradCAM* visualizes a class activation map, only the predicted class was used for visualization.
- Guided GradCAM: *Guided GradCAM* [129] is a guided variant of *GradCAM*. The absolute value of the *Guided GradCAM* output was used.

8.3.1 *Impact on Accuracy*

The results obtained with the proposed formulation were highly intelligible for the datasets employed in this study. *TSInsight* produced a sparse representation of

Table 8.2: Results for the different datasets in terms of accuracy for both the classifier and *TSInsight*.

Dataset	Model	γ	β	Acc. [%]	Diff. [%]
Synthetic Anomaly	Raw classifier	-	-	98.01	
Detection	TSInsight	1.0	0.001	98.13	+0.12
WESAD	Raw classifier	-	-	99.94	
	TSInsight	2.0	0.00001	99.76	-0.18
Character Trajectories	Raw classifier	-	-	97.01	
	TSInsight	0.25	0.0001	97.24	+0.23
FordA	Raw classifier	-	-	91.74	
	TSInsight	2.0	0.0001	93.26	+1.52
Forest Cover	Raw classifier	-	-	95.79	
	TSInsight	4.0	0.0001	96.26	+0.47
Electric Devices	Raw classifier	-	-	65.14	
	TSInsight	4.0	0.0001	65.74	+0.60
ECG Thorax	Raw classifier	-	-	86.01	
	TSInsight	0.1	0.0001	84.07	-1.94
UWave Gesture	Raw classifier	-	-	91.76	
	TSInsight	4.0	0.0005	92.29	+0.53

the input, focusing only on the salient regions. In addition to interpretability, with a careful tuning of the hyperparameters, *TSInsight* outperformed the pretrained classifier in terms of accuracy for most of the cases, which is evident from Table 8.2. However, it is important to note that *TSInsight* is not designed for performance, but rather for interpretability. Therefore, it is expected that the performance will drop in numerous instances depending on the amount of sparsity enforced.

8.3.2 *Suppression Comparison*

As described before, the performance of different attribution techniques is compared using the input suppression test. Since the input suppression test attempts to suppress an input signal which is not deemed to be important by the attribution, a good attribution method should result in a negligible loss in performance when suppressing the input, specifically when considering some inputs points to be suppressed. The results with different amount of suppression are visualized in Figure 8.3 and Figure 8.4 which are computed based on five random runs.

Since the datasets were picked to maximize diversity in terms of the features, there is no single method which can perfectly generalize to all the datasets. It is evident from the figure that *TSInsight* significantly superseded other methods on four out of eight datasets which include *Character Trajectories*, *ECG Thorax*, *FordA*, and *Synthetic Anomaly Detection* dataset. *Occlusion* sensitivity served as one of the strongest baselines throughout the different datasets as it directly captures the influence of the feature by explicitly masking the input, which is itself quite similar to the suppression test. It is interesting to note that in cases where *TSInsight* was unable to retain high accuracy after suppression, almost all the pure gradient-based methods struggled. As *Guided-backpropagation* overrides the backpropagation phase, it is not considered as a pure gradient-based method [3] which makes it superior in terms of performance as compared to other methods when the gradient is misleading.

It is also interesting to note that for the *WESAD* dataset, none of the most competing methods was in the top list due to the extremely different nature of the dataset. *TSInsight* turned out to be the most competitive saliency estimator on average in comparison to all other attribution techniques tested.

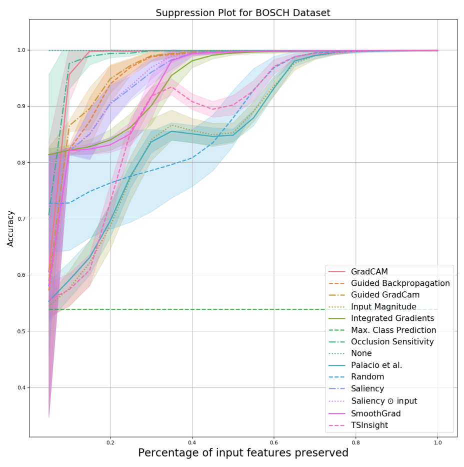
To qualitatively assess the attribution provided by *TSInsight*, an anomalous example from the *Synthetic Anomaly Detection* dataset is visualized in Figure 8.5 along with the attributions from all the commonly employed attribution techniques. Since there were only a few relevant discriminative points in the case of *Forest Cover* and the *Synthetic Anomaly Detection* datasets, *TSInsight* suppressed most of the input, making the decision directly interpretable. This highlights the fact that alongside the numbers, *TSInsight* was also able to produce the most plausible explanations.

8.3.3 *Loss Landscape*

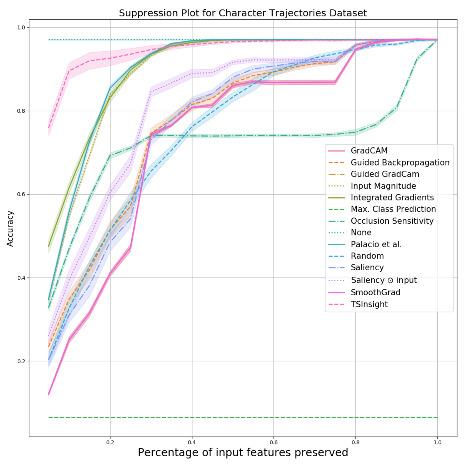
The loss landscape was analyzed to assess the impact of stacking the autoencoder on top of the original network on the overall optimization problem. The experiment follows the scheme suggested by Li et al. [73] where first filter normalization was performed using the norm of the filters. This allows the network to be scale invariant. Then it was sampled into two random directions (δ and η) and a linear combination of these directions was used to identify the loss landscape. The values of the classifier in the combined model were kept intact, since those parameters are treated as fixed. The function representing the manifold can be written as:

$$f(\alpha, \beta) = \mathcal{L}(\theta^* + \alpha\delta + \beta\eta) \qquad \forall \alpha, \beta \in \{-1.0, -0.95, ..., 0.95, 1.0\} \tag{8.44}$$

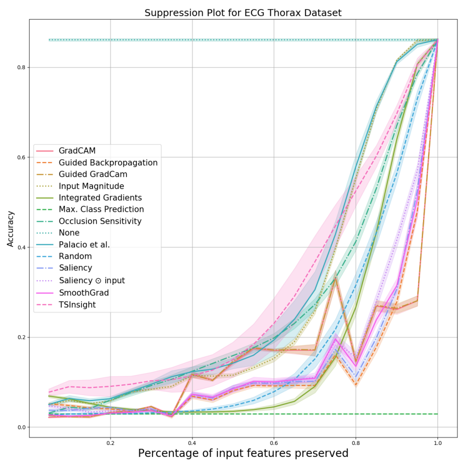
It was iterated over different values of α and β from -1 to +1 with a fixed step size. Once the loss function was evaluated for all the values of α and β (4,000 different combinations), the resulting function was plotted as a three-dimensional surface. This loss landscape for the model trained on forest cover dataset is visualized in Figure 8.6. The surface at the bottom (mostly in blue) signifies the loss landscape for the classifier. The landscape is nearly convex around the local minima found during the optimization. The surface on the top is from the model coupled with the autoencoder. It can be seen that the loss landscape has a kink at the optimal position but remains flat otherwise with a significantly higher loss value. This indicates that the problem of optimizing the autoencoder using gradients from the classifier is a significantly harder one to solve. This is consistent with the observation where the network failed to converge in numerous instances.



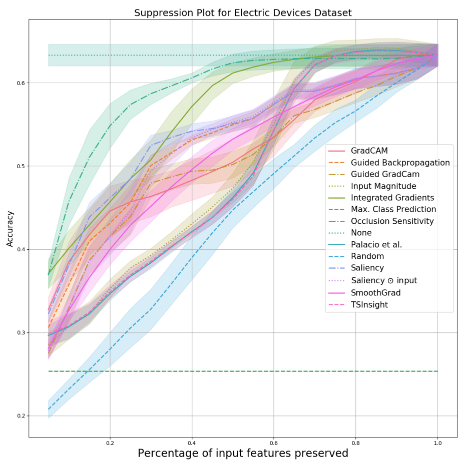
(a) WESAD



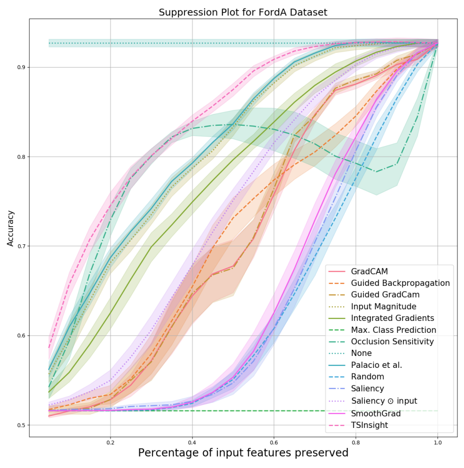
(b) Character Trajectories



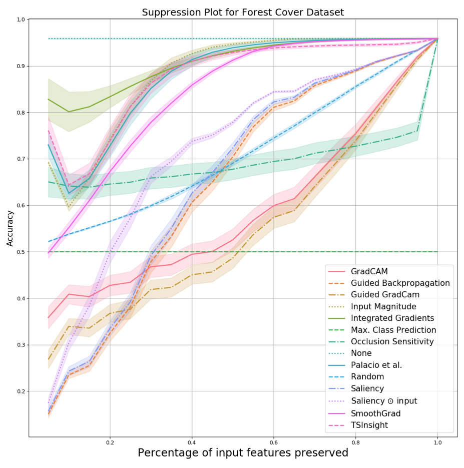
(c) ECG Thorax



(d) Electric Devices

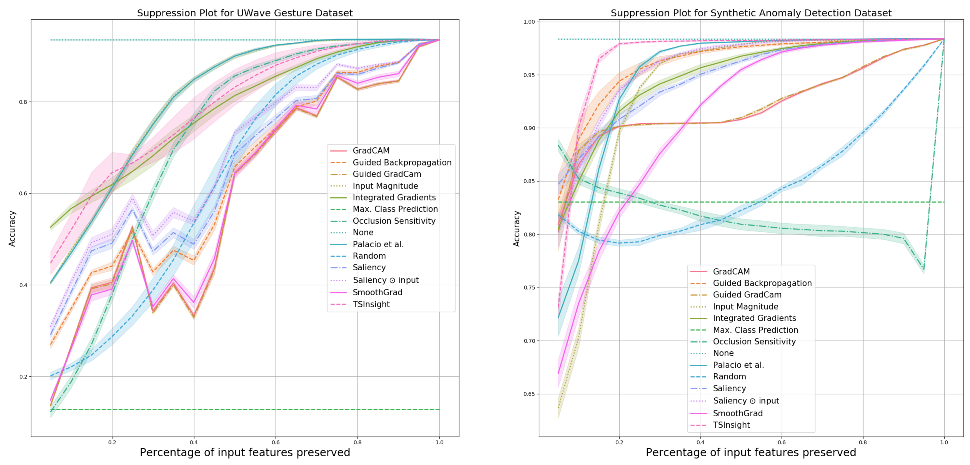


(e) FordA



(f) Forest Cover

Figure 8.3: Suppression results (1/2) against numerous baseline methods computed using five random runs.



(a) UWave Gesture (b) Synthetic Anomaly

Figure 8.4: Suppression results (2/2) against numerous baseline methods computed using five random runs.

Similar observations have been made by Palacio et al. [105] where they failed to fine-tune the complete autoencoder, resorting to only fine-tuning of the decoder to make the problem tractable. The results were very similar when tested on other datasets.

8.3.4 *Autoencoder's Jacobian Spectrum Analysis*

Figure 8.7 visualizes the histogram of singular values of the average Jacobian on test set of the *Forest Cover* dataset. The spectra of the formulation from Palacio et al. [105] and *TSInsight* were compared. It is evident from the figure that most of the singular values for *TSInsight* were close to zero, indicating a contraction being induced in those directions. This is similar to the contraction induced in contractive autoencoders [116] without explicitly regularizing the Jacobian of the encoder.

8.4 DISCUSSION

TSInsight is compatible with any base model. The method was tested with two prominent architectural choices in time series data, i.e. Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM). The results highlight that *TSInsight* was capable of extracting the salient regions of the input regardless of the underlying architecture. It is interesting to note that since LSTM uses memory cells to remember past states, the last point was found to be the most salient. For CNN on the other hand, the network had access to the complete information, resulting in equal distribution of the saliency. A visual example is presented in Figure 8.8.

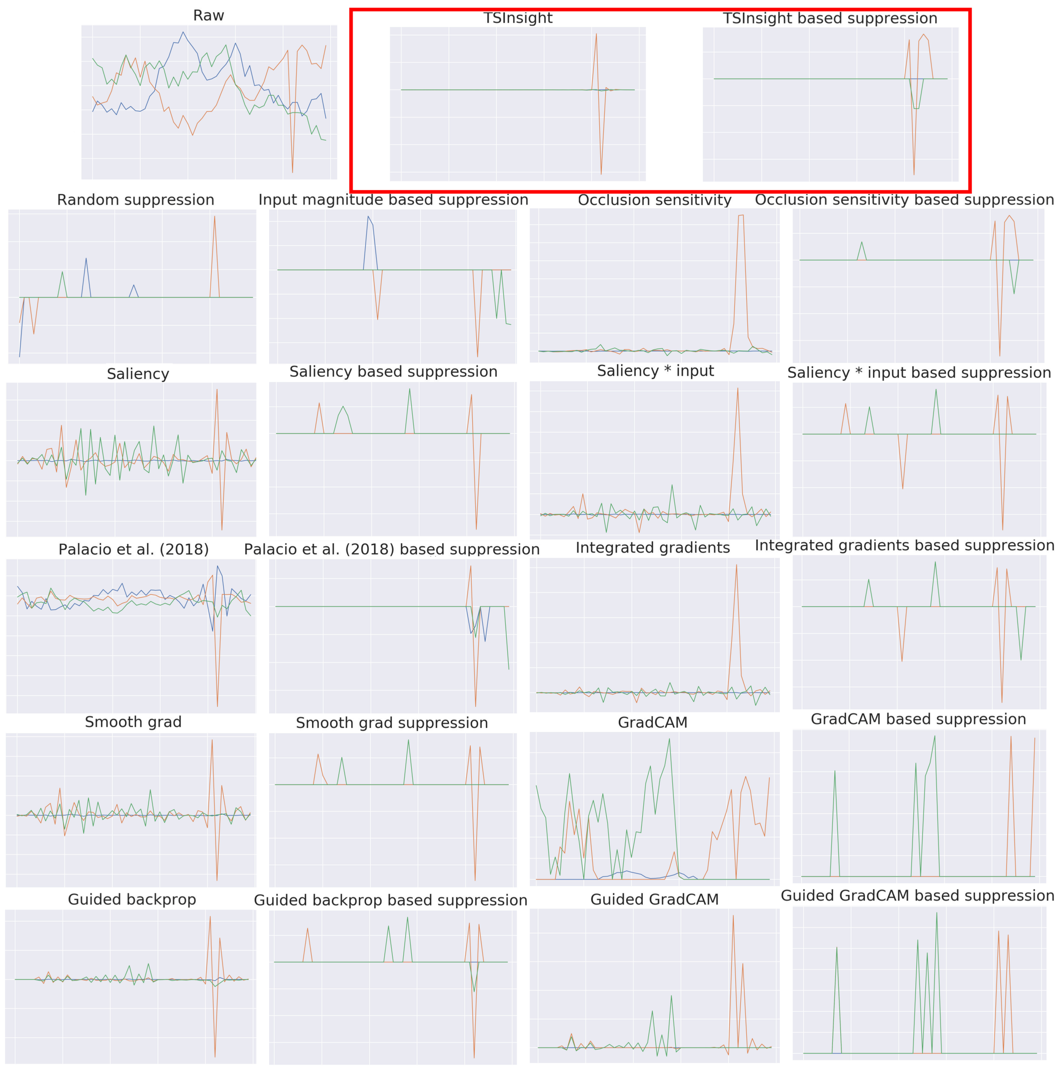


Figure 8.5: Output from different attribution methods as well as the input after suppressing all the points except the top 5% highlighted by the corresponding attribution method on an anomalous example from the *Synthetic Anomaly Detection* dataset (best viewed digitally). All methods were able to correctly identify the anomalous spike given the simplicity of this dataset. However, qualitative differences exist between different methods.

Below some of the interesting properties that *TSInsight* achieves out-of-the-box which includes output space contraction, its generic applicability and model-based (global) explanations are discussed:

- Adversarial Robustness: Since *TSInsight* induces a contraction in the input space, this also resulted in slight gains in terms of adversarial robustness. However, these gains were not consistent over many datasets and strong adversaries, therefore, omitted for clarity here. In depth evaluation of adversarial robustness of *TSInsight* can be an interesting future direction.

- Model-based vs Instance-based Explanations Since *TSInsight* poses the attribution problem itself as an optimization objective, the data based on

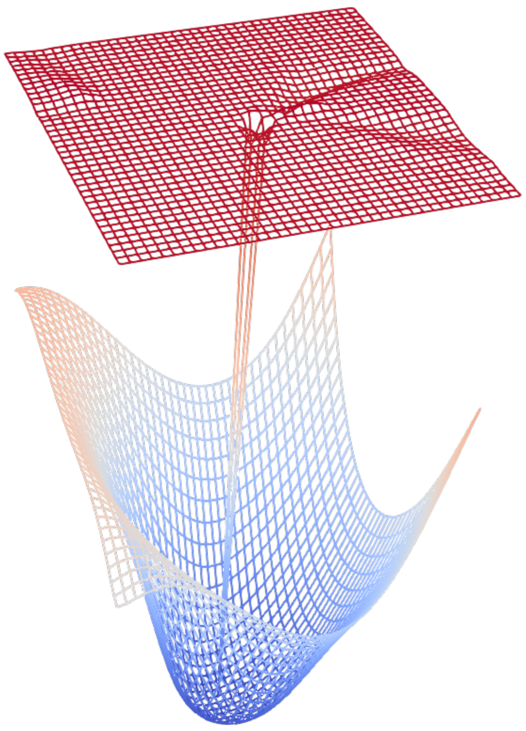


Figure 8.6: Loss landscape where the bottom surface indicates the manifold for the classifier while the surface on the top indicates the manifold for the autoencoder attached to the classifier.

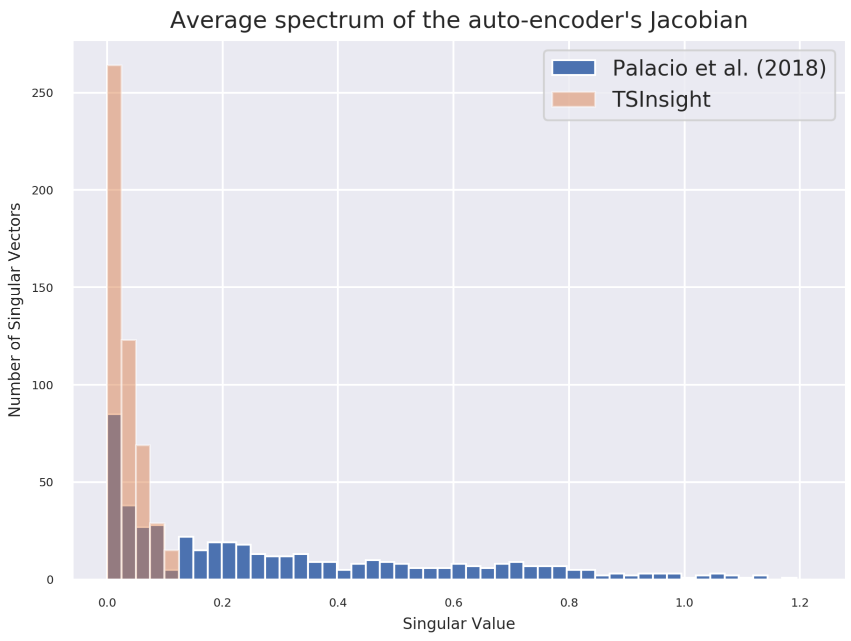


Figure 8.7: Spectrum analysis of the autoencoder's average Jacobian computed over the entire test set of the *Forest Cover* dataset. The sharp decrease in the spectrum for *TSInsight* suggests that the network was successful in inducing a contraction of the input space.

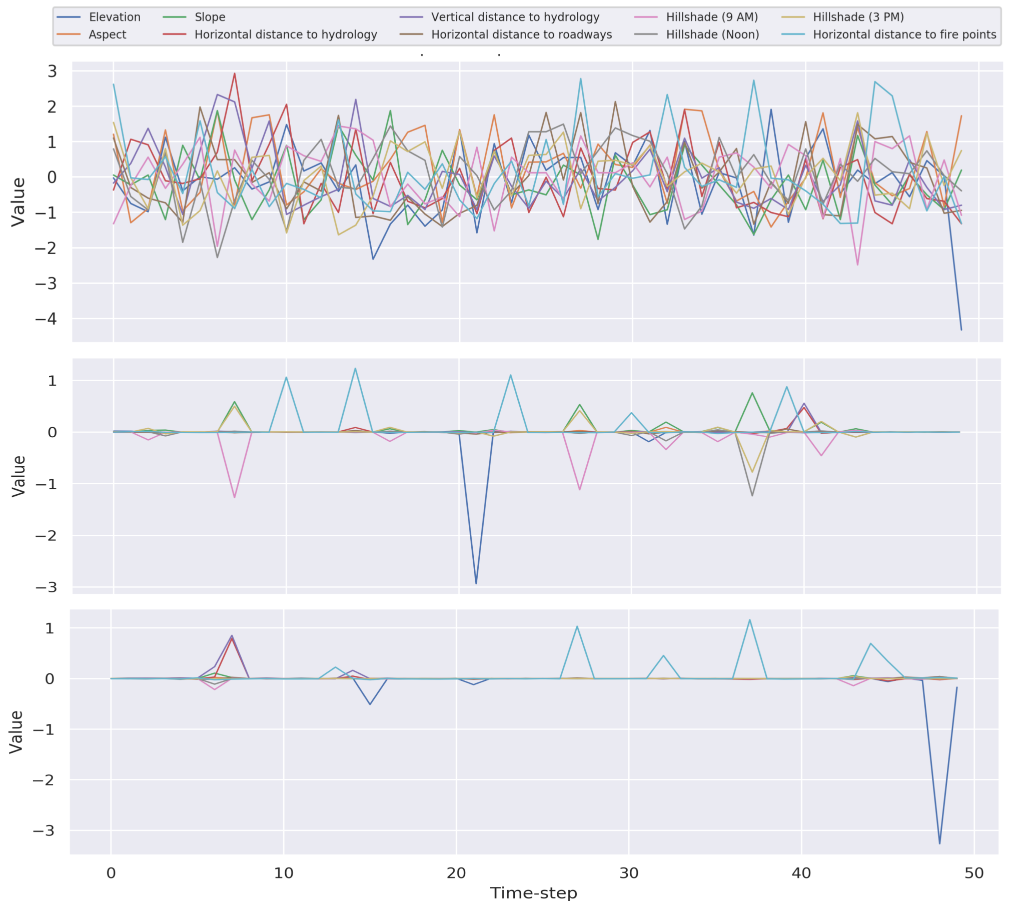


Figure 8.8: Autoencoder training with different base models: (a) Raw signal (b) TSInsight attribution for Convolutional Neural Network (CNN) (c) TSInsight attribution for Long Short Term Memory (LSTM).

which this optimization problem is solved defines the explanation scope. If the optimization problem is solved for the complete dataset, this tunes the autoencoder to be a generic feature extractor, enabling extraction of model/dataset level insights using the attribution. In contrary, if the optimization problem is solved for a particular input, the autoencoder discovers an instance's attribution. This is contrary to most other attribution techniques, which are only instance-specific.

8.5 CONCLUSION

This section presented a novel method to discover the salient features of the input for the prediction by using the global context. With the obtained results, it is evident that the features highlighted by *TSInsight* are intelligible as well as reliable at the same time. In addition to interpretability, *TSInsight* also offers off-the-shelf properties which are desirable in a wide range of problems. Interpretability is essential in many domains, and this method opens up a new research direction for interpretability of deep models for time series analysis. One major limitation of the current approach is the difficulty in tuning the hyperparameters (γ and

β) which offers a good compromise between the final accuracy of the classifier and the interpretability of the model. It is non-trivial to define a simple scoring measure, since interpretability itself is a subjective attribute.

9 DATA LENS: BENCHMARKING OF STATE-OF-THE-ART INFLUENCE FUNCTIONS

Most work covers instance- and compression-based attribution, however, the identification of input data points relevant for the classifier has recently spurred the interest of researchers for both interpretability and dataset debugging. E.g. analyzing mislabels is a valuable task to understand how well influence functions work and detect dataset biases that affect the model in its reasoning process. Also, the analysis of results provides insights concerning the generalization capabilities of the classifiers. The experiments in this section show the use of *Influence Functions* and *Representer Point* as two approaches for dataset debugging and interpretability.

9.1 DATASETS

Subjectivity and cherry-picking are two major challenges for explainability methods. To provide evidence for the methods and prove the correctness of the experiments, it is important to conduct experiments using different datasets. Therefore, three different publicly available datasets were used including point anomaly, sequence anomaly, and a classification task. Precisely, the *Character Trajectories* and *FordB* dataset from the UEA & UCR Time Series Classification Repository [9] and the *Synthetic Anomaly Detection* dataset from [91] were used. The dataset characteristics are shown in Table 9.1. Furthermore, these datasets cover both binary and multi-class classification tasks and come with different sequence lengths and a different number of channels to achieve the largest possible variation of properties.

9.2 EXPERIMENTS & RESULTS

Different experiments were conducted to shed light on several aspects concerning debugging rates, accuracy, time consumption, and interpretability. Besides a random selection used as a baseline and the network loss representing a direct measure, two well-known network interpretability methods that claim to improve mislabel correction namely the *Influence Functions* [66] and the *Representer Point* [157] were used. Finally, a comparison of the advantages and drawbacks of the used methods is presented. To create the datasets for the debugging, some labels were flipped within the dataset.

This chapter is an adapted version of the work presented in: D. Mercier et al. "Interpreting deep models through the lens of data." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

Table 9.1: Used Datasets and characteristics.

Dataset	**Train**	**Val**	**Test**	**Steps**	**Channels**	**Classes**
Syn. Anomaly Detection [91]	45,000	5,000	10,000	50	3	2
Character Trajectories	1,383	606	869	206	3	20
FordB	2,520	1,091	810	500	1	2

9.2.1 *Mislabel Correction Approaches*

To understand the debugging priority, the ranking mechanisms excluding the *Random* and *Loss* approach are explained as they are intuitive. Firstly, the *Influence Functions* [66] were used to provide negative and positive values to highlight harmful and helpful samples. Therefore, the most harmful, most helpful, and most influencing samples can be inspected. In addition, the influence scores were computed for each class individually (classwise) or over the complete set. Secondly, the representer values [157] were used that only provide information about inhibitory (low) and excitatory (high) samples.

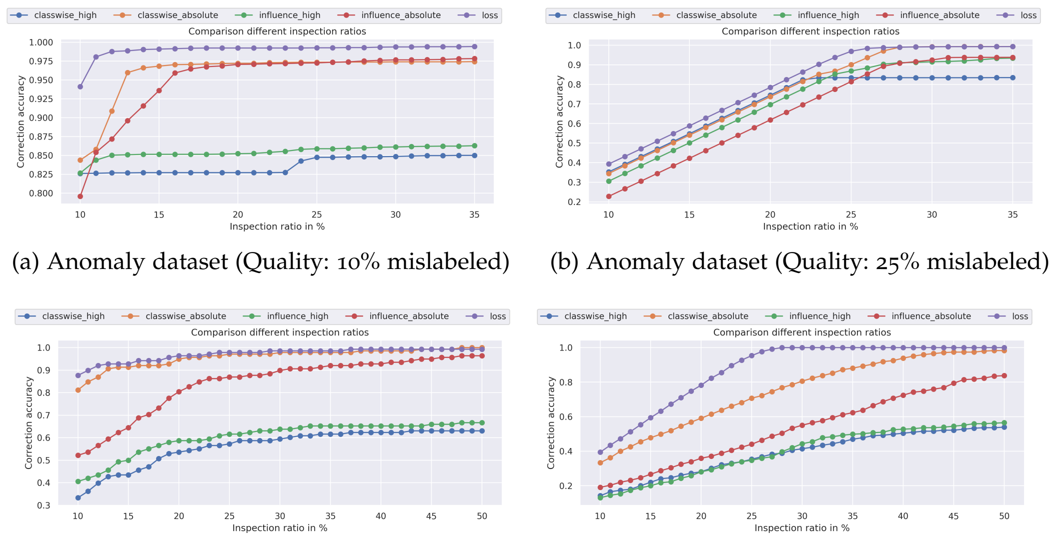
9.2.2 *Mislabel Correction Performance*

Although the process of finding possible mislabeled data can be automated, it is essential to achieve good accuracy when searching for mislabels as they have to be validated manually. Table 9.2 shows the correction ratio assuming that a subset of the data selected according to a ranking of the corresponding debugging approach is inspected manually. The best correction rates are highlighted, showing that with the increasing amount of mislabeled data, the model performance decreases up to a point where the model cannot learn the concept anymore and collapses. Intuitively, a model that does not learn the concept should be rather meaningless for the approaches that try to cover the debugging task, as they operate directly on the model using the learned concept.

Surprisingly, by looking at the second last column, the *Loss* approach achieved excellent correction accuracies, except for the two models that did not learn the concept correctly. One would expect that the more complex methods, using the model to draw detailed conclusions, outperform the *Loss* as they have additional access to more complex computations. Therefore, these results emphasize the use of the training loss for mislabel correction. Against the expectations, the *Influence Functions* outperformed the *Loss*, *Representer*, and *Random* method when the model was unable to learn the concept, indicating that the influence-based approach does not strongly rely on that. Overall, the *Loss* seems to be a suitable approach concerning the correction ratio, but the best correction accuracy does not necessarily lead to the best performance. The mislabels can have more or less impact, and it is mandatory to focus on those with the most impact.

Table 9.2: Detected mislabeled in percentage sorted by different datasets, dataset qualities (percentage of mislabeled data), and inspection (percentage of inspected data).

Dataset	Model Acc.	Mislabeled	Inspected	Detected Mislabels									
				Influence-based [66]						Representer theorem [157]		Loss	Random
				classwise low	classwise high	classwise absolute	low	high	absolute	low	high		
			10	14.34	82.60	84.37	12.48	82.65	79.57	11.74	10.40	**94.11**	9.40
	98.48	10	25	14.54	84.74	97.34	13.17	85.88	97.25	26.20	24.65	**99.25**	24.80
			50	14.74	85.25	97.54	13.45	86.54	98.00	50.71	49.28	**99.60**	49.91
			10	15.92	35.25	34.44	5.82	30.52	22.76	11.04	9.72	**39.29**	4.54
Anomaly	98.33	25	25	16.06	83.39	90.04	5.98	86.84	81.42	25.40	25.13	**96.89**	13.06
			50	16.32	83.68	**99.45**	6.27	93.72	93.93	50.60	49.39	99.42	37.04
			10	16.53	3.35	3.35	3.20	**16.71**	3.35	9.87	9.86	3.35	10.13
	16.97	50	25	41.55	8.30	8.30	8.28	**41.68**	8.30	24.94	24.78	8.30	25.25
			50	**83.06**	16.93	16.93	16.93	**83.06**	16.93	50.08	49.91	16.93	50.02
			10	33.33	33.33	81.15	29.71	40.57	52.17	2.17	38.40	**87.68**	8.69
	94.75	10	25	35.50	57.24	97.10	33.33	61.59	86.95	6.52	57.97	**97.82**	23.91
			50	36.95	63.04	**100.00**	33.33	66.66	96.37	19.56	80.43	99.27	57.97
			10	30.14	14.20	33.33	28.69	13.04	19.13	9.85	7.85	**39.42**	8.98
Character	89.73	25	25	39.13	35.36	70.72	37.97	34.78	44.05	26.66	20.00	**95.36**	27.24
			50	46.08	53.91	98.26	43.47	56.52	83.76	53.62	46.37	**100.00**	52.17
			10	**19.97**	0.57	11.57	**19.97**	0.14	8.24	11.43	6.94	**19.97**	10.56
	88.39	50	25	**49.63**	1.44	29.66	49.49	1.59	16.06	28.94	19.82	**49.63**	25.03
			50	91.17	8.82	57.88	89.86	10.13	35.89	56.00	43.99	**95.80**	49.92
			10	45.66	9.44	29.13	45.66	9.05	30.31	6.29	9.44	**70.86**	11.41
	66.61	10	25	48.03	40.94	64.96	48.03	40.55	57.87	17.71	26.77	**92.51**	25.19
			50	48.42	51.57	99.60	48.42	51.57	**99.60**	46.85	53.14	99.21	48.42
			10	19.49	27.98	19.81	18.86	28.93	28.93	9.90	7.23	**38.52**	9.43
FordB	59.83	25	25	35.53	46.38	51.41	33.01	46.38	58.17	22.64	22.48	**75.31**	22.95
			50	47.32	52.67	93.86	46.22	53.77	78.93	49.05	50.78	**95.44**	49.84
			10	5.34	**14.15**	**14.15**	**14.15**	5.34	**14.15**	10.22	9.11	13.60	9.74
	49.78	50	25	18.94	30.42	30.42	**30.50**	19.10	**30.50**	25.39	24.92	29.71	25.23
			50	48.50	**51.41**	50.70	**51.41**	48.50	50.70	50.07	49.84	51.33	49.84



(a) Anomaly dataset (Quality: 10% mislabeled) (b) Anomaly dataset (Quality: 25% mislabeled)

(c) Character dataset (Quality: 10% mislabeled) (d) Character dataset (Quality: 25% mislabeled)

Figure 9.1: Different correction accuracies for multiple inspection ratios with a fixed dataset quality.

9.2.3 *Influence of Inspection Ratio*

Furthermore, the impact of the inspection rate was analyzed, and it was found that the gain of a higher inspection rate heavily decreases after a certain point as shown in Figure 9.1. The horizontal axis provides the ratio of inspected data after ranking the samples according to the corresponding debugging approach, and the vertical axis shows the accuracy of corrected mislabels. In Figure 9.1a at 10% inspected data, the correction accuracy should be equal to 0.1 for the random correction and should increase linearly. Both figures do not show all measurements, but rather visualize the most successful approaches. The scores in Figure 9.1a provide information about the saddle point for the different methods. Also, for the two measurements considering inspecting the most helpful samples, the overall accuracy of the mislabel correction is much lower compared to the other selected methods. Furthermore, the *Loss* outperformed the other methods at any inspection rate.

In general, Figure 9.1b refines the previous results on a different dataset quality. It has to be mentioned that the *Loss* keeps the superior performance. An evaluation of the 50% mislabeled dataset could not provide meaningful results because the concept was not learned correctly by the model. For a complete analysis and to avoid that the previous finding is related to the properties of the *Anomaly* dataset, the same figures were created for the *Character* dataset because of the diversity of the data and the classification task. In addition, Figure 9.1 shows the correction accuracies for the *Character* trajectory datasets, which reflects that the behavior for the approaches was similar to the results presented for the anomaly dataset.

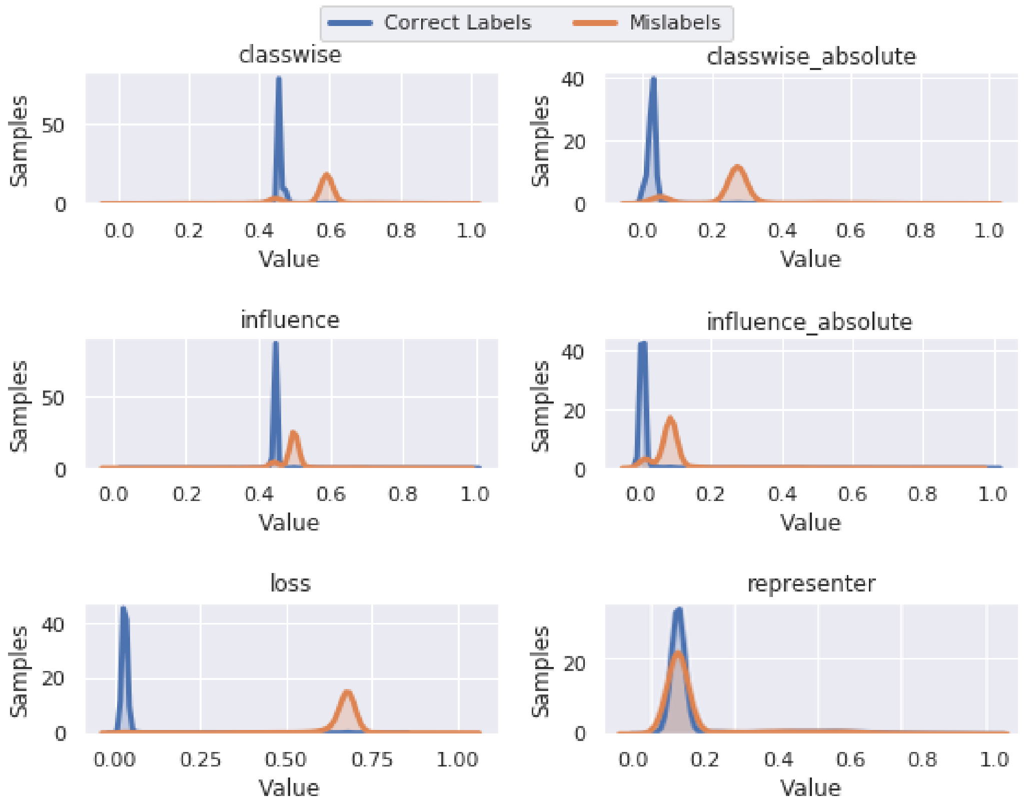


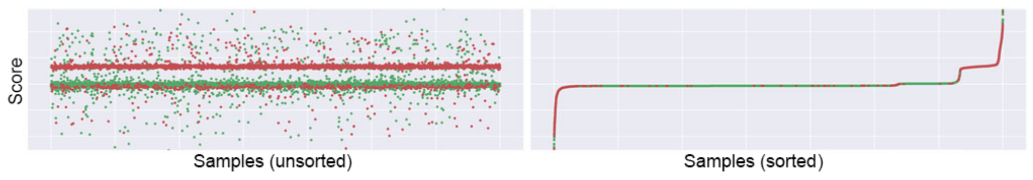
Figure 9.2: Normalized distribution of the different correction approaches for the *Synthetic Anomaly Detection* dataset (Quality: 10% mislabeled.

9.2.4 *Analyzing Score of Correction*

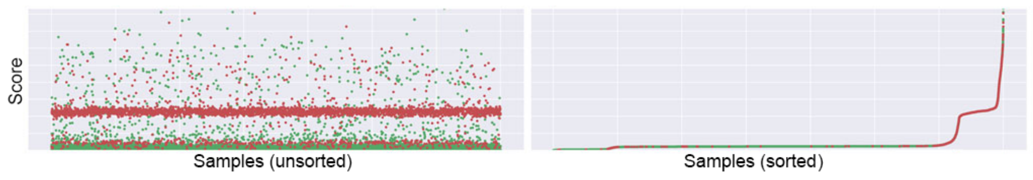
To understand the performance differences, a more detailed look into the distribution and the computed values is mandatory. In Figure 9.2 the distribution of these values has shown that for some methods the distribution highlights the two classes. E.g. the loss-based values showed a clear separation of the correct labels and the mislabels. In contrast to that, the representer values did not separate the data in such a manner. The same holds for the absolute influence values. Besides those two methods, all other methods provided an excellent separation of the data in the distribution plot. Although these plots of the distribution provided a rough understanding of the values, more detailed inspection is provided in the following paragraph.

To better align the findings of the distribution plot, the scores are visualized for each sample in the *Anomaly* datasets (Quality: 10% mislabeled) in Figure 9.3. The right column shows the sorted scores which were used for the experiments and provides a better overview of the separation of the labels.

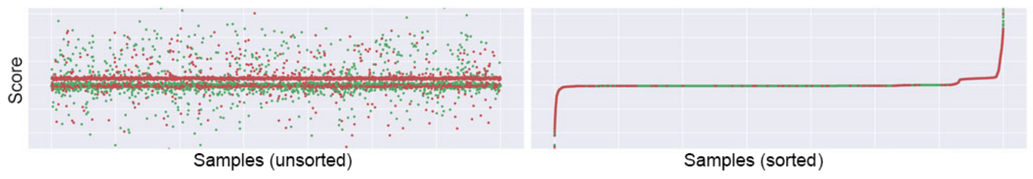
Figure 9.3a shows the scores for the classwise measurement in an unsorted (left) and sorted (right) manner, indicating that selecting the lowest or highest scores can lead to a good mislabel correction. The high values correspond to the helpful whereas the low are harmful samples, and it is possible to improve the quality of those. Figure 9.3b shows the absolute values of this measurement, and therefore it



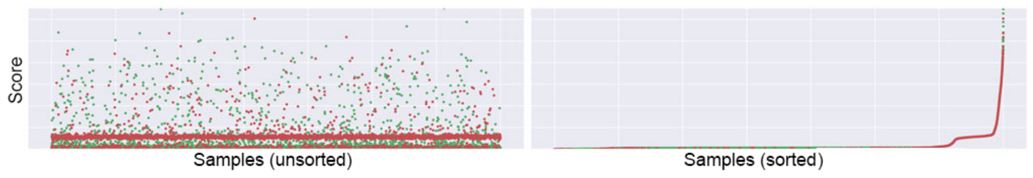
(a) classwise



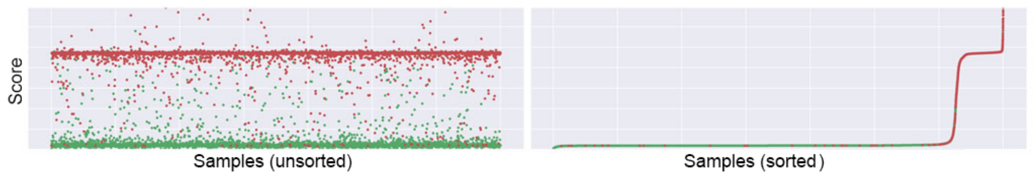
(b) classwise_absolute



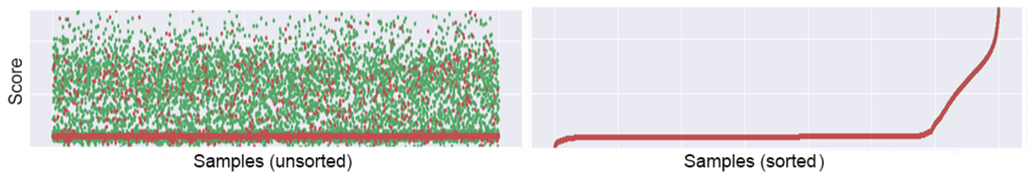
(c) influence



(d) influence_absolute



(e) loss



(f) representer

Figure 9.3: Left column shows the unsorted scores for one of the *Synthetic Anomaly Detection* datasets (Quality: 10% mislabeled).

is not possible to differentiate between helpful and harmful, resulting in a single influence value indicating only the importance concerning the classification.

The approaches shown in Figure 9.3c and Figure 9.3d do not compute the influence separate for each class. This can change the scores for some samples. Especially, if samples are more important for a specific class, this measurement does not capture this property.

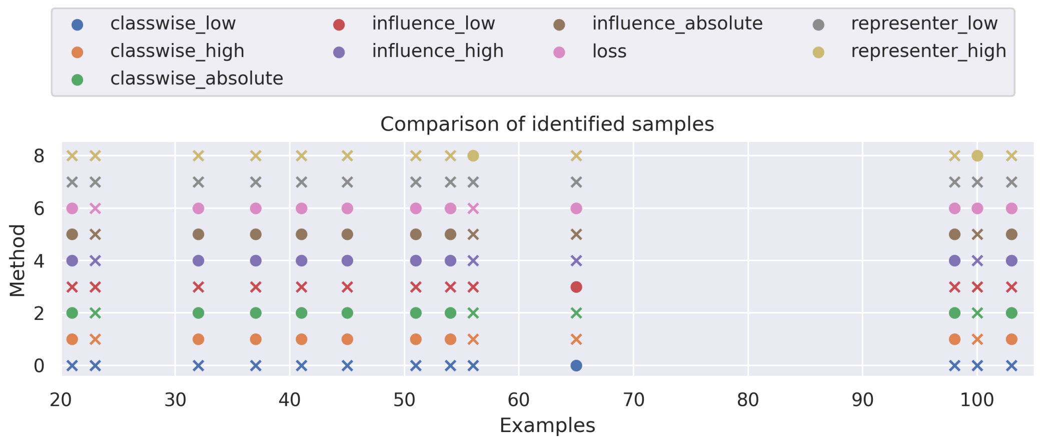


Figure 9.4: First 100 samples of the *Anomaly* dataset (Quality: 10% mislabeled). Dots indicate detected and crosses undetected mislabels.

In Figure 9.3e an almost perfect separation provided by the *Loss* is shown. The loss value for the mislabels is very high compared to the correct-labeled samples, and selecting the samples with a high loss indicates to be an excellent measurement when the learned concept is meaningful.

Finally, Figure 9.3f shows the representer values. The plot on the left side maybe lead to the conclusion that the mislabels have lower scores but inspecting the sorted values proves that this is not the case.

9.2.5 *Identification Differences - Sample Ranking*

Although it was shown that some methods separate the data better, it was decided to have a more detailed look at the samples that are not detected and the samples that are only detected by a specific method because not every sample has the same weight towards the classification accuracy. This is especially of interest when it comes to the classification performance rather than the correction accuracy. In theory, it is a good practice to aim for the highest mislabel correction rate, but this does not mandatory result in the best possible classifier. Therefore, a more detailed inspection of the different detected samples followed by an accuracy evaluation can provide a better understanding of the results as this could favor the *Influence Functions* [66] and *Representer Point* [157] performances.

As shown in Figure 9.4 the approaches detect different mislabels and a combination of the approaches could provide better correction results. For example, the *Representer Point* only detected two out of the 13 mislabels, but one of these was not detected by any other methods. Especially, the *Loss* which detected 11 out of the 13 shown label flips was unable to detect this sample.

Table 9.3: Detected mislabels for the best combinations. The first row of each setup highlights the best performance without any combination, and the following the best combined approaches. Numbers are given in percentage.

Dataset	Mislabeled	Inspected	Corrected	Influence-based [66]						Representer theorem [157]		Loss
				classwise low	classwise high	classwise absolute	low	high	absolute	low	high	
Anomaly	10	10	94.11	-	-	-	-	-	-	-	-	X
			94.25	X	-	-	-	-	-	-	-	X
			94.22	-	-	-	X	-	-	-	-	X
			94.11	-	-	-	-	-	-	X	X	X
	25	25	96.89	-	-	-	-	-	-	-	-	X
			96.94	X	-	-	-	X	-	-	-	X
			96.89	-	-	-	X	X	-	-	-	X
			96.89	-	-	-	-	-	-	X	X	X
	50	50	**83.06**	-	-	-	-	X	-	-	-	-
			83.06	-	-	-	-	X	-	-	-	X
			83.06	-	-	-	-	X	-	-	X	-
			83.06	X	-	-	-	-	-	-	-	X
Character	10	10	87.68	-	-	-	-	-	-	-	-	X
			89.85	X	-	-	-	-	-	-	-	X
			89.85	-	-	X	-	-	X	-	X	X
			88.40	-	-	-	X	-	-	-	-	X
	25	25	95.36	-	-	-	-	-	-	-	-	X
			96.81	X	-	-	-	-	-	-	-	X
			96.23	-	-	-	X	-	-	-	-	X
			95.36	-	-	-	-	-	-	X	X	X
	50	50	95.80	-	-	-	-	-	-	-	-	X
			96.52	X	-	-	-	-	-	-	-	X
			96.09	X	X	-	-	-	-	-	-	X
			95.80	-	-	-	-	-	-	X	X	X

9.2.6 *Combining Correction Approaches*

Concerning previous findings, a combination of the approaches could lead to even better results. To combine the methods, the ranking scores are normalized to make it possible to compare them linearly. Although this combination approach is simple, the results show the capabilities of a combination.

Table 9.3 presents the results for some selected combinations. The results refine the findings that the *Loss*, as a baseline, was excellent, and only in the case where the model did not learn the concept, the *Loss* was significantly worse than the other approaches. Also, this indicates that the combined methods can reach a very stable performance for the 50% mislabeled *Synthetic Anomaly detection* dataset. The results for the *Character Trajectories* dataset were similar to those of the *Synthetic Anomaly Detection* dataset. Besides, the combinations with the *Loss* perform well even for the 50% mislabel due to the correctly learned concept.

Furthermore, these experiments emphasized that a combination can improve the correction accuracy and improve the robustness compared to the use of a single measurement. Nevertheless, drawbacks exist, addressing the computational effort and the robustness. Some methods were not as reliable as the results of the *Loss* and using them can decrease the performance as well.

9.2.7 *Additional Time Consumption*

In contrast to the *Loss*, the other approaches need additional computation time. The training loss can be collected during the evaluation process without a significant slowdown. The *Influence Functions* approach [66] needs an already trained model, and the execution of this method is extremely time-consuming. Especially, the computation of the classwise measurement requires a lot of time. The same holds for the *Representer Point* method [157]. This method needs additional training to learn the representation to compute the representer value based on the pre softmax activations. In contrast to the *Influence Functions*, this additional training is class independent and depends on representation size.

The time consumption is visualized in Figure 9.5 and the *Loss* is excluded. As for the other approaches, the *Representer Point* has very low computational extra time. The computational effort for the influence strongly depends on the dataset size. Also, the computational effort for the classwise measurement suffers from the number of different classes. A comparison of the datasets showed that for the *Anomaly* and *FordB* dataset, the computation time for the classwise measurement increased about 40% for the *FordB* dataset and 50% for the *Synthetic Anomaly Detection* dataset as both have two classes. The *Character Trajectories* dataset has 20 classes, and therefore the increase in additional time is much higher.

9.2.8 *Detailed Sample Analysis*

There are two important questions during the dataset debugging: Why are some samples harder to identify compared to the majority of samples? How do these samples look like and do they provide any information concerning the learned

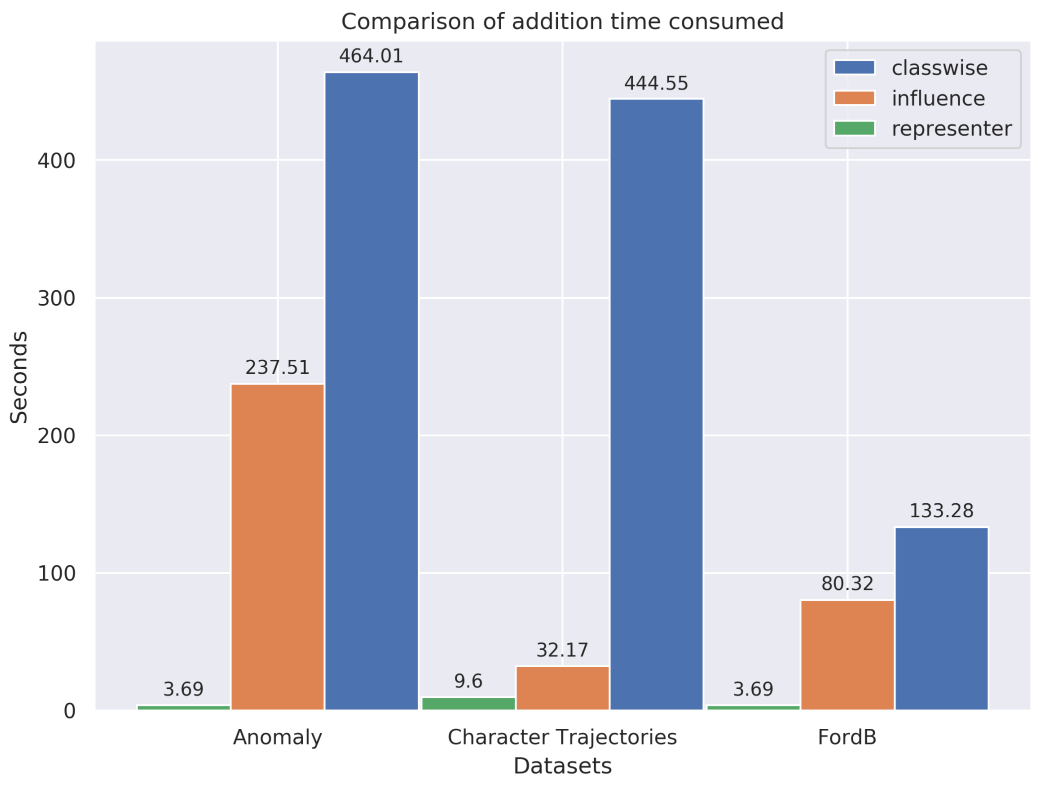


Figure 9.5: Additional computation time excluding any measurement that can be done during the evaluation process.

concept? Answering these questions or inspecting these samples can help to interpret the model. According to the previous findings, not all samples are similarly easy to find. This experiment covers the difficulty and properties of the samples. It has to be highlighted, that the results are visualized for the *Synthetic Anomaly Detection* dataset due to the easier interpretability of the problem but could be visualized for the other datasets as well.

In Figure 9.6 three samples of the previously mentioned slice for the loss-based approach are shown. These samples were selected to emphasize the specific properties of the approach. The label shows the correct label, whereas one corresponds to the anomaly and zero to the non-anomalous class. Therefore, all samples are classified as anomalies within the ground truth. Only the last sample (second row) was found by inspecting 10% of the data, as this includes the ranks 31,500 to 35,000 for the training dataset. The rank reflects the position in the dataset sorted according to a specific measurement, e.g. *Loss*. Furthermore, the second example (first row, right) was close to the threshold, and increasing the amount of inspected data to 12% would be sufficient to find this mislabel. Finally, for the first example (first row, left), there is an ambiguity concerning its ground truth label as it could either be a true mislabel or the model was unable to capture the precise concept of point anomaly concerning the less dominant peak.

According to the dataset creation process, the sample had the correct ground truth label, highlighting that when it comes to the interpretation and explainability of the model, this sample shows that the concept was not precisely

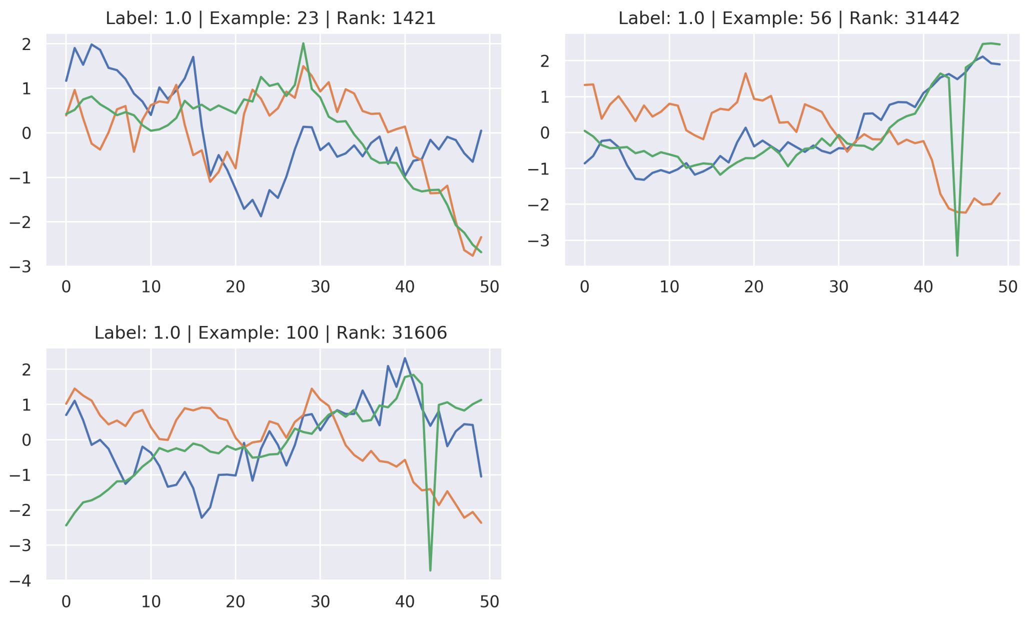


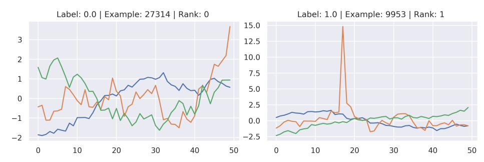
Figure 9.6: All samples are anomalies within the ground truth, but their labels were flipped during the training. Only sample 100 was successfully identified as mislabel by the loss-based approach.

learned. With this information, it is possible to include samples related to the missing concept parts or weight these kinds of samples to adjust the learned concept to cover the complete task.

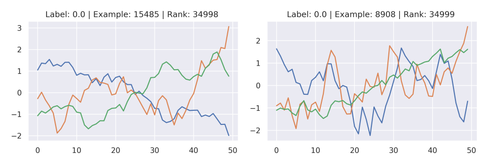
This means, that based on the ranking, it is possible to understand the learned concept and the dataset quality. Both can help to provide an understanding of the model to improve it. Also, the corresponding influence score ranked the ambiguous sample at position 25,556. This information states that the sample was not relevant to the classifier. This assumption was further validated by Figure 9.3 where the influence of the sample is zero. Therefore, it was not helping or harming the classifier's performance. The same result was given by the classwise influence score which had rank 23,197 and following the same procedure showed that this sample did not contribute much to the classifier. Finally, to provide the complete information for that sample, the score for the *Representer Point* which ranked the sample at rank 4,079 was checked and refined the assumption as well.

Using the information above, it is now possible to understand the mislabel, as this sample was not relevant for the classifier. To adjust the classifier to detect peaks like that, it is mandatory to increase the importance of these kinds of samples.

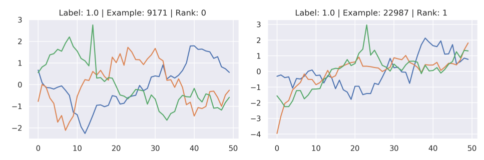
After the first conclusions based on the ambiguous sample, further experiments were conducted in this direction. Therefore, Figure 9.7 provides information about the importance of the samples with the highest and lowest scores. Starting with Figure 9.7a the two samples with the lowest loss are shown. These samples visualize two pretty good samples for the anomaly detection task. Their loss highlights the learned concept. In contrast to that, Figure 9.7b shows the samples with the highest loss. Important for these two samples is that they were mislabeled.



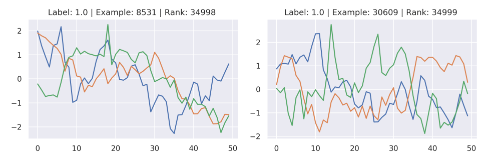
(a) Samples with the lowest loss (labeled as no anomaly and anomaly)



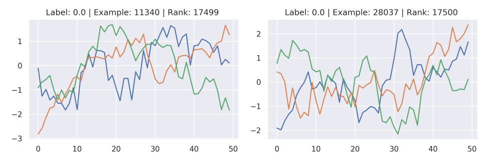
(b) Samples with the highest loss (both mislabeled as anomalies)



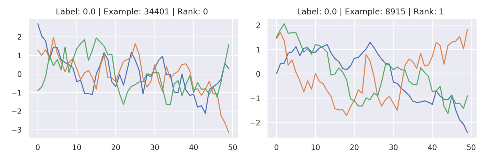
(c) Harmful samples with negative influence value (both mislabeled as no anomalies)



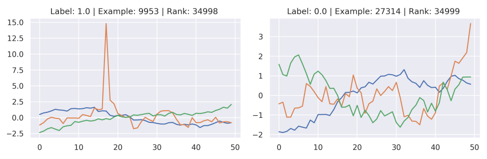
(d) Helpful samples with positive influence value (both labeled as anomalies)



(e) Samples with low absolute influence value, low impact (both labeled as no anomalies)



(f) Samples with the lowest absolute representer value (both labeled as no anomalies)



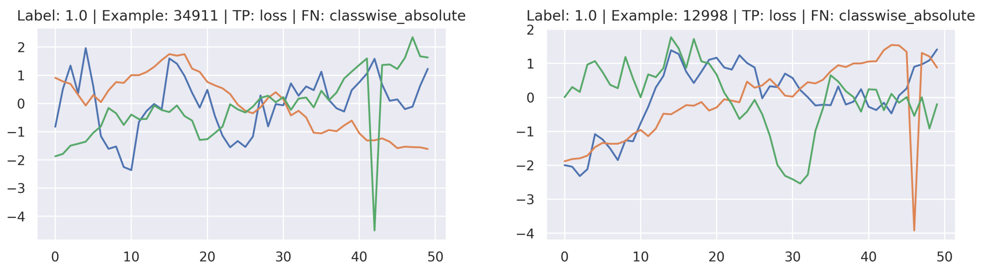
(g) Samples with the highest absolute representer value (labeled as anomaly and no anomaly)

Figure 9.7: Shows different selected samples and their scores based on the used approach.

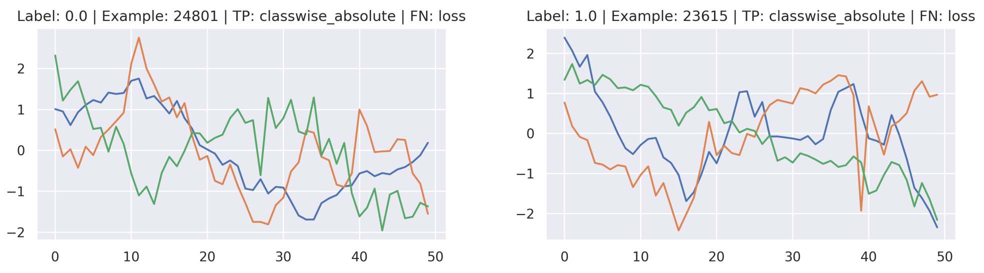
Both had the anomaly label and, as the figure shows, they should be classified as no anomaly samples. Therefore, their high loss showed that the model correctly learned the concept of anomaly detection.

Furthermore, Figure 9.7d shows the positive and Figure 9.7c the negative influencing samples. Figure 9.7e provides information about the least influencing samples. The negative influencing plots showed that the classifier worked correctly as both are mislabeled samples and the positive influencing and neutral ones were correctly labeled. Finally, Figure 9.7f shows the samples with a low representer value and Figure 9.7g the ones with high values. These samples did not include any mislabel. The combination of these insights again emphasizes that including the data and additional debugging methods, it is possible to not only detect the mislabeled samples but further show that the concept of the classifier was learned correctly.

As mentioned early on, the approaches detect different samples. Figure 9.8 shows some samples that were found either by the *Loss* or the *Influence*



(a) Mislabels found by loss



(b) Mislabels found by influence

Figure 9.8: Shows mislabeled samples that are only found either by the loss or influence approach.

Functions [66]. For example, the *Loss* provides be best mislabel correction rate if the model had a vague understanding of the problem, but it did not rank the samples according to their influence. Therefore, it could be that a significant lower mislabel correction accuracy results in superior classification accuracy. Contrary, the influence-based method provides information on how helpful and harmful the samples are but did not maximize the mislabel correction accuracy.

9.2.9 *Model Accuracy Comparison*

To complete the comparison of the methods, the change in the accuracy is represented for some representative experiments for the *Synthetic Anomaly Detection* dataset. In Figure 9.9 it is shown that the accuracy over ten runs for the 10% mislabeled dataset and the 20% mislabeled dataset is much better for some approaches and that the variance between the runs is minimal concerning the data quality.

Another aspect that is related to the previous analysis is the deletion of a subset based on the measurements. The suggested samples are deleted from the dataset instead of the manual correction, which needs time and additional effort. Therefore, the deletion of samples can be executed without human inspection and if the measurement is good, it should remove mislabeled data as well as other samples that harm the performance of the classifier. This results in a smaller dataset with improved data quality.

Figure 9.10 shows the performances for the mislabel correction compared to the deletion without inspection. In Figure 9.10a the deletion performed better for the 'classwise absolute' influence computation removing the most influencing

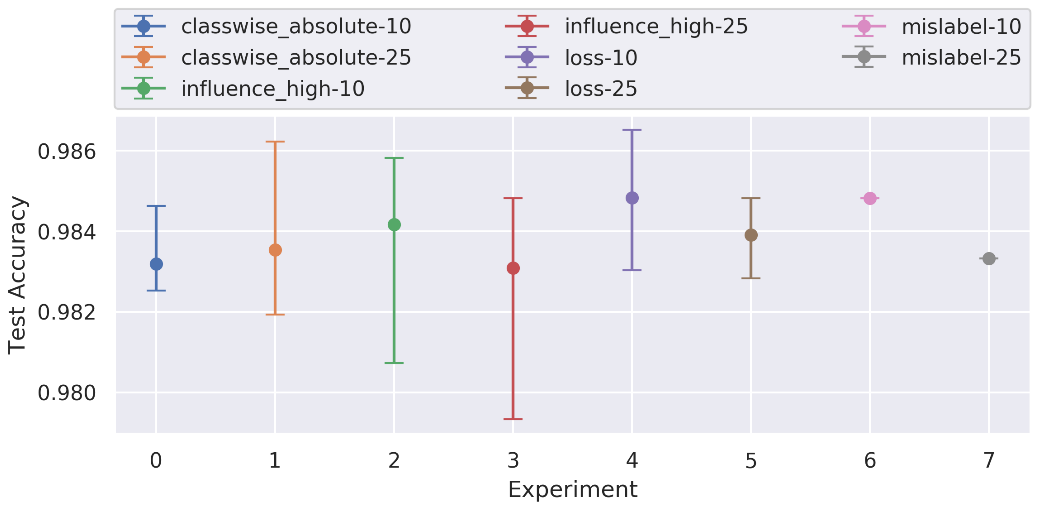
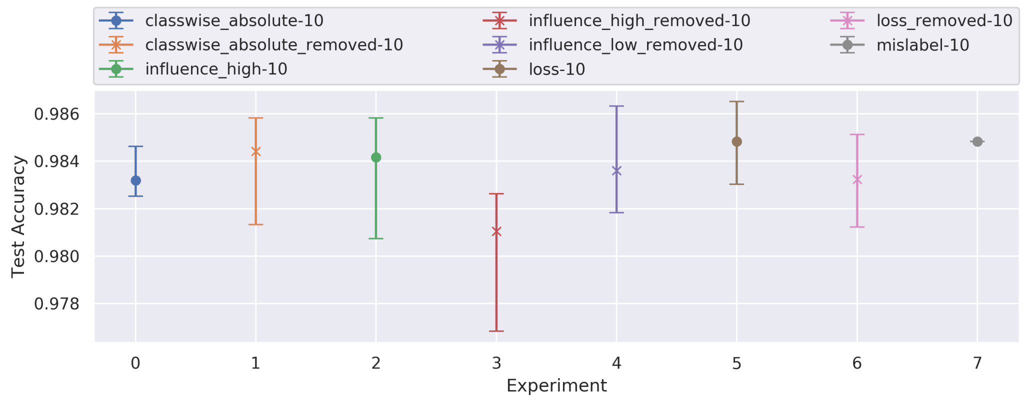
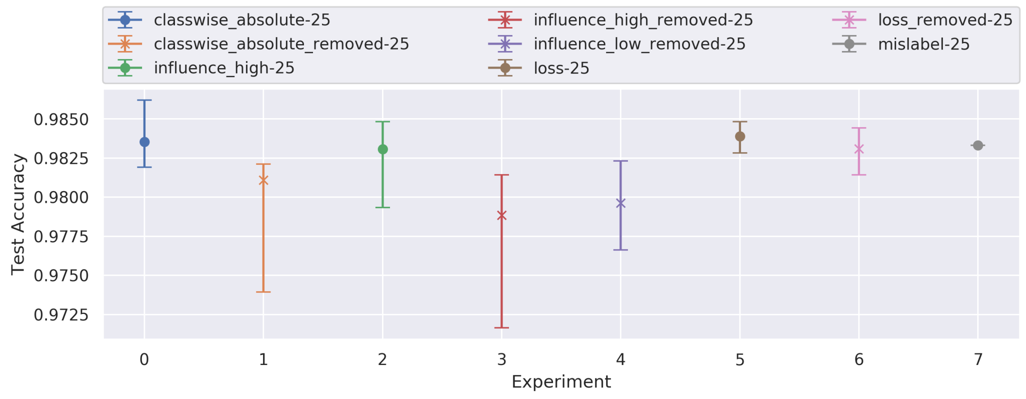


Figure 9.9: Accuracies of the different models for the *Synthetic Anomaly Detection* dataset (Quality 10% and 20% mislabeled) for the correction task.



(a) Accuracies for 10% deletion data.



(b) Accuracies for 25% deletion data.

Figure 9.10: Accuracies of the different models for anomaly dataset (Quality: 10% and 20% mislabeled) for the deletion task.

samples. Further, the scores for the influence computation [66] show that the deletion of samples with low scores improved the accuracy and the deletion of samples with high scores decreased the accuracy, reflecting the influence score concerning its definition of helpful and harmful samples. For the *Loss*, the accuracy dropped if the samples were deleted. This is especially the case because for the loss-based procedure the correction accuracy is excellent and the deletion of the samples just shrinks the data. The results have shown that except for the *Loss* the accuracies dropped compared to the mislabeled dataset. If a manual inspection is not a valid solution, the deletion of the samples based on the scores did not improve the quality of the data either.

9.3 DISCUSSION

When it comes to a stable, robust, and effective method to debug mislabels, the loss-based approach outperformed the other methods in accuracy and time consumption significantly. The only drawback is that there is no information about the influence of the detected samples, as this approach is not used for interpretability. The *Influence Functions* have shown to achieve nearly comparable results. Especially, when using the absolute values to check both the harmful and helpful samples, the correction rate was stable, providing additional influence information. The only drawback is the additional time, especially when the classwise evaluation is used. The *Representer Point* was outperformed by a large margin, making it not possible to compare it to the superior methods.

9.4 CONCLUSION

A comprehensive evaluation concerning the topic of automatic mislabel detection and correction was performed. Therefore, multiple experiments were examined and the performance of two well known existing methods in the domain of model interpretability was evaluated. In contrast to the expectations, the loss-based method handled the mislabel detection task better even though it is a direct measurement and the two already existing methods provide a much more profound understanding of the model. Also, it was shown that a combination of the methods can be more robust and lead to even better results. Furthermore, it has to be mentioned that the dataset debugging is only a subtask of the *Influence Functions* and *Representer Point*. Therefore, results were presented that help to interpret the model from a data-based perspective and used different measurements to provide an overview of the models' behavior. The most important samples for the model concerning the different approaches were identified. Finally, it was found that the deletion of the suggested mislabeled data did not work better than keeping the mislabeled data.

Part III

INTRINSIC INTERPRETABILITY

Intrinsic interpretability is less explored by the research community compared to the previous discussed post-hoc interpretability. However, that does not mean that this perspective is less important, as intrinsic interpretable networks provide a direct source of explanation. As they directly affect the reasoning process, they operate on a higher level of interpretability according to the GDPR. Similar to the post-hoc methods, most of the work for intrinsic approaches originates from the image domain and was designed to work well with the concepts and characteristics they provide. In this chapter, novel work related to intrinsic interpretability is presented.

First, an approach that divides the data and produces results on different levels is presented. The core idea of this approach is that dividing the problem into sub problems lowers the complexity and introduces some sort of explanation. E.g. information such as a small parts of the data belong to a class whereas other parts are irrelevant can be of high interest as it points out which parts are considered by the network.

The second part of this chapter introduces a novel prototype-based approach based on similarity. This approach is aligned with the time series characteristics and infers based on the similarity of data patches and representative prototypes. The prototypes are artificially learned during the training process and belong to small representative data pieces.

10 PATCHX: A NOVEL LEVEL-WISE CLASSIFICATION APPROACH

Cognitive load plays an important role when it comes to interpretability of data. It is known that the visual analysis of complete time series comes with an extensive cognitive overload, as it is difficult to perceive and leads to confusion. Therefore, dividing the problem into smaller subtasks can help to reduce the effort required to understand the explanation. This idea is used to come up with a novel approach, namely *PatchX*. *PatchX* performs a fine-grained patch classification using deep learning methods, followed by overall sample classification with a traditional machine learning classifier to understand the fine-grained patches and their interaction. The modular design allows exchanging the network and the classifier based on the needs. *PatchX* shows quantitatively and qualitatively superiority to its counterparts, with an increased interpretability.

10.1 METHOD

The proposed method is a hybrid approach using a neural network and a traditional machine learning approach to infer on the patch and the sample level. The architecture is modular and both the network and the classifier can be exchanged. The approach works with automatically created metadata for the patches extracted during the processing. Below, detailed information about the automatic label creation, required data transformation steps, the classification process, and the mathematical background is provided. The processing of a sample is divided into four steps shown in Figure 10.1.

10.1.1 *Data Transformation (Step 1)*

The initial processing step transforms the sample into patches used for further processing. To do so, a set of patch sizes and strides is passed to the framework that is used to define the boundaries. This set can include guesses, as the framework automatically handles the relevance of the provided sets. During the data transformation, every patch gets the label of its corresponding sample. An important aspect is that the transformation preserves the length of the data, which enables the use of different patch sizes within the same network. To do so, the data not related to the patch is set to zero, and an additional channel highlighting the true length of the patch is created.

$$x_i^p = \mathrm{transform}(x_i, p, s, l) \tag{10.45}$$

This chapter is an adapted version of the work presented in: D. Mercier, A. Dengel, and S. Ahmed. "PatchX: Explaining Deep Models by Intelligible Pattern Patches for Time-series Classification." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.

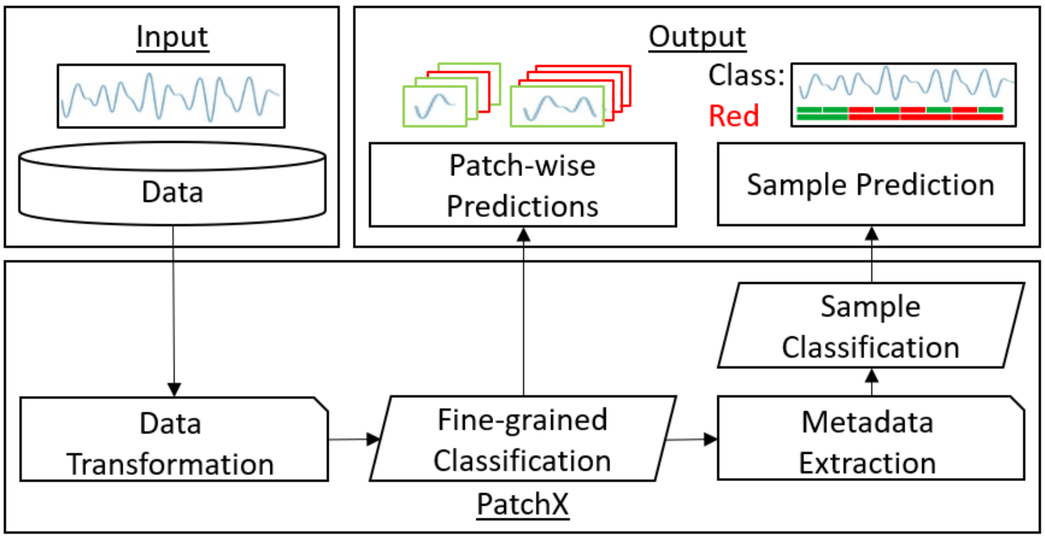


Figure 10.1: Shows the workflow of *PatchX*. *PatchX* only requires the time series to produce a fine-grained and an overall prediction.

$$\hat{X} = \{x_i^p \mid i \in N \land p \in \mathbb{N} \land p * s <\mid x_i \mid\} \tag{10.46}$$

Let x_i^p denote the transformed sample for the *p*-th patch of the *i*-th sample in *X*. Equation 10.45 shows the required parameters to compute x_i^p using the index *p* of the patch, *s* as the given stride, and *l* as the patch size. In Equation 10.46, the computation of $\hat{X}$ the set of all patches over each sample is shown. The transformed data $\hat{X}$ consisting of the different patches for each sample and the label y_i corresponding to the x_i is used to create x_i^p.

Due to simplicity reasons, it is assumed that only a single pair of *s* and *l* is used. However, the equations do not change dramatically as only the number of patches per sample increases, and a transformation for each setup is applied.

10.1.2 *Fine-grained Classification (Step 2)*

In the second step, the patch data is used to train a deep neural network to perform a fine-grained classification on the patch level. Although the dataset contains only the overall labels, it is possible to learn a patch-based behavior. The minimization of the network loss focuses on the samples that are class-specific. Therefore, the softmax prediction shows uncertainty for the patches that appear in different classes. This uncertainty highlights that the patches are not class-specific and shared between classes, serving as a confidence score.

$$H = \sum_{c=1}^{C} -y_{i,c} * \log(\Phi_c(x_i^p)) \tag{10.47}$$

$$L = \frac{1}{\mid N \mid * \mid P \mid} * \sum_i^N \sum_p^P H(x_i^p, y_i) \tag{10.48}$$

Equation 10.47 shows the cross-entropy for a single patch. In Equation 10.48, the loss over all samples is visualized. Therefore, P is denoted as the set of patches per sample and N as dataset size, corresponding to the size before splitting the data into patches. Furthermore, the patch classification network is denoted as Φ and C as the classes.

Discussion: Piece wise processed data is not available most time and its manual annotation is not suitable. Therefore, assigning the sample label to a patch introduces the question whether it is correct or not. In general, each patch can be classified as part of one of the following categories [62]:

- The pattern occurs only in a single class. In this case, the label is correct, and the classification of the patch will result in a very high value for the assigned class. This pattern is referred as a class-specific pattern.
- The pattern occurs in multiple classes. The network prediction for these samples will have medium values for some classes. These patterns are referred to as shared patterns.
- The pattern is not related to the label. These patches are actual mislabels, but due to the minority of this case, the loss minimization can handle them.

10.1.3 *Metadata Extraction (Step 3)*

The softmax prediction given by the fine-grained classification is then used to create a suitable representation for the sample classification. Therefore, a vector with the shape of the different classes is initialized. This vector is filled with the softmax values, using only the maximum value of each patch prediction and adding it to the corresponding class index. The vector is time-independent but represents the presence of different classes within the sample.

$$\tilde{x}_i^c = \sum_p^P \{\max(\Phi(x_i^p)) \mid \operatorname{argmax}(\Phi(x_i^p)) = c\} \tag{10.49}$$

$$\tilde{x}_i = \{\tilde{x}_i^c \mid c \in C\} \tag{10.50}$$

Equation 10.49 shows the computation of the entry $\tilde{x}_i^c$. $\tilde{x}_i$ is denoted as the feature vector used to compute x_i. The values of each $\tilde{x}_i^c$ are represented by the sum of the values corresponding to the class c of the patches for that c was predicted. Equation 10.50 shows $\tilde{x}_i$ as a set of $\tilde{x}_i^c$ computed for each class c in $\mid C \mid$ the set of classes.

Discussion: The drop of the temporal component during the metadata extraction can be explained with the previous patch prediction. The patch prediction is executed in a manner that takes care of the temporal component and inherently encodes the temporal component. This makes it possible to simply add the values to the corresponding class. Separating the class vector to preserve the patch sequence or the use of the complete metadata resulted in no information gain.

Table 10.1: The different datasets and their parameters.

Dataset	Train	Val	Test	Steps	Channels	Classes
Anomaly [91]	35,000	15,000	10,000	50	3	2
CharacterTrajectories	1,383	606	869	206	3	20
DailyAndSportActivities	22,344	9,576	13,680	60	45	19
ElectricDevices	6,244	2,682	7,711	96	1	7
FordA	2,520	1,081	1,320	500	1	2

10.1.4 *Sample Classification (Step 4)*

The last step is the classification based on the prepared metadata using a traditional ML algorithm. It was decided to use a support vector machine and random forest classifier as these had the best performance during the experiments. However, it is possible to replace them with any other classifier, e.g. a dense layer, nearest neighbor, or majority/occurrence voting.

$$y' = \{\Psi(\tilde{x}_i) \mid i \in N\} \tag{10.51}$$

In Equation 10.51 the final predictions are computed using the $\tilde{x}$ as a feature vector created for each x_i of the dataset.

10.2 DATASETS

A set of five publicly available time series datasets was used to show that the proposed approach works without a restriction to a specific dataset. Precisely, four datasets from the UEA & UCR Time Series Classification Repository [9], and a point anomaly dataset proposed in [91] were used. These datasets were selected to emphasize broad applicability and possible limitations. In Table 10.1 the different parameters of each dataset are shown.

10.3 EXPERIMENTS & RESULTS

A high-quality interpretable network architecture requires not only good accuracy, but further requires a suitable explanation and broad applicability. Therefore, several experiments were performed including a comparison of non-interpretable, and interpretable approaches, and different accuracy evaluations.

A convolutional network was used consisting of four conv-pool blocks and a fully connected layer on top of the four blocks. Each conv-pool block contains a 1d convolutional layer using Rectified Linear Unit (ReLU) activation and a 1d max-pool layer. On top of the last max-pool layer, a dense layer with ReLU activation and the final classification layer using a softmax activation were used. It was decided to test the approach using a Convolutional Neural Network (CNN) structure, but other architectures such as Recurrent Neural Networks (RNNs) are possible too. During training, a maximum of 50 epochs was set with a batch size of 32 and a learning rate schedule and early stopping callback.

10.3.1 *Accuracy Comparison*

A comprehensive accuracy evaluation over different classification approaches was performed to compare their performances and scaleability. Besides, the approaches were divided into non-interpretable deep learning, and interpretable methods. Furthermore, within the class of interpretable approaches, traditional machine learning algorithms and the proposed interpretable hybrid approach were differentiated. Finally, a feature extraction preprocessing step was included for some methods to evaluate the impact of extracted features against raw data usage. The different approaches and performances are shown in Table 10.2.

Although deep learning is known to perform well, the results of traditional machine learning including a feature extraction step have shown superior performance. However, this feature extraction approach limits the approaches significantly. E.g. for the *DailyAndSportActivities* dataset, the time consumption and memory usage scale poor, making it impossible to use the feature extraction for larger datasets. Besides, the feature extraction assumes that the numerical features of interest are already known and does not learn others directly from the data.

Excluding the limited feature extraction approaches, deep learning outperformed the traditional approaches in almost every task. Compared to the traditional approaches, the deep learning algorithms did not suffer from the increased dataset size, and CNNs have shown to be able to work with the raw data and produce high accuracy results. However, the results are not interpretable without additional efforts.

The results show that *PatchX* is a good compromise as its accuracy was only slightly lower compared to the deep learning approach, but it scales very well and produces explainable results. Furthermore, the computation time of the hybrid approach was only slightly higher than the computation of the black-box approach.

Table 10.2: Shows the accuracy comparison. *Feature* approaches include a feature extraction pre-processing. The *Trivial* approach covers a majority/occurrence voting after the fine-grained classification. Numbers are given in percentage. Bold numbers are the best scalable approach excluding feature-based due to their limitations concerning the required previous knowledge and feature selection.

	Scalable				**Not scalable**			
Dataset	**Black-box**	**Interpretable**						
		PatchX			**Traditional approaches**			
	CNN	**CNN + SVM**	**CNN + RF**	**CNN + Trivial**	**SVM**	**RF**	**SVM feature**	**RF feature**
Anomaly	**98.70**	98.41	97.25	98.23	97.74	96.34	99.08	99.99
CharacterTrajectories	96.55	95.17	94.13	82.74	**98.62**	98.16	92.98	98.16
DailyAndSportActivites	99.74	99.82	**99.88**	99.79	98.54	99.60	-	-
ElectricDevices	67.31	**69.29**	68.56	60.68	60.69	65.21	24.23	69.46
FordA	88.71	**90.08**	82.42	88.48	83.33	74.92	92.88	100.00

Table 10.3: Shows the time consumption results. *T* denotes the training time in seconds. *I* denotes the inference time. Used hardware: Intel Xeon (Quad Core), Nvidia GTX 1080 Ti, 64 GB memory.

Dataset	**Mode**	**PatchX**	**Black-box**	**SVM**	**SVM Feature**
Anomaly	T	168.5	45.3	1,290.0	3,006.5
	I	2.6	0.6	33.7	260.3
CharacterTrajectories	T	69.9	6.7	0.8	132.2
	I	1.4	0.2	0.5	45.2
DailyAndSportActivities	T	220.9	32.1	511.8	-
	I	5.8	1.2	504.5	-
ElectricDevices	T	83.2	7.4	9.3	161.9
	I	4.8	0.6	5.8	12.1
FordA	T	159.2	10.5	5.9	293.0
	I	3.0	0.2	2.0	100.8

10.3.2 *Computation Time Analysis*

Below, the experiments to evaluate the time consumption of the different approaches are discussed. In Figure 10.3 the results are shown for the complete training procedure and the testing using the test datasets. The results have indicated that the approaches that use the feature extraction scale pretty bad. Conversely, the deep learning-based methods scale pretty well. Furthermore, *PatchX* was slower than the traditional approach when the dataset was small, but scales significantly better when the dataset size increased.

$$t = \sum_{d \in D} \left(\sum_{j=0}^{S} \left(\left\lceil \frac{l_{sample}}{s_j} \right\rceil \right) * t_{net} + t_{head} \right) \tag{10.52}$$

Equation 10.52 shows the time it takes to train or perform predictions using the proposed approach. First, denote t_{net} and t_{head} as the time it takes to process an input using the network or the head, respectively. The head corresponds to the sample classifier, whereas the net to the network architecture used for the patch classification. *D* denotes the dataset, l_{sample} the number of time steps a sample has, and *S* denotes the set strides. In a traditional deep neural network, the inner sum is equal to one. However, the computation for the t_{head} is not directly comparable due to the different number of features. Usually, this number depends on the network architecture and the input space, however, for *PatchX* it depends on the number of patches per sample and classes.

10.3.3 *Hyperparameter Selection*

To produce interpretable results, a careful hyperparameter selection is important. Therefore, *PatchX* requires two types of parameters.

Table 10.4: Shows the performance comparison using different parameter sets. 'S' denotes the stride between the patches, and 'L' denotes the length of each patch. Numbers are given in accuracy percentage.

Dataset	**PatchX**		
	S 5 L 10	**S 10 L 20**	**S 5,10 L 10,20**
Anomaly	98.27	98.14	**98.29**
CharacterTrajectories	**94.82**	95.17	94.36
DailyAndSportActivites	99.66	99.74	**99.82**
ElectricDevices	65.78	69.06	**69.29**
FordA	51.59	**89.17**	87.20

10.3.3.1 *Patch Creation Parameters*

The first category includes the parameters that directly influence the patches. Precisely, *stride* and *length* are used to define the patches. As in any other use case, the *stride* defines the gap between the patches of the same sample, and the *length* the length of the data. Using *PatchX* it is possible to use multiple *strides* and *lengths* resulting in a larger dataset and different levels of explanation.

In Table 10.4 the impact of the different setups is shown. Therefore, the stride is defined to be half of the length to produce an overlap of the samples and used an SVM classifier for the sample classification. The results show that for all datasets except the *FordA* the small patch length was able to capture the pattern required to classify the samples. However, using the increased patch size, the approach was able to recover the network for the *FordA* dataset.

The results highlight how crucial the parameter selection is for interpretation and accuracy. Intuitively, smaller patch sizes are related to basic patterns and larger patch sizes cover complex patterns [37, 75]. Furthermore, the experiment has shown that the use of multiple parameter setups including the not working setup for the *FordA* data resulted in high accuracy values as the approach automatically takes care of setups that are irrelevant for the sample classification.

10.3.3.2 *Patch transformation parameters*

Considering the variable length of the patches within the experiment settings requires an advanced transformation. Three parameters are required to handle the transformation. Although it is possible to have different combinations of these parameters, only a limited set is valid:

1. *Zero* is used to set the data not included in the patch to zero. As the model has a fixed input size, it is mandatory to maintain the sample size for all patch sizes.

2. *Attach* indicates the data related to the patch using an additional channel.

3. *Notemp* removes the time component and shifts the patch to the beginning of the sample. This requires the use of *Zero* to indicate the end of the patch.

Table 10.5: Evaluation of data transformation. Patch classification on the *Anomaly* dataset. The first row shows an invalid run, as this setup performed a sample classification. Numbers are given in percentage.

Setup			Train Acc.	Val Acc.	Test Acc.
zero	**attach**	**notemp**			
~~False~~	~~True~~	~~False~~	~~100.0~~	~~98.54~~	~~98.41~~
True	False	False	98.50	98.29	98.01
True	True	False	98.59	98.57	98.27
True	False	True	98.30	98.04	98.04
True	True	True	98.61	98.40	98.27

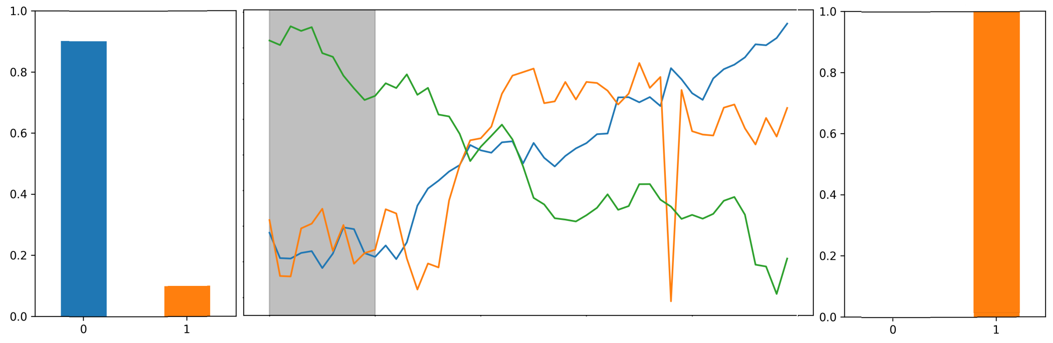


Figure 10.2: Classification of a sample. Blue: No anomaly. Orange: anomaly. Middle: Highlighted no anomaly part is classified. Left: Classification using *Zero*. Right: Classification not using *Zero*.

Table 10.5 shows different possible combinations. Surprisingly, the network was unable to exclude the data outside the patches using the *Attach* transformation as shown in Figure 10.2 on the right side. The non-anomaly patch was classified as an anomaly and the saliency map has shown that the network took the peak into account, which means the network performed a sample classification instead of a patch classification. Using this information, it can be concluded that the attachment of the channel is not enough to restrict the network. Therefore, setting the data not included in the patch to zero is mandatory to force a patch classification. In Figure 10.2 the result for the same patch using the *Zero* flag is shown on the left side. This time, the model performed a patch classification. Using the other parameters can be of relevance when there is information available about the dataset. However, the performance increase for the *Anomaly* dataset was limited due to the time independence.

10.3.4 *Local & Global Patch-based Explanations*

Using the patch classifier, it is possible to produce predictions for each patch. Based on the stride and length of each patch, this results in different overlapping explanations. As shown in Figure 10.2 it is possible to get detailed information about a patch and its classification. This patch explanation covers only a small

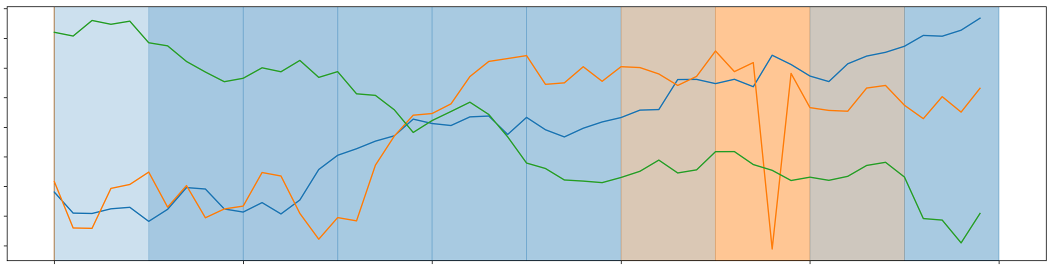


Figure 10.3: Shows the explanation overlay. Gradient highlights confidence. Blue: No anomaly. Orange: anomaly.

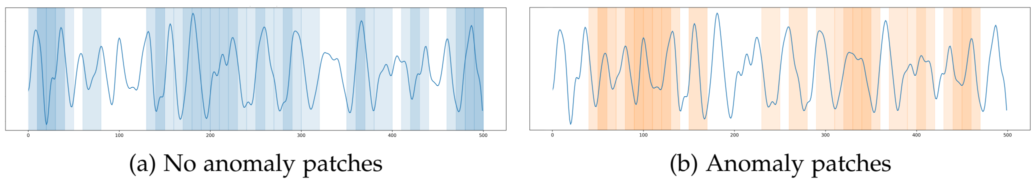


Figure 10.4: FordA sequence explanation. Class wise patch prediction overlay using PatchX. Blue: No anomaly. Orange: anomaly.

piece of data and is much easier to understand due to the lower cognitive load [104].

The combination of the patches results in a global sample explanation with fine-grained classification scores for every patch. The color gradient visualizes the confidence of the classifier. Lower gradients relate to patches class independent, whereas higher values relate to class dependent patches.

Figure 10.3 shows the combination of the patch classifications. The blue color is assigned to non-anomaly patches and orange to anomaly patches. Ultimately, it highlights that the patch length for this sample could be even smaller. Precisely, some data classified as non-anomaly by neighborhood patches is included in the anomaly patches. Using a setup with multiple patch sizes results in more precise localization.

Additionally, the overlay approach can provide visualizations restricted to a specific set of classes, e.g. an explanation of the *FordA* dataset consists of two classes. This dataset covers an anomaly detection task. In contrast to the previous results using the *Anomaly* dataset, the anomalies are not restricted to a single location. However, it is possible to highlight the class-specific regions, as shown in Figure 10.4. Especially, when the data covers numerous time steps and data unrelated to the class of interest, this visualization enables easy filtering. Precisely, the explanation shown in Figure 10.4 highlights small parts there were classified independently as an anomaly or no anomaly, enabling a piecewise inspection of these pieces.

10.3.5 *Global Patch Confidence*

As the local patch classifier predicts a label for each patch, it is important to differ between the following cases. Patches that have high value are likely to be class-

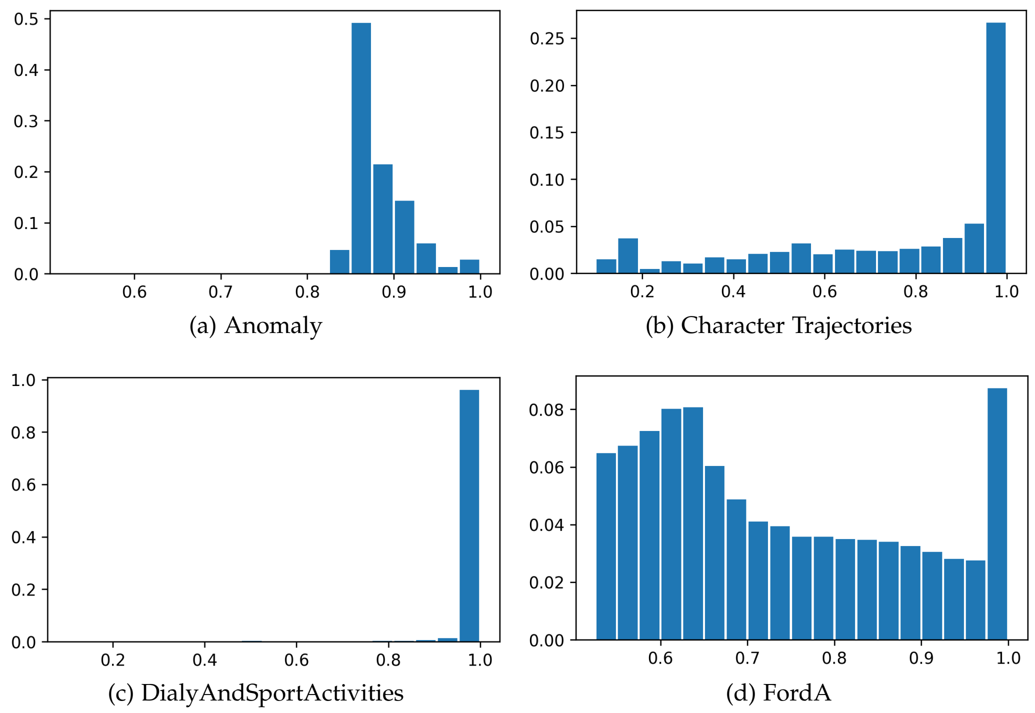


Figure 10.5: Shows the confidence for the patches. Y-axis: number of patches. X-axis: Softmax.

specific, low valued patches are likely to be class unrelated and medium values correspond to patches shared between classes.

Using the scores for each patch, it is possible to get global insights about the task. In Figure 10.5 the patch confidence over the dataset is visualized as a histogram. The confidence of each patch is calculated using the value of their softmax prediction to highlight their uniqueness. The patterns of the *DailyAndSportActivities* are unique to each class, highlighted by the high values. The same holds for the *Character Trajectories* dataset. However, some of the patterns are shared across the classes. Intuitively, when drawing a character, some parts of the drawing are not unique to a character. E.g. the classes 'e' and 'o' only differ in the first and last part of the signal. In contrast to the previously mentioned datasets, the *Anomaly* dataset shows different behavior as it covers only point anomalies the number of patches that appear exclusively in one class is smaller. The *FordA* dataset shows a combination of the previously mentioned behaviors as it contains class-specific, unrelated, and shared patterns.

10.3.6 *Class Boundary Evaluation*

The patch prediction and confidence scores can be used to explore class boundaries. It is shown that they help to identify the source of a class shift. In Figure 10.6a peak is visualized. The initial series (blue line) without any change has a non-anomaly label. When increasing the peak value, the label switches after the third change to an anomaly (orange line). Mathematically, the labels of this

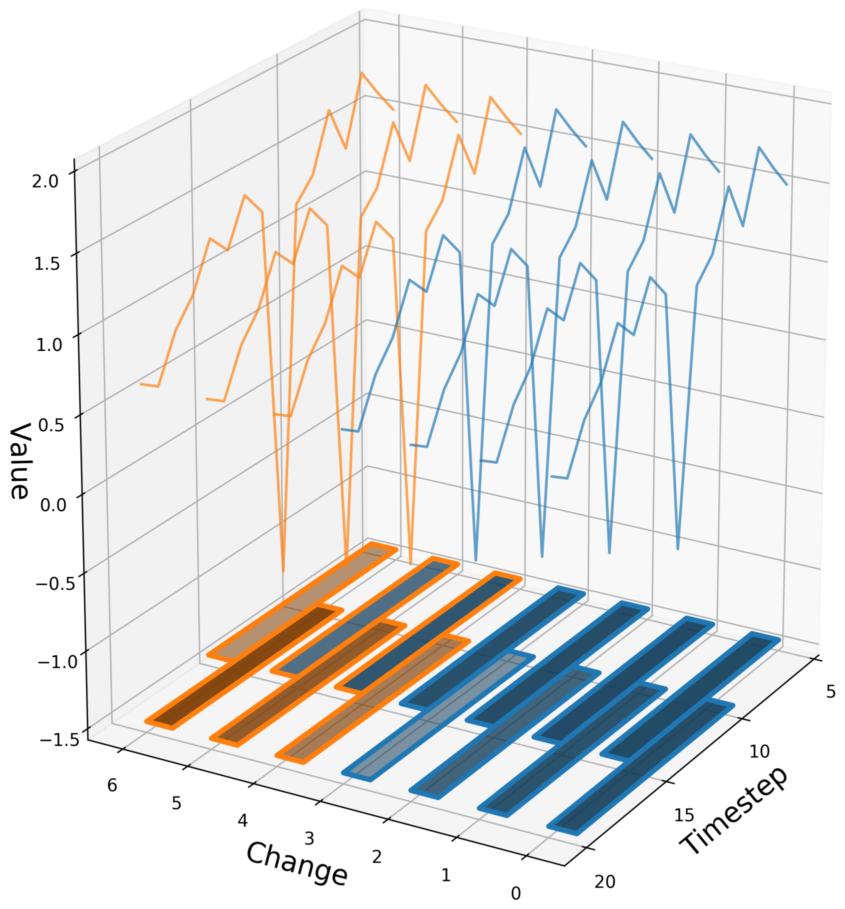
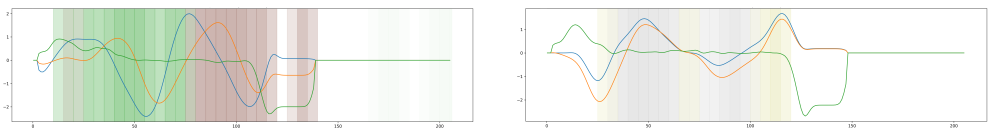


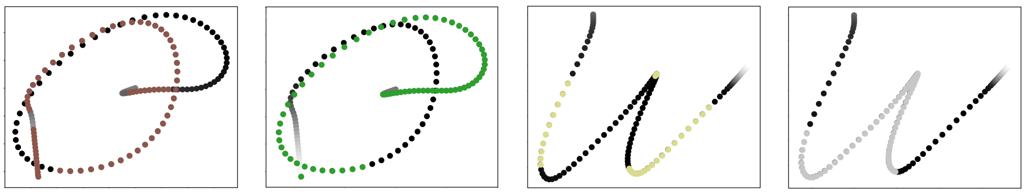
Figure 10.6: Shows how to convert the sample from one to another class. Line color: ground-truth. Bars border color: prediction. Bar color and the gradient: Patch prediction.

dataset are computed based on the mean and std within the signal. Therefore, it is possible to create a series of the same signal while slightly increasing the peak to shift the class. However, to understand if the classifier learned the class boundary correctly, a look at the change of the patch and overall prediction is required. The border color of the bars shows the sample classification, whereas the bar color shows the patch prediction. Two patches were used for this sample. The results provide evidence that the boundary is learned correctly, as the border color of the bars and the line color match. However, the patch covering time steps 10 to 20 changes first, whereas the prediction of the patch that only includes a part of the peak changes much slower.

Also, samples sometimes are mislabeled because they are close to a class boundary. In Figure 10.7 the results of two misclassified characters are shown. Thanks to the patchwise prediction, it is possible to directly understand the classification. The time series is projected back to the two-dimensional drawing of the character and used the overlay to highlight the predicted classes for the patches.

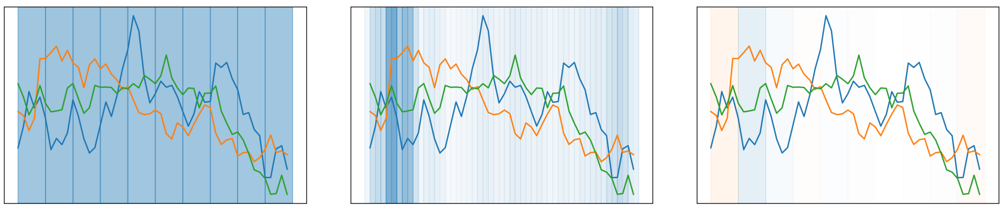


(a) Label: 'o'. Prediction: 'e' (b) Label: 'w'. Prediction: 'u'

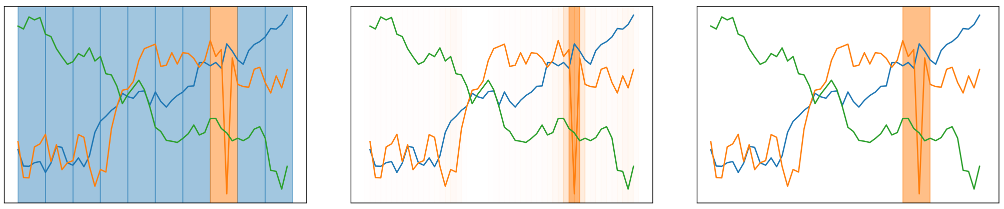


(c) Pred. as: 'o'. (d) Pred. as 'e'. (e) Pred. as 'w'. (f) Pred. as 'u'.

Figure 10.7: Shows the explanation for mislabeled data thanks to the patch classification. First row: Overlay of two mislabeled characters. Second row: Correct classified patches. Third row: Misclassified patches.



(a) PatchX (b) SHAP (c) LIME



(d) PatchX (e) SHAP (f) LIME

Figure 10.8: Comparison of PatchX to state-of-the-art approaches. First row no anomaly, and second row anomaly sample. Orange: Anomaly. Blue: No anomaly. White: not classified.

10.3.7 *Comparison with State-of-the-art Approaches*

This section provides a comparison of the proposed method with two well known state-of-the-art approaches, namely *LIME* [115] and *SHAP* [80]. As both methods use the black-box model to perform a model-agnostic explanation, the accuracy can be found in Table 10.2 using the black-box model. Figure 10.8 shows two samples of the anomaly dataset. Starting with the second row, all methods were able to precisely locate the peak within the signal. However, *SHAP* and *LIME* only provide information about the part relevant to the prediction. In contrast, *PatchX* provides information for every patch. The first row shows a sample with a significant peak. However, the explanation of *SHAP* and *LIME* is not as intuitive as the one of *PatchX*.

It has to be mentioned that *SHAP* provides additional information about the relevant channel, and it is possible to combine the approaches. *SHAP* can be used on the proposed fine-grained patch prediction model to provide additional insights about the neural network to enhance the explanation further.

10.4 CONCLUSION

This section has shown that the proposed hybrid approach can produce interpretable results for different time series classification tasks. The utilization of neural networks and traditional machine learning approaches provides local and global instance-based explanations. The approach improves the interpretation of mislabels, class boundaries, and sample explanations. Furthermore, the hybrid approach covers different levels of explanation, focusing on both low and high-level patterns. Finally, the results emphasize that the hybrid approach builds a bridge between the interpretable traditional machine learning algorithms and neural networks. It combines the scalability, performance, and interpretability advantages of both worlds.

11 P2EXNET: A NOVEL PATCH-BASED PROTOTYPE NETWORK ARCHITECTURE

Divide and conquer is a concept that is applicable in many situations and able to reduce the complexity of many problems. The previous section has shown a divide and conquer approach. Think one step further, it is possible to create not only patches but also prototypes for those. The novel interpretable network scheme proposed in this section is designed to inherently use an explicable reasoning process inspired by the human cognition without the need of additional post-hoc explainability methods and creates prototypes for patches of the input data. It is based on the standard deep neural network architecture, providing a global explanation using representative class-specific prototypes and an instance-based local explanation using patch-based similarities and class similarities. An analysis of the results on publicly available time series datasets reveals that *P2ExNet* reaches similar performance when compared to its counterparts while inherently providing understandable and traceable decisions.

11.1 METHOD

Inspired by human reasoning behavior, *P2ExNet* is aligned to rely on implicit knowledge about objects and examples already seen before. Precisely, new instances are compared to abstract concepts to include class-specific features. This is called prototypical knowledge and describes the knowledge about these concepts and covers the analogical process to map new to the existing knowledge [46]. Following this process, the proposed method uses shallow representations. The prototypes encode class-specific pattern and provide the decision based on similarity.

11.1.1 *Architecture*

Inspired by the work of Gee et al. [45], an autoencoder is combined with a prototype network. The autoencoder consists of several convolutional and max-pooling layers serving as a feature encoding network to provide a latent representation that encodes the relevant features of an input sequence. This representation is fed forward to a custom prototype layer to generate prototypes. Motivated by the work of Chen et al. [17], multiple prototypes are used to represent a sample rather than a single one for the complete input. Precisely, the prototype layer has randomly initialized variables representing patch prototypes of user-defined size. Larger sizes will result in composed concepts, and smaller

This chapter is an adapted version of the work presented in: D. Mercier, A. Dengel, and S. Ahmed. “P2exnet: Patch-based prototype explanation network.” In: *International Conference on Neural Information Processing*. Springer. 2020, pp. 318–330.

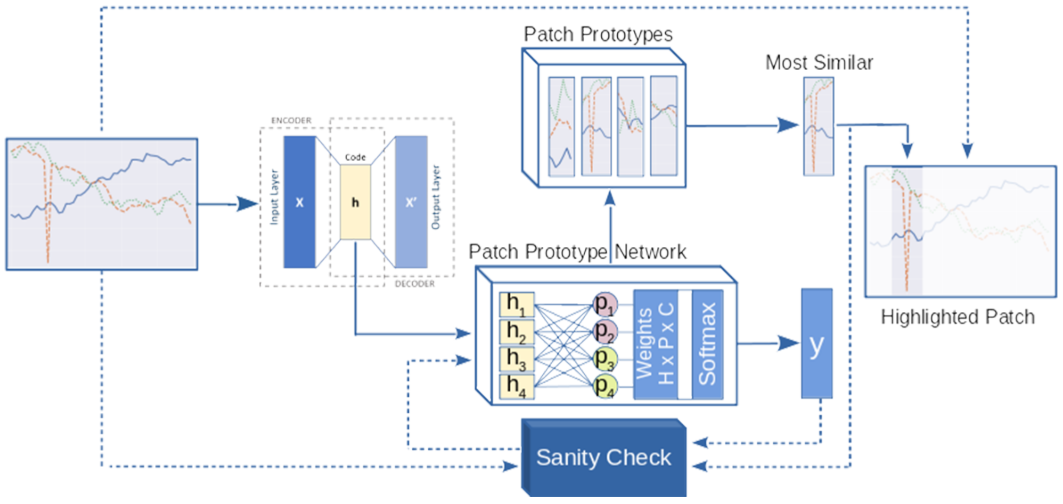


Figure 11.1: Inference and testing workflow. Artificially, computed prototypes are evaluated in a similarity-based manner to suggest class-specific patches.

sizes result in more basic concepts. On top of the prototype layer, a prototype weight layer is attached to encourage class-specific prototypes and weight their position within the sample to cover the local importance. Finally, a softmax classification evaluates the similarity scores produced by the prototype layer multiplied with weights of the prototypes, as shown in Figure 11.1.

11.1.2 *Mathematical Background*

P2ExNet uses a novel combined loss that captures several aspects, enabling the network to produce a meaningful set of patch prototypes based on the losses proposed by [17, 45]. For the following equations, let S_x be the set of patches corresponding to a sample x and the set P of prototypes.

11.1.2.1 *Distances*

The L^2 norm is used to compute the distance between any two vectors. Furthermore, the minimum distance between a sample and any prototype (D_{s2p}) and vice versa (D_{p2s}) is computed. D_{p2p} is denoted as the minimal distance between a prototype and all others and calculate the minimum distance to a prototype of the same class D_{clst} and to the other classes D_{sep} w.r.t. y. Therefore, P_y denotes the subset of P assigned to the class label of y. The distances are shown in Equations 11.53 to 11.56.

$$D_{s2p}(s) = \min_{p \in P} L^2(s, p) \tag{11.53}$$

$$D_{p2p}(p) = \min_{p' \in P} L^2(p, p') \tag{11.54}$$

$$D_{clst}(s, y) = \min_{p \in P_y} L^2(s, p) \tag{11.55}$$

$$D_{sep}(s, y) = \min_{p \in \{P \setminus P_y\}} D_{s2p}(s, p) \tag{11.56}$$

11.1.2.2 *Loses*

To ensure high-quality prototypes, a novel patch loss is introduced. This loss is a combination of different objectives to achieve good accuracy and an explanation that does not contain duplicates or prototypes that are not class-specific. This loss combines the following losses:

- Autoencoder loss: Mean Squared Error (MSE) is used to encourage reconstruction, later used for prototype reconstruction.
- Classification loss: To produce logits for the softmax cross-entropy, the reciprocal of D_{s2p} and the prototype-weight layer are multiplied.
- L_{p2s} and L_{s2p}: These losses preserve the relation between the input and the prototypes and vice versa as shown in Equation 11.57 and 11.58.
- L_{div}: The diversity among the patch prototypes is computed as shown in Equation 11.59.
- L_{clst} and L_{sep}: To encourage the network to learn class-specific prototypes, L_{clst} and similarly to L_{sep} but with a negative sign are computed. This penalized prototypes close to samples of the wrong class w.r.t. their assigned class.

$$L_{p2s}(x) = \frac{1}{|P|} \sum_{p \in P} D_{p2s}(p) \tag{11.57}$$

$$L_{s2p}(x) = \frac{1}{|S_x|} \sum_{s \in S_x} D_{s2p}(s) \tag{11.58}$$

$$L_{div} = \log(1 + \frac{1}{|P|} \sum_{p \in P} D_{p2p}(p))^{-1} \tag{11.59}$$

$$L_{clst}(x, y) = \frac{1}{|S_x|} \sum_{s \in S_x} D_{clst}(s, y) \tag{11.60}$$

The proposed final loss is a linear combination considering previously mentioned aspects and ensures meaningful, diverse, and class-specific patch prototypes shown in Equation 11.61. By default, all lambda values except λ_c are set to one to find the best compromise between the objectives, preserving high accuracy.

$$\begin{aligned} \text{Patch\_Loss}(x, y) = {} & \lambda_c H(x, y) + \lambda_{mse} MSE(x, x) + \lambda_{p2s} L_{p2s}(x) \\ & + \lambda_{s2p} L_{s2p}(x) + \lambda_{div} L_{div} + \lambda_{clst} L_{clst}(x, y) + \lambda_{sep} L_{sep}(x, y) \end{aligned} \tag{11.61}$$

Table 11.1: Datasets used in to benchmark P2ExNet.

Dataset	Train	Val	Test	Steps	Channels	Classes
50words	183	83	286	270	1	13
Adiac	261	129	391	176	1	37
Anomaly Detection [91]	35,000	15,000	10,000	50	3	2
Character Trajectories	1,383	606	869	206	3	20
Crop	5,040	2,160	16,800	46	1	24
Electric Devices	6,244	2,682	7,711	96	1	7
FordA	2,520	1,081	1,320	500	1	2
Pen Digits	5,242	2,252	3,498	8	2	10

11.1.3 *Training Process*

The training process of *P2ExNet* consists of two stages. In the first stage, the weights of the pre-initialized prototype weight layer are fixed to ensure class-specific prototypes. Then the network is trained until it converges. In the second learning phase, all layers except the prototype weight layer are frozen, and the network learns to adjust the prototype weights. The adjustment corrects the prototype class affiliation using the previously trained latent representation.

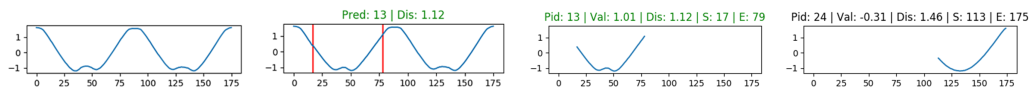
11.2 DATASETS

Eight publicly available time series datasets were used to emphasize the broad applicability of the proposed approach and examine possible limitations. As a representative set, seven different datasets from the UEA & UCR Time Series Classification Repository [9] and a point anomaly dataset proposed in [91] were used. These datasets and their parameters are visualized in Table 11.1. To have better coverage of different types, the datasets were selected based on the characteristics concerning the number of classes, channels, and time steps to cover several conditions and show the prototypes.

11.3 EXPERIMENTS & RESULTS

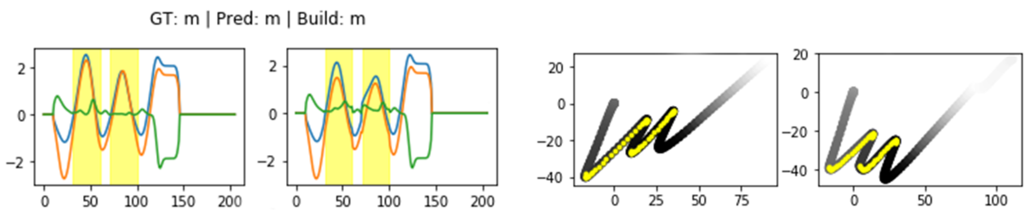
In this subsection, the experiments and results concerning the performance, applicability, and resource consumption for the proposed approach are presented.

The proposed method provides the possibility to identify and highlight the parts of the input that were most relevant for the classification. Besides, it provides prototypes along with a sample containing the prototypes to compare it to the original input. Figure 11.2 shows highlighted regions that were important for the inference on the *Adiac* dataset sample. This explanation includes the original sample of the *Adiac* dataset, a modified version, and two prototypes. In the modified version shown in Figure 11.2b, the part between the two red lines was replaced with the most important patch prototype to show how close it is to the



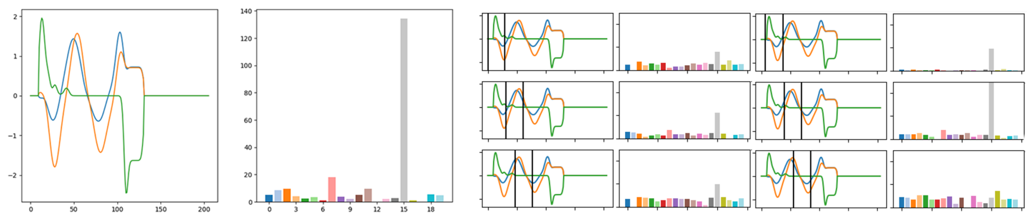
(a) Original (b) Modified (c) Prototypes

Figure 11.2: *Adiac* dataset prototype explanation. a) shows the original series. b) shows the series with the prototype between the red bars. c) shows two prototypes.



(a) Time series (b) Character of the class 'm'

Figure 11.3: *Character Trajectories* dataset prototype explanation. a) shows the original series and the series with the prototypes. b) shows the character output and the modified character.



(a) Overall distribution (b) Patch distribution

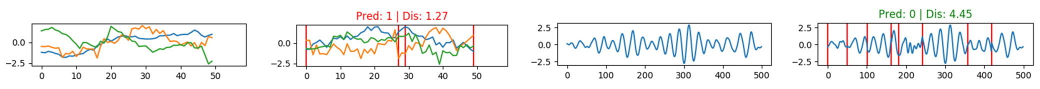
Figure 11.4: Class and prototype distribution for character dataset. a) shows the class similarities. b) shows some patches and the corresponding class similarities.

original part. Figure 11.2c shows two prototypes. The value of each prototype denoted as 'Val' highlights its contribution towards the classification result.

11.3.1 *P2ExNet: Instance-based Evaluation*

Similarly, Figure 11.3 shows a sample from the *Character Trajectories* dataset and the mapping of the time series back to the character. The black value highlights the pressure of the pen, and the yellow part shows the mapping of the prototype back to the input space.

Furthermore, in Figure 11.4 the classwise overall and patchwise distribution provides additional information about similar classes and important patch positions. Especially in Figure 11.4b, it is shown that not all patches have the same importance when it comes to the classification. There are sensitive datasets for which the re-classification can change if the original data gets replaced with a prototype. However, for the classification and the explanation, this is not a problem as it can be solved. A proper scaling and adjustment can remove the offset between the prototype and the time series.



(a) Original (b) Modified (c) Original (d) Modified

Figure 11.5: Prototype substitution. a) and c) show original time series. b) and d) show the corresponding modified samples and their re-classification.

Table 11.2: A comparison of interpretable and the corresponding non-interpretable counterpart.

Dataset	**CNN Acc. [%]**	**P2ExNet Acc. [%]**
50words	76.84	**81.98**
Adiac	**63.54**	60.15
Anomaly Detection	**99.79**	93.79
Character Trajectories	**96.53**	91.78
Crop	68.27	**68.54**
Devices	55.42	**62.53**
FordA	85.44	**89.32**
Pen Digits	**94.29**	93.95

In Figure 11.5b such a jump in the orange signal is shown and leads to an anomaly highlighted by the red caption. However, the classification of the original signal with the network was correct. Furthermore, some datasets are invariant to small offsets, shown in Figure 11.5d. That is why scaling should be done based on the problem task. In the case of a point anomaly task, the patches have to align.

11.3.2 *P2ExNet: Evaluation as a Classifier*

Usually, intrinsic interpretability approaches come with an accuracy drop. In Table 11.2 the accuracy trade-off is presented, highlighting that *P2ExNet* is on the same level as the black-box counterpart. To create a network similar to the proposed architecture without the interpretable part, the prototype layer with a dense layer and a cross-entropy loss as suggested by Chen et al. [17] was used. Furthermore, the decoder was removed as there is no need to restrict the latent representation as no reconstruction is required. This comparison was conducted for all datasets, showing that *P2ExNet* achieves comparable or better performance in comparison to the non-interpretable variant. Overall, the interpretable network has an insignificant performance increase of 0.03%. Each architecture was superior in four out of the eight datasets. The results prove that the accuracy using the interpretable model dropped about 6% on the *Anomaly Detection* dataset but increased 7% on the *Electric Devices* dataset.

Table 11.3: Replacement of original patch. Second column: Percentage replaced by prototypes. Third column: Prediction agreement. Fourth column: *P2ExNet* accuracy on original samples. Fifth column: *P2ExNet* accuracy on modified samples. The first row of each dataset corresponds to replacements with the most similar, whereas the second row with the most different prototype. Numbers are given in percentage.

Dataset	Replaced	Equal Pred.	Acc.	Modified Acc.
50words	36.43	93.01	81.98	**77.20**
	52.88	62.50		56.98
Adiac	35.22	85.97	60.15	**55.98**
	69.90	9.11		14.84
Anomaly Detection	71.99	87.43	93.79	**91.78**
	67.32	19.45		22.72
Character Trajectories	18.15	92.93	91.78	**85.30**
	52.90	31.71		32.87
Crop	50.50	94.08	68.54	**66.94**
	81.12	22.01		23.28
Electric Devices	52.36	81.65	62.53	**60.52**
	65.81	49.81		39.11
FordA	51.17	99.92	89.32	**89.40**
	44.95	23.09		32.69
Pen Digits	69.47	99.31	93.95	**93.54**
	68.65	8.83		11.0

11.3.3 *P2ExNet: Sanity Check*

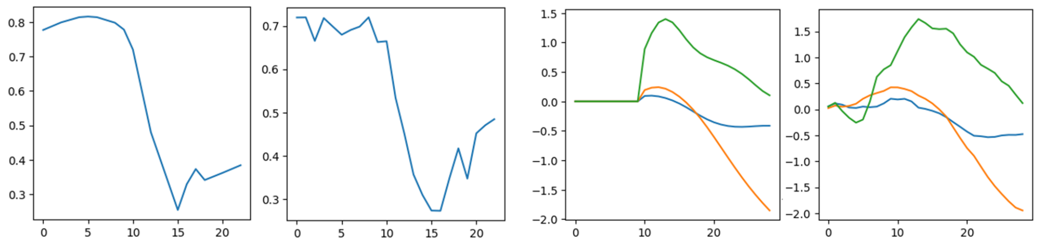
To prove the class-specific and meaningful behavior of the prototypes, the original time series is replaced once with the most positive and once with the most negative influencing prototypes. Table 11.3 shows that the replacement with the most confident prototypes corresponding to the predicted class achieved results close to the default accuracy, whereas the best fit prototype of a different class dramatically decreased the performance as the prediction switched. These results indicate that the prototypes are class-specific.

However, the second sanity check investigated into the need for the decoder to produce latent representations that are close to the representative prototypes. Table 11.4 shows that for the *Character Trajectories*, *50words*, and the *FordA* dataset there is a significant difference if the decoder gets excluded. Also, the representative and decoded prototypes and visualized two prototypes are compared in Figure 11.6 highlighting the small difference between the selected representative sample (left) and the decoded one (right).

Furthermore, the latent representation of the *Character Trajectory* prototype is provided in Figure 11.7. Each plot represents one of the three channels, and the blue color encodes the part of the selected sample, whereas the orange color decodes the latent representation of the prototype. It is clearly visible that both

Table 11.4: Closeness of prototypes. The difference between representative and generated latent patch prototypes for *P2ExNet* with and without the use of the decoder are shown. Lower values are better.

Dataset	With decoder	Without decoder	Improvement [%]
50words	**0.0413**	0.2086	505.1
Adiac	0.5380	**0.4993**	-6.2
Anomaly Detection	0.6393	**0.4929**	-22.9
Character Trajectories	**0.0099**	0.5887	5,946.5
Crop	**0.4420**	0.4815	8.9
Electric Devices	0.4135	**0.3399**	-17.8
FordA	**0.7018**	1.0315	47.0
Pen Digits	**0.5123**	0.5622	9.7



(a) Crop dataset (b) Character trajectories dataset

Figure 11.6: This figure shows the representative patch based on the distance to the latent prototype and the reconstruction of the latent representation.

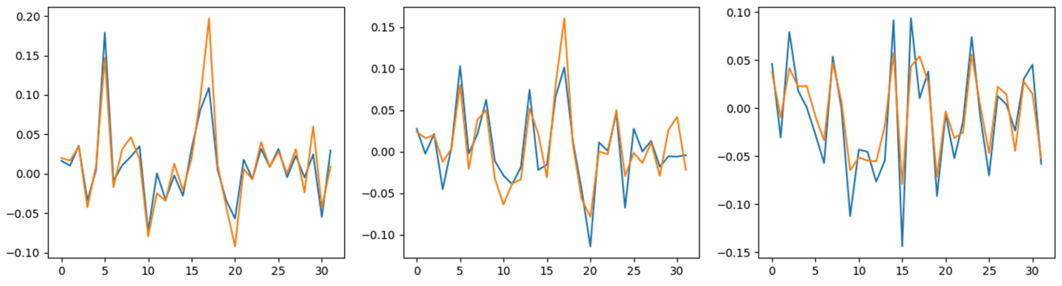
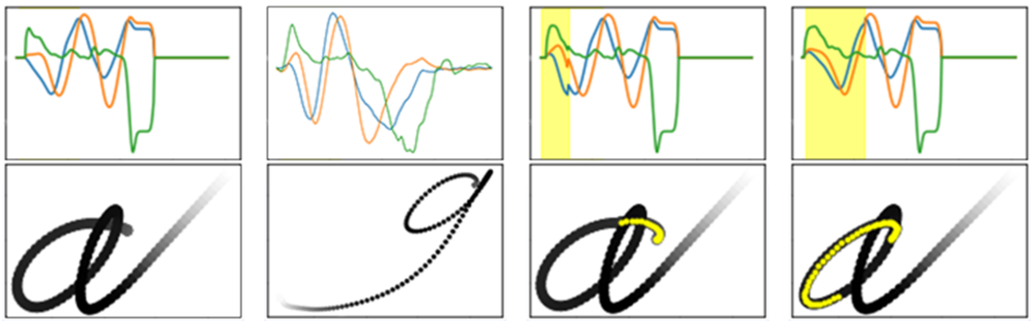


Figure 11.7: The difference between the prototype (orange) and the real sample (blue) in the latent space for each channel is shown.

latent representations share the same pattern and therefore result in a similar decoded presentation, as shown in Figure 11.6b.

11.3.4 *Comparison with Existing Prototype-based Approaches*

Furthermore, the proposed method was compared against existing work from Chen et al. [17] and Gee et al.[45]. Figure 11.8 shows the explanation of each approach for a character 'a' sample. While Gee et al. [45] points out the class with a prototype providing a single prototype capturing the complete sample,



(a) Original (b) Gee et al. [45] (c) Chen et al. [17] (d) P2ExNet

Figure 11.8: Different explanations prototype explanations of the character 'a'.

the approach from Chen et al. [17] is based on parts of the input leading to a more detailed explanation similar to *P2ExNet*. Precisely, this means additional position information is available. Lastly, the proposed method provides the same information about the location but offers re-scaling as well as an implicit comparison to other prototypes and a class distribution for the complete sample and the patches, as shown in Figure 11.4b. Furthermore, *P2ExNet* prototypes are class-specific and revertible. It is possible to decode them for a comparison with the representatives.

11.4 CONCLUSION

In summary, the novel network architecture, together with a loss and training procedure, leads to interpretable results and an inference process similar to human reasoning without significant performance degradation. Furthermore, it was proven that the proposed method works for several time series classification tasks and when excluding the class-specific prototype assignment, the approach is suitable to produce prototypes for regression and forecast tasks. Besides, the proposed method was compared with existing prototype-based methods concerning their interpretable output and time consumption, finding *P2ExNet* superior in both aspects.

Part IV

DIRECT PRIVACY

Direct privacy is the most used category of privacy approaches. The sensitive data is protected thanks to techniques directly applied to the models during the training procedure. This is mandatory as the information about the sensitive values is stored in the trained model and needs to be protected against the data leakage. Therefore, different approaches can be applied to a model and change its reasoning.

First, a comprehensive benchmark of a selected set of methods applied to time series classification is presented. As most of the privacy methods are not benchmarked for time series and their design might introduce large performance drops, it is required to evaluate their applicability for the time series domain. The evaluated methods focus on software solutions whereas hardware solutions, e.g. secure environments, are nor evaluated as these do not affect the model performance and actually do not address the problem of information leakage of the model.

Second, the effect of these methods on post-hoc interpretability techniques is shown. Intuitively, running an experiment in a protected setup preserves the information but also introduces unpractical restrictions. For the evaluation of the impact on interpretability methods, this chapter focuses on attribution techniques, as they are mostly independent of the model architecture and the changes within these maps can be evaluated precisely.

12 PPML: BENCHMARKING STATE-OF-THE-ART PRIVACY-PRESERVING APPROACHES

With the advent of machine learning in applications of critical infrastructure such as healthcare and energy, privacy is a growing concern in the minds of stakeholders. It is pivotal to ensure that neither the model nor the data can be used to extract sensitive information used by attackers against individuals. However, safety-critical analysis concerning the application of privacy-preserving approaches on time series is currently underrepresented. This work validates the efficacy of encryption for deep learning, the dataset dependence of differential privacy, and the broad applicability of federated methods.

12.1 DATASETS

A subset of datasets from UEA & UCR [9] repositories was selected for the experimentation, addressing privacy critical classification tasks from some of the most critical sectors to benchmark the applicability and performance of existing privacy-preserving methods on sensitive time-series data. The datasets cover high-stakes fields such as energy, communication, transportation, industry, and healthcare. In addition to the variety of tasks, sequence lengths, numbers of channels, and dataset sizes, the subset addresses different types of data including sensor or EEG/ECG data. Table 12.1 lists the different characteristics of the datasets used in this study.

12.2 EXPERIMENTS & RESULTS

Figure 12.1 outlines the experimental setup in which different privacy-preserving data analysis methods are applied on the same pre-processed data to assure commensurability. Throughout the experiments, Differentially Private SGD algorithm (*DP-SGD*) [1] and *FedAVG* [53] were used as techniques for Differential Privacy (DP) and Federated Learning (FL). In the case of federated training, data is split into N distinct data silos and experiments were conducted five times with different reproducible splits to account for variations in data distribution. In addition, the best run out of the five was always selected as representative performance for the corresponding setup.

A one dimensional version of AlexNet [67] was constructed as a baseline. Due to its sufficiently large number of parameters, the network can properly generalize on the utilized datasets while still remembering parts of the training data, thus leaving room for improvement of data privacy. Every model was trained for 100

This chapter is an adapted version of the work presented in: D. Mercier et al. "PPML-TSA: A modular privacy-preserving time series classification framework." In: *Software Impacts* (2022), p. 100286.

Table 12.1: UEA & UCR Datasets related to critical infrastructure.

Sector & Dataset	**Train**	**Test**	**Steps**	**Channels.**	**Classes**
Communications					
UWaveGestureLibraryAll	896	3,582	945	1	8
Critical manufacturing					
FordA	3,601	1,320	500	1	2
Energy					
ElectricDevices	8,926	7,711	96	1	7
Food and agriculture					
Crop	7,200	16,800	46	1	24
Strawberry	613	370	235	1	2
Information Technology					
Wafer	1,000	6,164	152	1	2
Public health					
ECG5000	500	4,500	140	1	5
FaceDetection	5,890	3,524	62	144	2
MedicalImages	381	760	99	1	10
NonInvasiveFetalECGThorax1	1,800	1,965	750	1	42
PhalangesOutlinesCorrect	1,800	858	80	1	2
Telecommunications					
CharacterTrajectories	1,422	1,436	182	3	20
HandOutlines	1,000	370	2,709	1	2
Transportation systems					
AsphaltPavementType	1,055	1,056	1,543	1	3
AsphaltRegularity	751	751	4,201	1	2
MelbournePedestrian	1,194	2,439	24	1	10

epochs using softmax cross-entropy loss with early stopping, SGD optimizer, if not stated otherwise. The learning rate was halved upon plateauing of the validation loss.

12.2.1 *Performance Benchmark*

Preserving privacy in data analysis usually involves the disguise of sensitive information, and therefore an inherent trade-off between privacy and model performance. The missing information would have potentially contributed to solving the problem at hand, as targeted partial disguise of non-relevant information is nearly impossible in complex, high-dimensional data. In the first experiment series, AlexNet was evaluated for all privacy-preserving methods mentioned before, comparing their performance on the complete selection of datasets. This performance benchmark does not yet provide any information about the amount of preserved privacy but serves as an initial comparison of the

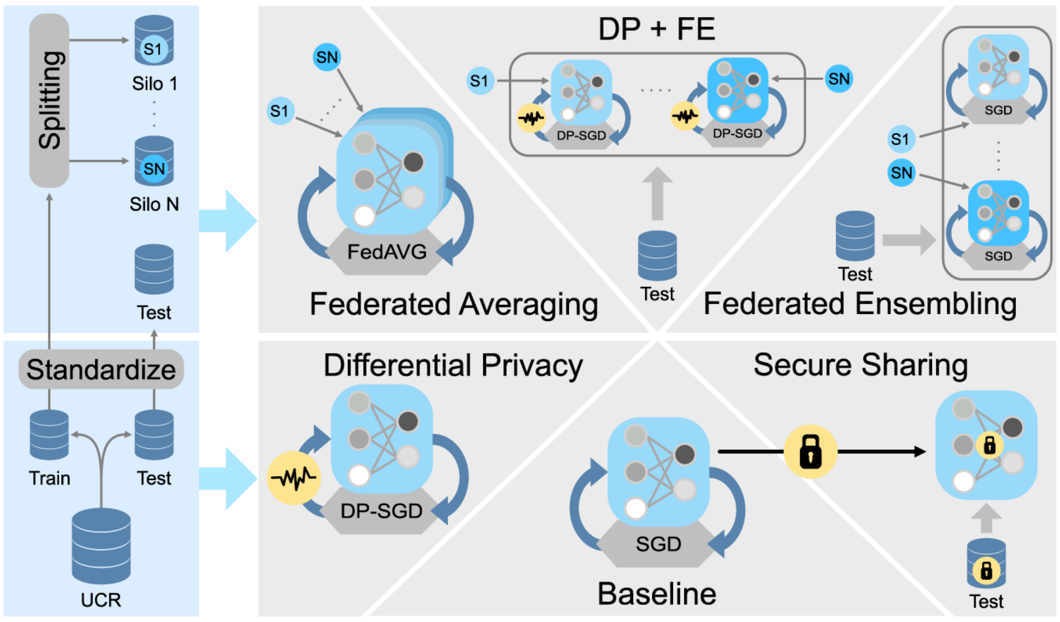


Figure 12.1: Visualization of the different approaches, their combination and data used by those methods. Differential Privacy (DP) + Federated Ensemble (FE) refers to the fusion approach using differential privacy and federated ensemble.

baseline models' performance as compared to the application of PPML methods such as DP, FL, and Secure Sharing.

A direct comparison of the approaches highlights a prevalent performance decrease when applying methods with higher privacy levels. However, most performance losses were in a reasonable frame which would not impede practical application. Table 12.2 shows the detailed comparison of weighted F1 scores for all evaluated methods and datasets.

Except for some datasets, comparable performance has been achieved by the *DP-SGD* approach. However, all datasets except for *HandOutlines* and *AsphaltRegularity* exhibit varying drops in performance. It appears that the application of DP overall results in notable performance losses for many datasets. This might indicate a sensitivity of neural networks regarding the clipping of gradients and the addition of noise. Both privacy and performance highly depend on the selection of the correct hyperparameters. Further experiments indicated that there seems to be no general rule, except for empirical testing, leading to suitable hyperparameters resulting in an optimal trade-off.

Overall results show a similar performance loss as compared to DP, disregarding small datasets resulting in non-converging models. Surprisingly, the simple ensemble approach has shown much better performance compared to both previous approaches. However, it has to be noted that in contrast to *FedAVG* and DP, the ensemble does not provide any protection against model inversion or similar privacy attacks. The coexistence of multiple models trained on fewer data could even simplify such attacks in some cases.

Table 12.2: Comparison of baseline AlexNet model and different privacy-preserving methods, reporting best weighted F1 scores in percentage. *N* corresponds to the number of clients used in federated settings. Results of models that did not converge are struck out.

Dataset	**Baseline**	**Diff. Privacy**	**FedAVG N=2**	**FedAVG N=4**	**Fed. Ens. N=2**	**Fed. Ens. N=4**
AsphaltPavementType	88.30	81.90	85.22	80.88	88.93	86.22
AsphaltRegularity	98.93	98.93	98.54	96.27	99.07	98.80
CharacterTrajectories	99.37	97.88	96.98	89.29	99.09	98.74
Crop	75.16	48.70	56.64	38.06	74.18	72.14
ECG5000	93.37	89.58	88.57	87.49	93.33	92.70
ElectricDevices	64.01	52.71	65.14	64.76	65.91	65.90
FaceDetection	63.58	51.43	62.11	62.34	64.55	64.59
FordA	92.80	91.06	90.90	85.91	93.49	93.11
HandOutlines	91.29	98.81	86.99	85.73	91.31	89.85
MedicalImages	77.20	51.21	~~34.95~~	~~34.95~~	72.14	64.13
MelbournePedestrian	94.94	86.55	18.44	20.77	86.19	87.60
NonInvasiveFetalECGThorax1	90.81	78.01	35.60	~~5.15~~	90.65	87.36
PhalangesOutlinesCorrect	82.29	~~46.60~~	~~46.60~~	~~46.60~~	79.97	78.91
Strawberry	96.77	~~50.36~~	~~50.36~~	~~50.36~~	95.43	95.41
UWaveGestureLibraryAll	96.06	90.26	92.33	89.91	95.54	93.52
Wafer	99.50	98.10	~~84.12~~	~~84.12~~	98.67	98.19
Average	87.77	75.76	68.34	63.91	**86.78**	85.45

12.2.2 *Architecture Comparison*

Furthermore, the impact of the different methods on a variety of deep network architectures was evaluated. A selection of five common Deep Learning (DL) architectures (AlexNet, LeNet [72], Fully Connected Network (FCN), Fully Dense Network (FDN), LSTM) was used for the experiments to assure significant claims. It is important to notice that the models vary in their number of parameters, and no comparison across the models was done. The focus was on each architecture in an isolated way to evaluate the impact of the privacy methods applied to them. The FCN and FDN structures are aligned with the respective parts in AlexNet. The LSTM consists of two bidirectional LSTM layers. These architectures cover the main set of layers used in time series analysis. Moreover, transformer architectures were excluded due to dataset-specific knowledge and embeddings required as well as further obstacles like model size, computational expense and lack of compatibility with the used frameworks.

Table 12.3 shows that the average performance trade-off of AlexNet when using privacy methods is superior to the other models when applying privacy-preserving methods. In addition, LeNet resulted in bad performance across almost all setups. Considering the generally lower performance of LeNet on the baseline models, it can be assumed that these issues arise from the reduced model capacity as compared to AlexNet. Furthermore, only the AlexNet and the FDN network were able to converge across all the setups, whereas the FCN and LSTM converged for all setups except one. However, besides LeNet all methods showed to be compatible with the most common privacy-preserving methods.

12.2.3 *Differential Privacy: Hyperparameter Evaluation*

The impact of different hyperparameters on the privacy obtained by *DP-SGD* was evaluated next. However, first, it is important to understand the mathematics and meaning behind differential. Differential privacy is defined by a privacy budget that defines the loss of privacy when using the approach. To retain a certain privacy loss, approaches such as *DP-SGD* [1] introduce noise during the training such that an adversarial cannot differ between individuals from the dataset and those of an adjacent dataset. Mathematically, it is possible to calculate the loss of privacy using the ϵ-differential privacy as it is defined in Abadi et al.[1]. Furthermore, as the ϵ-differential privacy comes with certain limitations, it is possible to use the (ϵ, δ)-differential privacy to ensure privacy with a certain probability. Below the definition as mentioned by Abadi et al.[1] given: Let $\epsilon > 0$ than a random function $\mathcal{M} : D \to R$ with domain D and range $\mathcal{R}$ is ϵ-differential private if Equation 12.62 holds for all adjacent datasets $d, d' \in D$ and for any subset $S \subseteq R$.

$$\Pr\left[\mathcal{M}\left(d\right) \in S\right] \leqslant e^{\epsilon} * \Pr\left[\mathcal{M}\left(d'\right) \in S\right] \tag{12.62}$$

When ϵ is small, an adversary cannot distinguish whether database d or d′ is the private database. Respectively, a larger value of ϵ means that the adversary can determine which database or dataset is the real one and which one was generated.

Table 12.3: Comparison of different model architectures reporting weighted F1 scores in percentage. Four clients were used for the federated approaches. Results of non-converging models are struck out.

Model	**Dataset**	**Baseline**	**Privacy-Preserving Methods**			
			DP	**FedAVG**	**Fed. Ens.**	**Average**
AlexNet	ECG5000	93.37	89.58	87.49	92.53	
	ElectricDevices	64.01	52.71	64.76	62.80	
	FordA	92.80	91.06	85.91	92.80	
	Average	83.39	77.78	79.39	**82.71**	**79.96**
LeNet	ECG5000	87.70	~~43.03~~	~~43.04~~	43.04	
	ElectricDevices	63.28	~~60.18~~	31.10	61.11	
	FordA	~~31.58~~	~~35.11~~	~~35.61~~	35.12	
	Average	60.85	46.11	36.58	**46.42**	43.04
FCN	ECG5000	88.43	88.51	86.91	91.67	
	ElectricDevices	50.05	46.16	~~9.46~~	60.11	
	FordA	69.70	84.38	61.86	91.82	
	Average	69.39	73.02	52.74	**81.20**	68.99
FDN	ECG5000	93.08	88.24	89.89	90.61	
	ElectricDevices	51.50	53.56	52.89	52.63	
	FordA	82.58	67.01	80.30	76.52	
	Average	75.72	69.60	**74.36**	73.25	72.41
LSTM	ECG5000	92.57	85.38	85.98	89.10	
	ElectricDevices	70.33	62.18	57.30	62.12	
	FordA	42.22	~~0.00~~	42.34	48.03	
	Average	68.37	49.19	61.87	**66.42**	59.16

The ϵ-differential privacy, also called (ϵ,0)-differential privacy, is a special case of the (ϵ, δ)-differential privacy as $\delta = 0$, and therefore delta was removed from Equation 12.62. To relax the privacy making more applicable, a delta on the right side of the equation is added as the probability of privacy leakage. Equation 12.63 shows the privacy budget including the delta parameter. Typically, the delta parameter should be smaller than the inverse of the dataset size, as otherwise too much information is accessible by an adversarial. It is important to understand that ϵ-differential privacy ensures that the observed output is almost equally likely to the output observed using a neighboring database. However, this does not hold for the (ϵ, δ)-differential privacy as this formulation guarantees the bounded ϵ only with a probability of at least $1 - \delta$.

$$\Pr\left[\mathcal{M}\left(d\right) \in S\right] \leqslant e^{\epsilon} * \Pr\left[\mathcal{M}\left(d'\right) \in S + \delta\right] \tag{12.63}$$

During the experiments, *DP-SGD* was used to ensure the privacy of the classifier. A common approach to apply *DP-SGD* is to use the Gaussian noise mechanism to introduce noise to the answer of a network, making it less

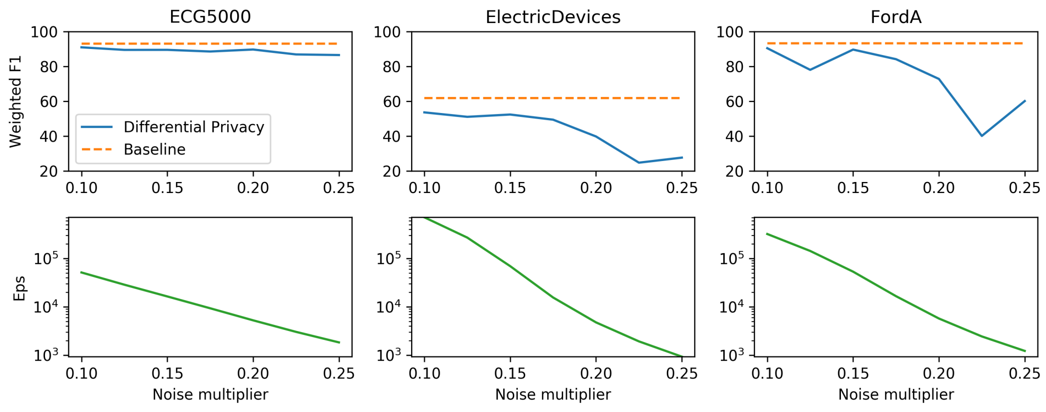


Figure 12.2: Evaluation of the loss in weighted F1 score and change in privacy when using different noise multipliers. Lower values of *Eps* correspond to higher privacy.

vulnerable to an adverse. Equation 12.64 shows how $\mathcal{M}(d)$ is calculated using the Gaussian noise distribution defined as $\mathcal{N}\left(0, S_f^2 * \sigma^2\right)$ with a mean of zero. Furthermore, $S_f * \sigma$ defines the standard deviation.

$$\mathcal{M}(d) \triangleq f(d) + \mathcal{N}\left(0, S_f^2 * \sigma^2\right) \tag{12.64}$$

Three representative datasets from different domains were selected to perform the experiments to understand the impact of the noise multiplier on the privacy budget and the performance. These datasets were selected due to their varying sequence lengths, training data sizes, and the number of classes. The *FordA* dataset covers an anomaly detection task, whereas the *ECG5000* and *ElectricDevices* datasets cover classification tasks. All parameters except for the noise multiplier were kept fixed, as they have only an insignificant impact on the privacy/accuracy trade-off. Each run is performed with gradient clipping threshold set to one and a batch size of 32. Moreover, the impact on the privacy level was examined when changing each training parameter isolated. This impact can be computed independently of model training and is therefore evaluated using numerous different conditions.

Figure 12.2 provides detailed insights on the impact of noise on model performance and the corresponding change in privacy. The *Eps* value on the y-axis is an indicator for privacy. A detailed explanation of the parameter, including the mathematical background, can be found in [95]. It is enough to note that *Eps* depends on multiple different parameters, and that lower values indicate higher privacy levels. The ratio of noise added to the gradients is controlled by the noise multiplier n_ϵ, where the gradient is left unaltered for $n_\epsilon = 0$, but privacy is only increased for $n_\epsilon > 0$. Larger values of n_ϵ bear the risk of generating noise that dominates the actual gradient information, rendering fine-tuning crucial.

The results show that for all datasets, the performance decreases significantly after a certain value of n_ϵ. *ECG5000* exhibits a relatively low and linear decrease of 3% when changing n_ϵ from 0.1 to 0.25. This does not hold for the remaining datasets. On the *ElectricDevices* dataset, a stable F1 score up to $n_\epsilon = 0.175$ was achieved. For higher n_ϵ significant drops in the performance are shown. Similar behavior as exhibited by *FordA*. Moreover, *FordA* covers a binary anomaly

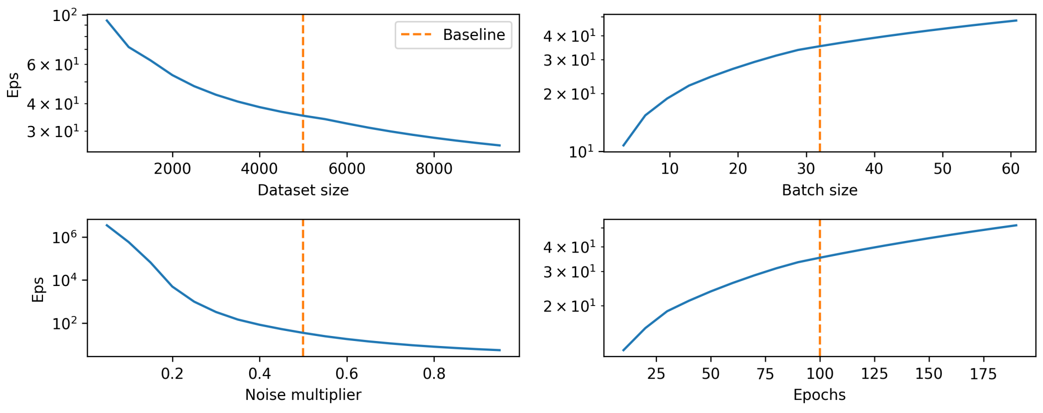


Figure 12.3: Evaluation of different parameters regarding the privacy. Lower *Eps* values correspond to higher privacy.

detection task that reflects an unacceptable performance loss for noise multiplier values larger than 0.2.

The results can be summarized as follows: The *Eps* value is a good and inexpensive indicator that can be used to provide a solid estimate of the privacy achieved in a specific parameter setup, before model training. However, its absolute value is difficult to interpret and greatly depends on the dataset. The noise multiplier n_ϵ has a drastic impact on the model performance, but this impact is dependent on the data distribution and problem at hand.

Another important aspect when applying DP is the impact of other parameters such as dataset size, batch size, and the number of epochs. Using the estimation approach mentioned above, the expected *Eps* values were calculated in a controlled environment. As start point, a fixed setup using 5,000 samples, 100 epochs, batch size 32 and n_ϵ: 0.5 was used.

Only one of the parameters was changed at a time to assess the impact of parameters independently. The results are presented in Figure 12.3. The baseline is marked with vertical orange lines. Confirming intuition, the dataset size, and the noise multiplier lower increase privacy whereas the batch size and the number of epochs decrease it. The results emphasize that the method can give a good idea about the possible setup required to achieve a certain level of privacy before training. However, the consideration of *Eps* does not provide any information about the convergence guarantees, which must be adjusted through batch size and epochs.

12.2.4 *Federated Ensemble: Ensemble Size Evaluation*

The number of participating clients, as well as the amount and quality of data contributed by individual clients, are the most critical factors in federated learning. This experiment investigates the impact of increasing numbers of clients in the most simplistic case of federated ensemble. Both batch size ($b = \{8, 16, 32, 64\}$) and learning rate ($lr = \{1e-2, 1e-3, 1e-4\}$) were tuned to obtain the best performance in each setting. Federated experiments are conducted five times to account for variations in data distribution of single data silos.

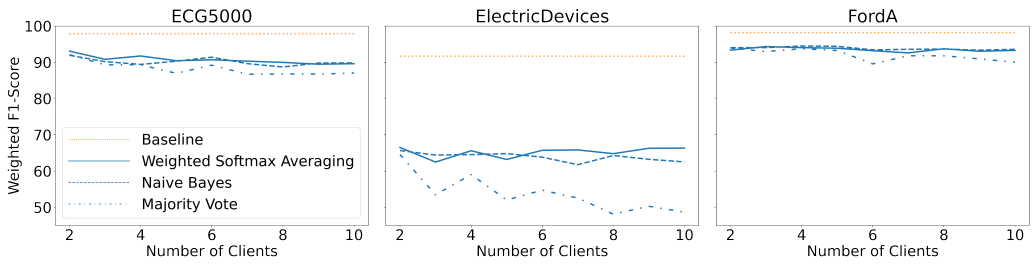


Figure 12.4: Performance evaluation of three different ensemble voting techniques. Weighted F1 scores are presented for three different datasets.

Three ensemble methods were evaluated on the federated training of *ECG5000*, *ElectricDevices* and *FordA* datasets. Figure 12.4 gives an overview of the performances achieved. The results indicate that weighted softmax averaging and naive Bayes classification achieved similar performance on the test datasets, while ensemble by majority vote resulted in the worst weighted F1 scores. It can be observed that the performance of Majority Voting follows a downward trend with an increasing number of clients. Ensembles trained on *ECG5000* and *FordA* both suffered from a minor decline in classification and anomaly detection performance, whereas the F1 score for *ElectricDevices* significantly decreases with a higher number of clients.

12.2.5 *Differential Privacy in a Federated Setting*

Furthermore, the possibility to train local data in a federated setting using *DP-SGD* at each client machine was theoretically examined to consider the resulting gain in privacy. The evaluation was done on all datasets for different number of clients $N = \{2, 4\}$ and batch sizes $b = \{16, 32\}$, with fixed gradient clipping parameter $L2 = 0.5$ and noise multiplier $n_\epsilon = 0.1$.

Combine DP & FE

Table 12.4 shows the results of combined differential private training of federated ensembles on all datasets. Many datasets showed decent performance losses over all tested settings. Overall, the results showed that depending on the dataset at hand, a combination of DP and FE can be feasible to combine its strengths. Higher performance could be achieved by extensive hyperparameter tuning on the specific use case, as experience showed that specially *DP-SGD* is sensitive to certain hyperparameters.

A combination of differentially private with federated training results in a non-linear combination of the privacy levels as the privacy achieved by *DP-SGD* depends on the dataset size, batch size, and the number of epochs, which might vary when switching from an aggregated to a federated setting. Training in a federated setting aids training data privacy in two ways, by ensuring that a client's data remains on-site and by introducing an averaging which mitigates some model inversion attacks. The additional application of differential private training on-site adds further noise to the process, which consequentially results in an overall improvement of the training data privacy. Whether a combination of DP and FE is suitable highly depends on the dataset sizes available at individual

Table 12.4: Comparison of baseline weighted accuracies using both methods separately and their combination to achieve better privacy reporting weighted F1 scores in percentage. *N* corresponds to the number of clients used for the federated approaches. *FE* corresponds to federated ensemble. Results of non-converging models are struck out.

Dataset	**DP**	**N=2**		**N=4**	
		FE	**DP + FE**	**FE**	**DP + FE**
AsphaltPavementType	81.90	88.93	78.44	86.22	77.96
AsphaltRegularity	98.93	99.07	97.87	98.80	96.40
CharacterTrajectories	97.88	99.09	97.72	98.74	97.66
Crop	48.70	74.18	63.18	72.14	62.99
ECG5000	89.58	93.33	90.01	92.70	89.52
ElectricDevices	52.71	65.91	61.22	65.90	55.39
FaceDetection	51.43	64.55	51.66	64.59	51.84
FordA	91.06	93.49	93.33	93.11	91.36
HandOutlines	98.81	91.31	68.33	89.85	87.40
Medical Images	51.21	72.14	47.53	64.13	37.95
MelbournePedestrian	86.55	86.19	87.95	87.60	86.47
NonInvasiveFetalECGThorax1	78.01	90.65	76.18	~~87.36~~	~~1.79~~
PhalangesOutlinesCorrect	~~46.60~~	79.97	62.29	78.91	50.70
Strawberry	~~50.36~~	~~95.43~~	~~50.36~~	~~95.41~~	~~50.36~~
UWaveGestureLibraryAll	90.26	95.54	93.55	93.52	92.24
Wafer	98.10	98.67	96.16	98.19	95.48
Average	75.76	**86.78**	75.98	85.45	70.34

client locations as well as the complexity of the problem, and must therefore be decided on a case-by-case basis. As previously concluded in the hyperparameter evaluation of DP, a lower dataset size, as well as a higher number of epochs, decrease privacy. Both of which are likely to be the consequence of switching from an aggregated to a distributed setting. A securely aggregated, differentially private training therefore might result in a higher privacy level as compared to local, federated training on smaller datasets.

12.2.6 *Secret Sharing Runtime Evaluation*

Training and validation runtimes are major considerations for the practical applicability of data-driven methods, especially in time-critical real-time applications. The feasibility of applying secret sharing to time series applications was evaluated by assessing training and validation runtimes, comparing the implementations of the same two-dimensional AlexNet for time series in vanilla PyTorch versus CrypTen [65]. CrypTen has been evaluated in the most basic setting, performing encrypted training with only a single client. Note that a

Table 12.5: Evaluation of runtimes over one batch of size 8. All Values are given in seconds. Used hardware: Intel Xeon (Quad Core), Nvidia GTX 1080 Ti, 64 GB memory.

Dataset	**Framework**	**Training**		**Inference**	
		Avg [s]	**Std [s]**	**Avg [s]**	**Std [s]**
ECG5000	CrypTen	35.132	0.594	8.561	0.363
	PyTorch CPU	0.105	0.019	0.024	0.002
	PyTorch GPU	**0.004**	**0.001**	**0.001**	**0.000**
ElectricDevices	CrypTen	30.196	0.145	7.113	0.030
	PyTorch CPU	0.086	0.005	0.019	0.000
	PyTorch GPU	**0.004**	**0.001**	**0.001**	**0.000**
FordA	CrypTen	68.110	0.484	18.673	0.931
	PyTorch CPU	0.186	0.016	0.050	0.005
	PyTorch GPU	**0.004**	**0.001**	**0.001**	**0.000**

Table 12.6: Performance loss for encrypted inference compared to baseline AlexNet reporting weighted F1 scores in percentage.

Model	**ECG5000**	**ElectricDevices**	**FordA**
AlexNet Baseline	**93.37**	**64.01**	92.80
AlexNet Encrypted	90.10	63.14	**93.03**

two-dimensional model was chosen to have a comparable number of parameters in both settings. Unlike CrypTen, vanilla PyTorch is not restricted to two-dimensional architectures, which results in a minor slow down.

An evaluation of training and inference runtimes comparing the implementations of the same one dimensional AlexNet for time series in vanilla PyTorch versus CrypTen gives a first estimate about the feasibility of encrypted secret sharing in practice. Table 12.5 shows that both training and inference using CrypTen is significantly slower than vanilla PyTorch in the case of CPU (roughly factor 350) and even more in the more realistic case of GPU computation. This highlights the impracticality of encrypted Secret Sharing for current real-world applications.

12.2.7 *Encrypted Inference Evaluation*

In a final experiment, a different secret sharing scenario is considered, where a model is trained on public data and encrypted for inference on secret data. The potential performance deviations arising from the encrypted evaluation of data and model at inference time were assessed.

Table 12.6 shows the weighted F1 scores obtained by private prediction on an encrypted AlexNet, trained on public data. It can be observed that the performance of *ECG5000* and *ElectricDevices* decreased negligibly, and *FordA* even increased slightly. This minor deviation of the original results is expected, as encrypted computation results in some change due to the noisy encryption.

12.3 DISCUSSION

The image domain is usually in the focus of new ML developments due to the ease of problem understanding and intuitive interpretation of context. The conducted experiments serve as a first overview of the applicability and usability of current state-of-the-art PPML applications for time series classification in safety-critical domains. The experience with available open-source frameworks showed that PPML methods applicable to time series classification already exist. However, for some applications, minor and sometimes major adjustments are required for the proper utilization as most of the frameworks are not in a productive state and offer only limited support concerning features specifically required for time-series. For instance, most of the frameworks cover only implementations for two-dimensional image processing, although time series classification is an essential modality that is used in almost all the sixteen safety-critical domains.

During experimentation, some challenges of PPML specific to the domain of time series classification were revealed. DP is a useful tool for ensuring the privacy of remote time series data. The applicability of the method is, however, strongly linked with a trade-off between privacy and accuracy, which depends a lot on the dataset and machine learning task at hand. The selection of the right hyperparameters to ideally balance this trade-off is especially complicated in real-world scenarios, where model providers have to select hyperparameters for unseen data on the client-side. In such cases, a top-down strategy is recommended in which the noise and gradient clipping parameters should initially provide maximum privacy in critical infrastructure use cases while gradually being relaxed until an acceptable model performance is achieved while data privacy is still tolerable. However, a set of possible setups can be discovered using the mathematical equation to compute the privacy value related to the differential privacy approach. Doing so provides a possible set. However, it is not possible to know the degree of network convergence without training the network using the actual setting. Summarizing the findings, it is highly beneficial to know the dataset features and their susceptibility regarding noise. Therefore, the understanding of the classification task and the value ranges can be used to approximate suitable parameters for the approach.

Both FL and secret sharing did not prove to present unusual challenges when applied to time series classification. For FL in general, but especially in time series classification, it is of prime importance that the pre-processing of data is performed identically. Whereas pre-processing of other data types such as images is much more natural and standardized, pre-processing of time series data is very application and problem dependent and must be communicated to all participating clients in a learning federation. The application of time series data partially alleviates the common downside of the high temporal and computational cost related to homomorphic or partially homomorphic encrypted computation. Despite HE exhibiting unbearable computation times, making it unfeasible for practical application in critical infrastructure, the private sharing of data for encrypted inference turned out to be a suitable approach. Furthermore, the experiments using federated learning showed that the combination of privacy-persevering methods, namely DP and FE, performs similarly well. This indicates that the combination of several feasible privacy-preserving methods can be used to develop a comprehensive privacy concept for real-world applications. Overall, the performance of FL is comparable to the DP approach. Whereas DP is more sensitive to hyperparameters like noise, FL is more sensitive towards small dataset sizes and uneven data distributions. Intuitively, the combination of both approaches suffers from both aspects and achieved a lower average accuracy but an increase in privacy. However, if certain aspects of the datasets are known, it is possible to adjust for these aspects.

12.4 conclusion

The experiments in this section benchmarked methods and open-source frameworks to provide a first overview of the applicability of PPML methods to the time series domain, which plays a crucial role in a variety of critical infrastructure application fields like energy, industry, and healthcare, and highlighted challenges specific to this particular type of input data. The benchmark covered different model architectures commonly used in the time series domain. Furthermore, a set of carefully selected datasets was used to cover various aspects regarding their domain, data shape and task. The findings highlighted that it is possible to successfully apply DP, FL, and a fusion approach to different architectures and datasets. Furthermore, the findings highlight the importance of a proper hyperparameter selection for the DP and the drawbacks of using HE regarding the computational effort.

13 PPML X XAI: INTERACTION PRIVACY-PRESERVING APPROACHES AND XAI

XAI and PPML are both crucial research fields, aiming at mitigating some of the drawbacks of prevailing data-hungry black-box models in DL. Despite brisk research activity in the respective fields, no attention has yet been paid to their interaction. This section investigates the impact of private learning techniques on generated explanations for DL-based models. The findings suggest non-negligible changes in explanations through the introduction of privacy.

13.1 DATASETS

To comprehensively analyze the impact of privacy-preserving methods on explanations, a variety of different datasets were utilized, as listed in Table 13.1. Except for the *Anomaly Detection* dataset [91], the datasets are taken from the UEA & UCR repository [9]. The selection includes both univariate and multivariate time series with different numbers of classes. The *Anomaly Detection* dataset and the *FordA* dataset consider the task of anomaly detection. The *Anomaly Detection* dataset deals with point anomalies, and the *FordA* dataset with sequence anomalies. The point anomalies are very interpretable for humans, as in their case the data is more or less noise and contains a large peak that indicates the anomaly. Even without the annotation, it is possible to understand whether the explanation for such a sample is correct or not. This is not the case for the *FordA* data, as the sequences are very long and there is no annotation. In this dataset, the anomaly can be a long part of the sequence that varies from the expected behavior. The *Character Trajectories* dataset was selected as it is possible to transform it back to the 2D input space to understand the explanation. It consists of three channels covering the acceleration within the x and y direction and the pen force, and enables precise identification of whether an explanation is good or not. In addition, the dataset size of the time series datasets differs significantly, to properly represent the influence of data volume.

13.2 EXPERIMENTS & RESULTS

A broad experimental basis, covering various domains, applications, and configurations is necessary, to make general statements about the impact of privacy techniques on explanations. Therefore, a selection of state-of-the-art classifiers is trained on a range of different datasets. Each combination of model and dataset is trained in four different settings, including training without privacy (*Baseline*), with differential privacy (*DP*), federated training (*FedAVG*),

This chapter is an adapted version of the work presented in: D. Mercier et al. “Privacy Meets Explainability: A Comprehensive Impact Benchmark.” In: *arXiv preprint arXiv:2211.04110* (2022).

Table 13.1: Shows the datasets used to evaluate the impact of PPML on XAI methods.

Dataset	Domain	Train	Test	Steps	Channels	Classes
Anomaly [91]	Synthetic	50,000	10,000	50	3	2
Character Traj.	Communication	1,422	1,436	182	3	20
ECG5000	Medical	500	4,500	140	1	5
FordA	Manufacturing	3,601	1,320	500	1	2
Wafer	Information	1,000	6,164	152	1	2

and federated training with client-side differential privacy (*FedAVG-DP*). Different explanation methods are finally applied to every model instance to compare their generated explanations.

Evaluating explanations and judging their quality is a common problem not only in XAI research [162], but also in the social sciences [93]. Multiple evaluation dimensions have to be considered to make clear statements about the impact of privacy-preserving model training on the explainability of DL-based models. Human-centered evaluation is laborious and requires domain experts. Instead, functionality-grounded methods are best suited for the domain- and dataset-wide fair comparison and quality assessment of XAI and are therefore utilized throughout this study. In the experiments, the focus was on the two main properties of explanations as defined in [162], namely their fidelity and interpretability. Fidelity measures soundness and completeness to ensure that explanations accurately reflect a model's decision-making behavior. Interpretability refers to the clarity, parsimony, and broadness of explanations, and therefore describes factors related to the ease of communication on the interface of machines and humans. Functionality-grounded methods make use of formal mathematical definitions as proxies of perceived interpretability.

13.2.1 *Experiment Setup*

Baseline networks were trained using the standard SGD or Adam optimizer with varying numbers of epochs per dataset to ensure convergence. For all other settings, training and privacy hyperparameters have been manually tuned to find a good trade-off between privacy and model performance matching the baseline. This is important to guarantee a sufficiently fair comparison between the methods, since a significantly worse network would also show worse attribution results. However, all models were trained with overall comparable settings. The reported performances correspond to the accuracies of the model performing best on the test datasets. InceptionTime [35] and ResNet [36] were used as representative networks, since they achieve state-of-the-art performances for the utilized datasets. The training data was split between training and validation with a factor of 0.9, wherever no validation dataset had been provided.

Some XAI methods pose specific requirements on the model architecture or training procedure, complicating the application of privacy-protection techniques. Therefore, this work solely focuses on the commonly used post-hoc explanations. Different attribution methods vary considerably in their realization and their

Table 13.2: Test accuracies on all datasets for different architectures and privacy-preserving settings, divided by application domain. For configurations containing DP or FL, the ϵ and n_c values are provided, respectively.

Datasets & Models	**Acc$_{\text{Baseline}}$**	**Acc$_{\text{DP}}$ / ϵ**	**Acc$_{\text{FedAVG}}$**	**Acc$_{\text{FedAVG-DP}}$ / ϵ**
Anomaly Detection			$n_c = 4$	$n_c = 4$
InceptionTime	98.74	92.87 / 5.0	98.77	89.50 / 5.0
ResNet	98.70	97.02 / 5.0	98.60	97.36 / 5.0
Character Trajectories			$n_c = 4$	$n_c = 4$
InceptionTime	99.44	91.85 / 5.0	98.82	87.26 / 50.0
				68.73 / 5.0
ResNet	99.44	85.03 / 5.0	98.19	82.10 / 50.0
				59.19 / 5.0
ECG5000			$n_c = 4$	$n_c = 4$
InceptionTime	94.38	89.07 / 5.0	93.36	89.29 / 5.0
ResNet	94.16	89.64 / 5.0	92.78	88.87 / 5.0
FordA			$n_c = 4$	$n_c = 4$
InceptionTime	95.61	92.88 / 5.0	97.70	94.17 / 50.0
				91.43 / 5.0
ResNet	94.32	86.14 / 5.0	93.94	87.12 / 50.0
				76.44 / 5.0
Wafer			$n_c = 4$	$n_c = 4$
InceptionTime	99.22	89.21 / 5.0	97.81	89.21 / 5.0
ResNet	98.75	89.21 / 5.0	89.21	89.21 / 5.0

associated underlying assumptions. Therefore, it was decided to apply a broad range of diverse methods differing in their implementations and theoretical foundations. The work covers a total of nine methods, including gradient-based *Saliency*, *InputXGradient*, *GuidedBackpropagation*, *IntegratedGradients*, *DeepSHAP*, and *DeepLift*, but also gradient-free methods such as *Occlusion*, *LIME* and *KernelSHAP*.

The Fidelity of the explanations is quantified using Sensitivity [155], Infidelity [155], and Area Over the Perturbation Curve (AOPC) [124]. The interpretability of explanations was measured using the Continuity metric. For time series, Continuity is the absolute change between each subsequent point in a sequence. Lower Continuity scores indicate better interpretability. Due to computational and time restrictions, the influence of PPML on attribution methods was quantified using a subset of the respective test sets, limited to a maximum of 1,000 examples as it is assumed this is a sufficient quantity to generalize the findings to the complete test datasets.

13.2.2 *Impact on Model Performance*

Applying privacy-preserving training techniques for DL models can have a very diverse impact on their test performances. The severity depends on multiple factors including model architecture, type of dataset, as well as the various hyperparameters for model and private training. Table 13.2 shows the results on the respective test sets for all experiment configurations when trained with different private training techniques.

Even in privacy-preserving training settings, all models converged and demonstrated acceptable accuracies. However, the best accuracies were usually achieved in *Baseline* or *FedAVG* settings. DP had a considerable impact on the models' test performances. In contrast to DP, *FedAVG* always resulted in significantly lower performance losses. The combination of *FedAVG* and DP almost exclusively resulted in a lower performance, considering a comparable ϵ-value.

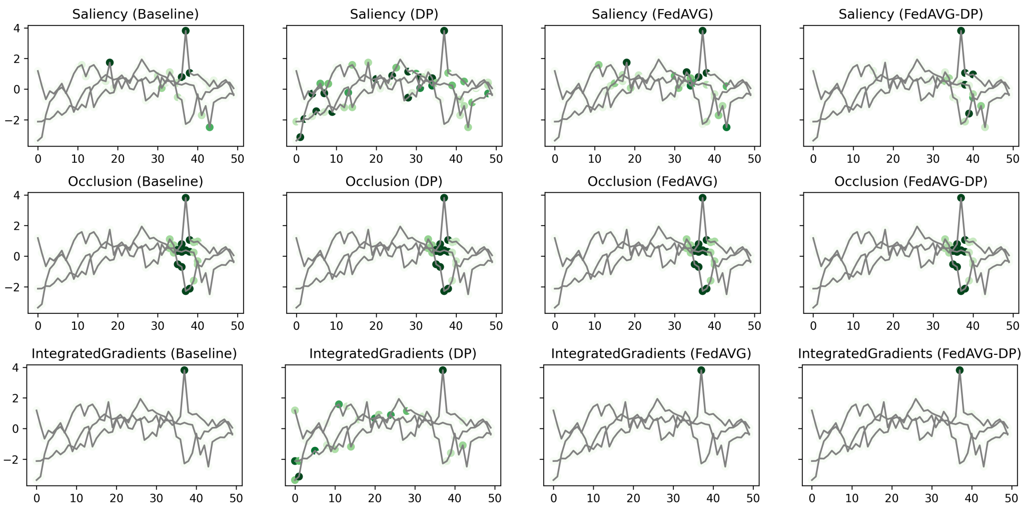
The results indicate that InceptionTime is usually affected slightly less by private training, in direct comparison with ResNet. The only exception is the *Anomaly Detection* dataset, which experienced almost no performance loss with ResNet. One possible explanation for this is the advanced architecture of InceptionTime including residual connections and inception modules. This enables the InceptionTime to be more robust against noise and outliers. All datasets, except for the *Anomaly Detection*, and *ECG5000*, considerably suffered from the combination of DP with *FedAVG*. For the *Character Trajectories* and *FordA* datasets, the ϵ-value had to be increased to achieve adequate results.

13.2.3 *General Impact on Explainability (Qualitative)*

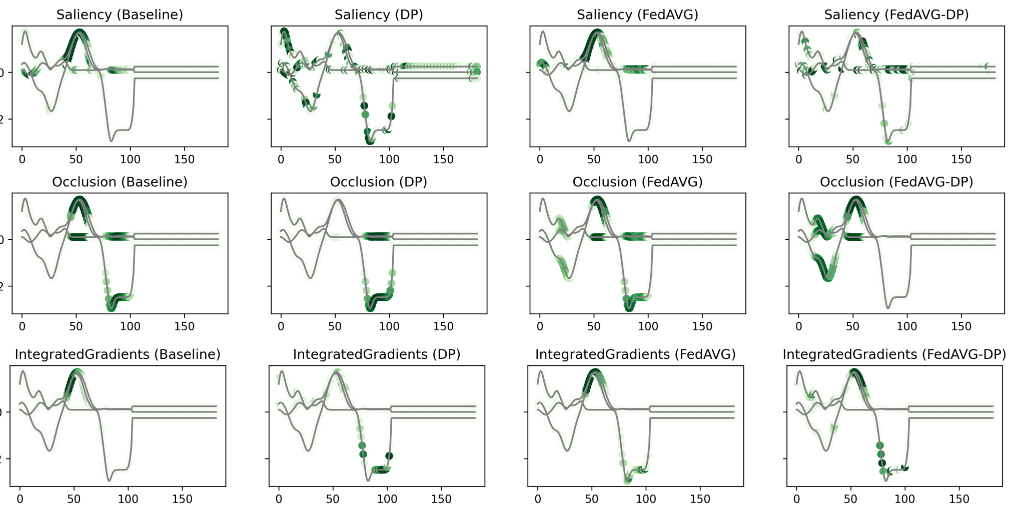
A visual inspection of individual explanations gives a first impression of the influence privacy-preserving techniques can have on the trained models. These local impressions are then further validated on the dataset level through the qualitative analysis of summary statistics.

13.2.3.1 *Individual Analysis*

Figure 13.1 shows explanations generated for the *Anomaly Detection* and *Character Trajectories*. For the *Anomaly Detection* dataset, it was observed that there is a general overlap between the explanations from different training settings, always highlighting the anomaly. In some cases, DP increased the amount of noise in the signal's relevance around the anomaly, yielding unclear and misleading explanations by highlighting distant points which do not correspond to the anomaly at all. However, this was not the case when additionally adding *FedAVG* in the *FedAVG-DP* setting. By contrast, DP-trained models showed remarkable deviations from the original *Baseline* explanation when trained on the *Character Trajectories* dataset. This observation holds not only for DP but also *FedAVG-DP* settings. *FedAVG*, on the other hand, showed explanations close to the *Baseline* setting, with only minor deviations. To capture overall trends and characteristics in attribution maps of different configurations, further analysis of dataset-level statistics of the generated attribution maps was performed.



(a) Anomaly Detection

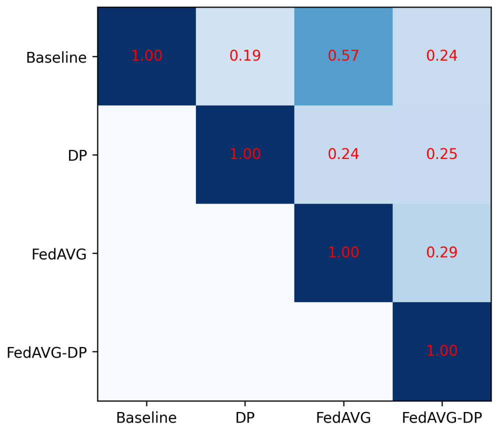


(b) Character Trajectories

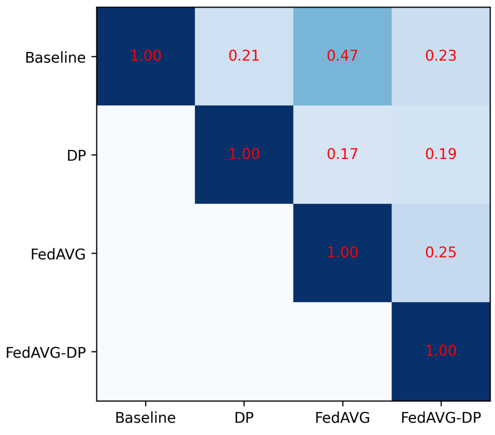
Figure 13.1: Shows the change in the attribution for ResNet for three selected samples of the *Anomaly Detection* and *Character Trajectories* dataset, respectively. DP-based training techniques tend to add additional noise and alter the explanation. *FedAVG*, by contrast, is closer to the original attribution of the baseline.

13.2.3.2 *Dataset-wide analysis*

Figure 13.2 shows the Pearson correlation of the explanations generated by different training settings for the *Anomaly Detection* dataset. Therefore, the correlation across the different training approaches was computed using all available attribution maps. Precisely speaking, the attribution between the corresponding attribution maps was calculated and the average over the number of samples was taken. The final correlation shows the score averaged over the attribution methods and the samples. For both architectures, it is evident that the privacy methods significantly change the produced attribution maps. However, *FedAVG* yields a significantly higher correlation to the *Baseline* setting as compared to the DP-based approaches. Moreover, it is surprising that the



(a) InceptionTime



(b) ResNet

Figure 13.2: Shows the average Pearson correlation of the attribution maps compared between the different privacy approaches for the *Anomaly Detection* dataset. *FedAVG* shows higher similarity to the *Baseline* setting, as compared to the DP-based approaches.

correlation between DP and *FedAVG-DP* is rather low. The remaining matrices are excluded as they showed similar results to the presented ones and do not provide any additional information.

This qualitative analysis already drew an interesting initial picture, proving that to some degree, any privacy-preserving training technique has an impact on the generated explanations. Furthermore, the results suggest that DP-trained models generate explanations that tend to be noisier and cover potentially unimportant regions, harboring the danger of misleading the explainer. However, it is unclear whether noise added by DP only concerns the explanations, or whether this reflects the model decision (Fidelity). The first results also indicated that the *FedAVG* approach can even improve explanations, leading to more focused, and meaningful explanations in some instances.

13.2.4 *General Impact on Explainability (Quantitative)*

The quantitative analysis serves as a means to further verify the findings from the previous section and allows the investigation of whether privacy-preserving techniques impact only the explanations, or also the underlying model behavior.

13.2.4.1 *Continuity*

Measuring the Continuity of an explanation helps to understand how difficult the interpretation of an explanation might be for an explainer. Humans usually struggle when confronted with high-dimensional, diffuse data. A smoother map results in a lower Continuity. Figure 13.3 shows the CD diagrams for the Continuity. The ranked results are averaged over all datasets and attribution methods. The results clearly show that the *Baseline* and the *FedAVG* settings yield better Continuity scores as compared to the DP-based

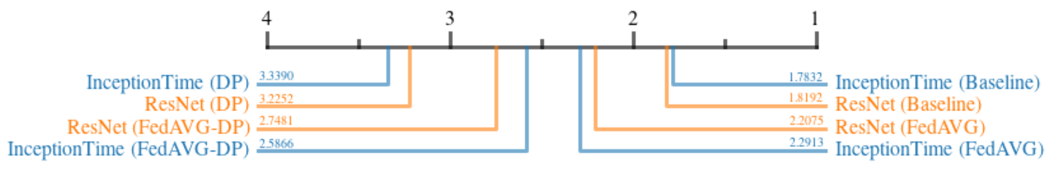


Figure 13.3: Critical difference diagram for the Continuity of models trained on the different datasets. Privacy results in less Continuity and therefore noisier explanations.

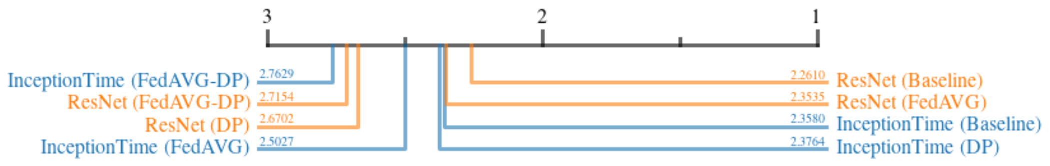


Figure 13.4: Critical difference diagram for the AOPC of models trained on the different datasets. The baseline showed the best performance.

approaches. This confirmed that DP-based private models generate significantly more discontinuous attribution maps compared to the other training techniques. Comparing only *Baseline* and *FedAVG* models, the results emphasized that the *Baseline* shows a better Continuity compared to the *FedAVG*. Moreover, *FedAVG-DP* achieved better ranks as compared to DP. DP and Non-DP approaches even show a clear visual separation in the Critical Difference Diagrams (CDDs).

13.2.4.2 *Area over the Perturbation Curve*

The AOPC measures how removing features deemed relevant by the explanation affects local model predictions. This provides important insights into the Fidelity of the explanations. Intuitively, removing features with lower importance should affect the prediction less, whereas the deletion or perturbation of important features should result in significant prediction changes. In this experiment, features were removed sequentially starting with the most important, as per the attribution map. Figure 13.4 shows all critical difference diagrams for the AOPC measure. The most prevalent pattern was that Non-DP-based settings occupied the higher ranks. ResNet showed the clear superiority of the *Baseline* and the *FedAVG* compared to DP and *FedAVG-DP*. For InceptionTime, DP surprisingly achieved almost similar performance as compared to the *Baseline*. Apart from this outlier, the presented results suggest that adding DP during training decreases the explanation's fidelity.

13.2.4.3 *Infidelity*

The Infidelity measure provides information about an explanation's fidelity by evaluating a model's adversarial robustness in regions of varying explanation relevance. Perturbations are both applied to the attribution map and the input image, while comparing the predictions of the unperturbed and noisy input. It

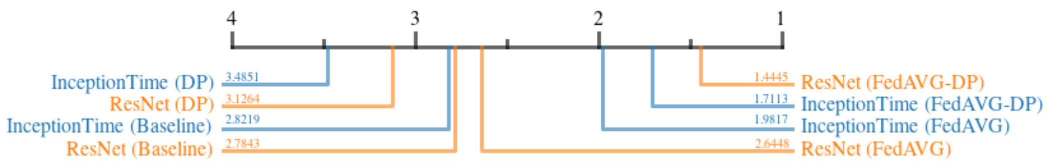


Figure 13.5: Critical difference diagram for the Infidelity of models trained on the different datasets. Federated learning showed the best performance.

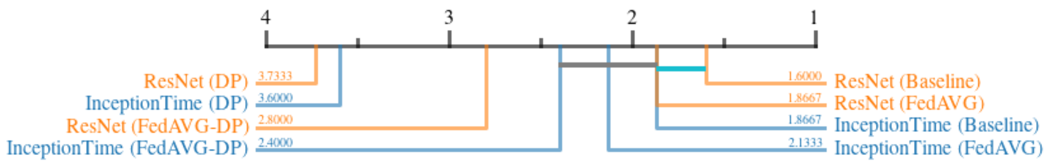


Figure 13.6: Critical difference diagram for the Sensitivity of models trained on the different datasets. *Baseline* and federated learning achieved the best results.

is expected that the perturbation of a more important feature leads to a larger change in prediction. Figure 13.5 shows the Infidelity ranks for all configurations. Approaches involving *FedAVG* achieved the highest scores. Interestingly, the addition of DP to *FedAVG* settings resulted in higher scores, whereas the sole use of DP during training led to the worst outcomes. Furthermore, the *FedAVG* approaches were the best performing approaches during the experiments. This indicates that *FedAVG* increases adversarial robustness and Fidelity, while DP alone leads to lower Fidelity.

13.2.4.4 *Sensitivity*

In contrast to Infidelity, Sensitivity quantifies Fidelity by perturbing the input directly. The change in the generated explanation is measured before and after the input is insignificantly perturbed. Small changes in the input should not result in large changes in the attribution map. Figure 13.6 shows the Sensitivity ranks for all configurations. The results show a clear ranking, with *Baseline* and *FedAVG* being superior to *FedAVG-DP*, followed by DP. However, no clear statistical distinction can be made between *Baseline*, *FedAVG*, and *FedAVG-DP* for InceptionTime, and for *Baseline* and *FedAVG* for ResNet.

13.2.5 *Impact of Noise on Different Settings*

The results so far suggest that the introduction of DP during the training process has a considerable impact on the generated explanations. Moreover, it was found that using the combination of *FedAVG* and DP can sometimes mitigate the negative effects of the added noise during the training process. These experiments cover an investigation of whether the degree to which the quality of explanations is affected, differs, for different attribution methods and datasets. Therefore, the relative increase in Continuity score was measured when comparing the *Baseline*

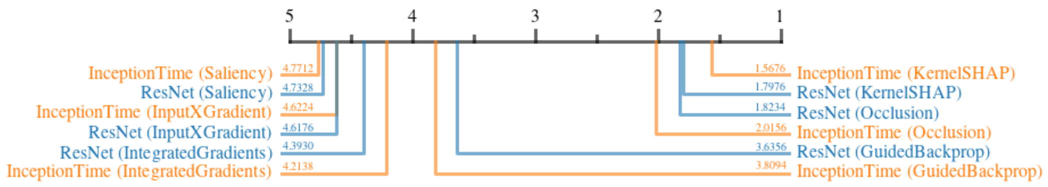


Figure 13.7: Critical difference diagram showing the impact of adding DP during training, on the quality of explanations generated by different attribution methods.

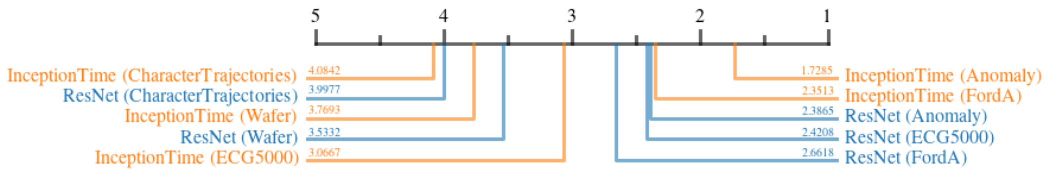


Figure 13.8: Critical difference diagram showing the impact of adding DP during training, on the quality of explanations when applied to different datasets.

with the DP training setting. A higher relative increase indicates a bigger impact, resulting in a lower rank.

13.2.5.1 *Attribution Methods*

Figure 13.7 shows the ranks of different attribution methods when applied to different architectures before and after adding DP to the training. A prominent separation of two distinct groups was noticed. Both *KernelSHAP* and *Occlusion* were affected significantly less by DP as compared to the remaining methods. Similarly, *KernelSHAP* and *Occlusion* outperformed most other methods.

13.2.5.2 *Datasets*

Figure 13.8 shows the impact of DP on the quality of explanations for different datasets. It can be seen that noise had the least impact on the *Anomaly Detection* dataset, as the decision-relevant anomaly was not affected much by the added noise. On the other hand, *Character Trajectories* dataset was highly affected by noise. This can be explained by the fact that the dataset consists of raw sensor values that describe drawn letters. Slight noise distributed over the time series can have a devastating influence on the meaning of a given sample, as the error adds up over time.

13.3 DISCUSSION

First, it was shown that not every PPML method has the same impact on model performance. DP models have shown to almost always deteriorate test accuracy. *FedAVG*, on the other hand, yielded accuracies similar to the *Baseline* setting, sometimes even improving the results. It has to be mentioned, though, that both DP and *FedAVG* follow different goals in the domain of privacy. Whereas

DP aims at preventing models to capturing individual sample information, which could be used for reconstruction, *FedAVG* mainly aims at minimizing the exposure of sensitive information by keeping the training data local. Although *FedAVG* also generates an aggregated model which might have less vulnerability to reconstruction attacks due to averaging effects, it still needs to transfer information about the local models to the orchestration server. Therefore, the combination of *FedAVG* and DP provide the highest privacy, yielding in numerous instances similar performance compared to only DP.

The qualitative and quantitative analysis revealed various interesting findings regarding the impact of different privacy-preserving techniques on explanations. DP, for example, stood out in almost all configurations for its property to add noise to the attribution maps. This has been reported in many individual samples and could be confirmed by dataset level analysis, as well as quantitative analysis, where DP-based methods stood out for increased Continuity values. One possible reason for this phenomenon is the addition of noise during the training process with DP. The introduction of noise in the parameter update most likely leads to contortions in the parameter space, which are never completely compensated, and translate into the prediction process. This effect might be counteracted by slightly tweaking the optimization, such as fine-tuning public datasets, or by increasing the batch sizes during training.

The degree to which noise is added has been investigated in the previous subsection. The results suggest, that perturbation-based methods are a lot less prone to changing their explanation's continuity when influenced with noise. The main reason for perturbation-based methods being less affected by noise in terms of Continuity is their higher resolution which neglects fine nuances in relevance, and the fact that randomly introduced noise is prone to cancel out within a patch. However, Continuity is only a mathematical approximation of an explanation's interpretability. Figure 13.1b illustrated that the occlusions are often significantly changed when introducing DP during training. Furthermore, the high interpretability of heatmaps is worthless if their fidelity is not ensured. As reported, DP exclusively led to the deterioration of metrics indicating an explanation method's fidelity. Therefore, even when applying *Occlusion*, it needs to be clarified how truthful the generated explanations remain to be.

In contrast to DP, FL often resulted in smoother attribution. Interestingly, combining *FedAVG* with DP can lead to more continuous attribution maps compared to the *Baseline* setting, reducing the negative effects introduced by DP alone. However, *FedAVG-DP* has also been reported to decrease the fidelity of explanations. Therefore, whenever XAI is required and DP is applied, it might be worth considering a combination of DP and FL. This will also be possible in cases where FL is not required, as the federated setting can easily be simulated by dividing the dataset into chunks. In addition, a better Continuity score for *Baseline* settings was observed, however, the fact that *FedAVG* shows a slightly lower Continuity has a strong theoretical basis. Averaging models during training inevitably prevents the final model from overemphasizing granular features or noise.

The presented experiments also indicated that the influence of PPML on XAI depends on the choice and feature scales of the dataset. The noise introduced by

DP has, above all, a detrimental impact on classification tasks that rely on fine-grained and nuanced features or patterns. For simpler anomaly detection tasks or tasks focusing on the detection of overall, coherent structures are seemingly less affected by privacy-preserving training techniques.

Besides the different influences PPML has on XAI, there is another fact that needs to be considered when combining both techniques. No matter how private a system has been made, exposing an explanation is in itself always a potential point of attack for a system, revealing sensitive information about the decision-making process. This is, for instance, particularly evident with *Saliency*, which provides the raw gradients of a single input instance. For truly critical applications, one should ask of whom, in the end, should be authorized to request explanations, and under which circumstances. Moreover, it might even be required to further obfuscate the exact generation process for explanations, or rely exclusively on global explanations for applications with extremely high privacy requirements.

The experiments revealed several general trends which will affect explanations on a global scale when applying private training strategies to DL models. However, one major limitation of such studies is the examined basis of comparison. When comparing explanations of separate model instances, there is always the risk of obtaining different local minima, i.e., different classification strategies. Previous research [57] suggests that one dataset can have multiple, redundant, but fundamentally different features. Therefore, even models with identical test performance could have, in theory, picked up entirely different cues to solve the same problem, hence yielding deviant explanations per model. When training models using different training strategies, it cannot be avoided to obtain models with deviating classification strategies. This is also clearly reflected in the naturally lower model performance of DP models.

Further limitations are related to the evaluation of the explanation's quality through quantitative metrics. As already mentioned, quantitative quality metrics for XAI are simply mathematical approximations of factors that could account for human interpretability or test assumptions of Fidelity that should be satisfied by good explanations. Many such metrics still have inherent limitations like AOPC, Sensitivity, and Infidelity, introducing out-of-distribution samples through the perturbation of samples.

13.4 CONCLUSION

This part of the work showed that, although the exact effect on explanations depends on a multitude of factors including the privacy technique, dataset, model architecture, and XAI method, some overall trends can be identified. It was shown that DP, on average, decreases both the interpretability and fidelity of heatmaps. However, FL was found to moderate both effects when used in combination. When used, alone, *FedAVG* was even sometimes found to improve the interpretability of attribution maps by generating more continuous heatmaps. The results suggest considering FL before DP, where appropriate. Moreover, it is recommended always to choose Differnetial Private Federated Learning (DPFL) as well as perturbation-based XAI methods, if an application requires both privacy and explainability.

Part V

INDIRECT PRIVACY

Instead of protecting the model that is trained on a classification task using private data, there is the possibility to generate data. While there are certain restrictions to the generative model, it enables the free use of the synthetic data. The core idea behind this indirect privacy preservation is that the final classification model can be public. It cannot expose private data as it has never seen the private data and is only trained on synthetic data that looks similar. One additional benefit of this is that the interpretability approaches are not biased, as they are used on the classifier only. In addition, this enables the use of intrinsic interpretability, which is not possible for direct privacy preservation without complex adjustments through the privacy mechanism and the models. The main difficulty for this indirect privacy preservation is to produce indistinguishable data to preserve the privacy of the real data and make it possible to apply the model trained on the synthetic data to the private data.

FROM PRIVATE TO PUBLIC: BENCHMARKING GENERATIVE PRIVACY

Deep learning has proven to be successful in various domains and for different tasks. However, when it comes to private data, there are several restrictions making it difficult to use deep learning. Recently, generating data in a private manner instead of applying privacy preserving mechanism directly has gained more attention. Therefore, in this section, two very prominent GAN-based architectures were evaluated in the context of private time series classification. The experiments provide evidence for the successful use of *GSWGAN* and highlight the architecture impact.

14.1 DATASETS

The experiments below were conducted across multiple datasets of the UEA & UCR repository [9]. The datasets were selected in a way that covers multiple different aspects such as a different number of channels, classes, dataset size, and problem statement. Furthermore, the selection covers different domains to emphasize the broad applicability of the methods as well as to highlight limitations. This way, the selection of datasets presents a comprehensive set covering both easy and difficult datasets concerning the generation of data. The datasets are shown in Table 14.1.

14.2 EXPERIMENTS & RESULTS

To evaluate the performance of selected GANs, private data is generated, and a classification task is performed. To do so, InceptionTime [35] was used as a classification network, as it produces state-of-the-art performances for the datasets. During pre-processing, the datasets were standardized and normalized to range between zero and one for the sigmoid function of the generator. As a baseline, an InceptionTime network was trained for each dataset using a learning rate scheduler and Adam optimizer. These models are referred to as private models as they have trained directly on real UEA & UCR [9] datasets and should not be shared.

14.2.1 *Accuracy Comparison of DP, DPWGAN, and GSWGAN*

In this experiment, the capabilities of generative models and the direct application of privacy on the classifier were compared. In addition, the difference between

This chapter is an adapted version of the work presented in: D. Mercier, A. Dengel, S. Ahmed, et al. "From Private to Public: Benchmarking GANs in the Context of Private Time Series Classification." In: *arXiv preprint arXiv:2303.15916v2* (2023).

Table 14.1: Datasets related to critical infrastructures.

Domain & Dataset	Train	Test	Steps	Channels	Classes
Transport Systems					
AsphaltPavementType	1,055	1,056	1,543	1	3
AsphaltRegularity	751	751	4,201	1	2
Communications					
CharacterTrajectories	1,422	1,436	182	3	20
HandOutlines	1,000	370	2,709	1	2
UWaveGestureLibraryAll	896	3,582	945	1	8
Wafer	1,000	6,164	152	1	2
Pulbic Health					
ECG5000	500	4,500	140	1	5
FaceDetection	5,890	3,524	62	144	2
Critical Manufacturing					
FordA	3,601	1,320	500	1	2

public, differential privacy, and gradient sanitized generative models is shown. Therefore, for each dataset first, a private classifier was trained as a baseline. Furthermore, a differentially private version of the classifier was trained. It has to be mentioned that this classifier suffers concerning interpretability, as it was trained on noisy data to fulfill the privacy constraints. Next, Wasserstein Generative Adversarial Model (WGAN) [7], *DPWGAN* [151], and *GSWGAN* [18] were trained on the private data resulting in a private generator for the WGAN and public generators for the other approaches. Using the data generated by the generator, a classifier without privacy constraints was trained. Except for the WGAN generator, this results in a public classifier that preserves privacy due to the privacy constraints involved during the training of the generator.

Table 14.2 shows the results for both the classification of the private and public generated datasets. The models and data are denoted with either a plus or a minus sign, highlighting the used data and the training conditions. Models with a minus (m-) correspond to the baseline classifier and therefore cannot be used in a public manner, whereas models with a plus (m+) are trained on generated data and can be used in a public manner, except for the WGAN approach. Data with a plus (d+) corresponds to the generated test data, and data with a minus (d-) corresponds to the private test data. Throughout the rest of the experiment section, this denotation is used. In addition, InceptionTime was always used as the architecture for the classifier, whereas the number of layers for the generative models varied based on the datasets.

Table 14.2 shows that applying Differential Privacy (DP) on the classifier directly reduces the performance by about 10% on average compared to the baseline without privacy. However, the accuracy drop strongly varied between the datasets, e.g. the accuracy for the *UWaveGestureLibraryAll* dataset declined by 48% whereas

Table 14.2: Accuracy Comparison: Shows the weighted F1 scores in percentage for classifiers trained on private or public data. Private (d-) and public (d+) correspond to the privacy of the evaluation dataset. The model (InceptionTime) was trained on private (m-) and public (m+) data. A noise multiplier of 0.5 was used for all privacy preserving approaches.

Dataset	**Base**	**DP** [1]	**WGAN** [7]		**DPWGAN** [151]		**GSWGAN** [18]	
			m- d+	**m+ d-**	**m- d+**	**m+ d-**	**m- d+**	**m+ d-**
AsphaltPavement	90.82	83.56	78.45	69.55	57.34	34.72	57.30	**72.99**
AsphaltRegularity	99.73	96.14	93.74	94.79	81.61	45.75	76.52	**83.99**
CharacterTraj.	99.65	89.55	97.16	97.49	60.06	55.74	97.72	**98.87**
ECG5000	93.18	89.55	87.69	53.29	85.03	19.06	91.89	**88.11**
FaceDetection	49.44	59.73	33.24	35.92	33.33	**34.16**	34.60	33.40
FordA	94.92	94.17	94.85	92.50	42.36	46.90	96.06	**92.20**
HandOutlines	65.40	50.02	67.63	53.47	45.73	**72.46**	67.20	53.04
UWaveGestureLib.	90.32	42.38	92.64	86.58	85.66	70.31	94.01	**84.61**
Wafer	99.79	95.21	98.80	97.69	97.94	**85.81**	98.08	85.01
Average	87.03	77.81	82.69	75.70	65.45	51.66	79.26	**76.91**

on the *FordA* dataset the drop was less than one percent. Overall, the DP approach converged for all datasets, but does not enable sharing of any data. In contrast to that, the *DPWGAN* and *GSWGAN* allow the exchange of the data as it is created synthetically. The WGAN serves as a baseline for the generative power and has shown a similar drop as the DP approach. The average performance of the WGAN was about 12% lower when the classifier was trained on public data produced by the WGAN and tested on the private. Similarly, training the classifier on the private data and testing on the public data generated by the WGAN resulted in a 5% performance loss. Intuitively, adding DP constraints to the WGAN resulted in lower performance. The *DPWGAN* has shown a decline of 36% when tested on the private data and trained on the generated data. For most of the datasets, the models showed a significantly lower performance due to the additional privacy constraints that limit the discriminator and therefore the generative capabilities. *GSWGAN* which addresses the limitations of the discriminator has shown superior performance across all datasets with an average decline of 10% compared to the baseline. *GSWGAN* was on par with the DP classifier but, in addition, enables the free use of the generated data and the trained classifier. Across all datasets, its performance was stable and could be improved further when the hyperparameters were tuned. The superior performance can be explained by the fact that privacy is only applied to the generator, as this is the only vulnerable part. It has to be mentioned, that the *FaceDetection* dataset overall has shown a bad performance independent of the classifier and generative model.

14.2.2 *Finding the Best Stopping Criteria for GSWGAN*

Training a generative model is a challenging task, as these models can collapse or produce worse samples when trained for too many iterations. Usually, when it comes to privacy-preserving approaches, these are trained for a fixed number of iterations based on the available privacy budget. However, finding a good compromise between the budget available and the performance is difficult to measure. In addition, training a generative model is costly and takes time. Therefore, it is necessary to find a good stopping criterion. Below, the evaluated criteria are listed:

- Training the Generative Adversarial Network (GAN) without any stopping criteria for a fixed number of iterations results in using the maximum privacy budget.
- The Frechet Inception Distance (FID) introduced by Heuse et al. [54] measures the distance of samples in the latent space of a classifier model. Using this measurement provides evidence that the generated data is close to the original when considering the abstract latent space representation.
- The Inception Score (IS) proposed by Salmimans et al. [121] computes the similarity of the distribution based on the samples. The idea is to measure the quality of the samples based on their similarity.
- Loss and accuracy are two well known direct metrics. However, these do not provide any information about the quality of the generated data, but rather focus on the performance of the classifier attached to the GAN. The drawback of this method is that it can produce data that looks different from the original data, but performs well concerning accuracy.

For the metrics that require an additional classifier, the trained InceptionTime of the baseline can be used. This does not lower the privacy, as the measurements are not back propagated to the generative models. However, this means that additional time is required to create the baseline model.

In Table 14.3 the results of *GSWGAN* on a set of selected datasets are presented. The results on the remaining datasets do not provide additional insights and are excluded. Furthermore, *DPWGAN* was excluded, as the results in Table 14.2 emphasize the use of *GSWGAN*. The rest of this work focuses on *GSWGAN* as it is the most promising concerning stability and performance.

The results provide evidence that the FID score produces better results on average, leading to a higher quality of generated samples. The classifier performance using the FID showed that the average performance is 3% better than using the accuracy. The worst result was achieved using the IS score, which showed a decline of 7% compared to the FID. Training for a fixed number of iterations has shown the second worse results. This provides evidence that the FID score, which measures the similarity in the latent representation, is a good measurement to use as an early stopping criterion for the privacy-preserving training of *GSWGAN*. The largest gains were observed for the *FaceDetection* and *Wafer* dataset. In the case of *FaceDetection* and *Wafer*, the FID approach has shown an increase of 15% and 10% compared to excluding the stopping criteria. In addition, it has to be mentioned that except for the *FaceDetection* dataset, FID showed the best performance for every tested dataset.

Table 14.3: Stopping criteria: F1 scores of classifiers trained on generated data of *GSWGAN* [18] and tested on private dataset (m+ d-). Limit of 50,000 iterations and patience of 2,500 iterations. Values are given in percentage performance.

Dataset	**None**	**FID [54]**	**IS [121]**	**Loss**	**Accuracy**
CharacterTraj.	98.87	**98.95**	98.40	97.72	98.26
ECG5000	88.11	**89.39**	88.45	88.90	88.90
FaceDetection	33.40	48.33	33.56	44.07	**50.13**
Wafer	85.01	**95.76**	83.28	83.28	82.06
Average	76.35	**83.11**	75.92	78.50	79.84

14.2.3 *Impact of Architecture on GSWGAN*

The architecture of the discriminator and generator plays an important role. In this experiment, the difference between a dense and a convolutional setup for the generative models was tested on four datasets. Therefore, *GSWGAN-dense* was created using fully connected layers, while *GSWGAN-conv* uses convolutional and transposed convolutional layers. However, using convolutional layers requires defining different strides filter sizes, leading to a closer optimization of the model architecture as these parameters depend on the dataset parameters. The fully connected network does not require these parameters, making it easier to find a working architecture. Based on the previous findings, the networks are trained with the early stopping based on the FID score.

In Table 14.4 the results are shown. It is visible that the convolutional setup has achieved a better performance across all datasets. This is further reflected by the IS as the convolutional one is higher than the dense one. Concerning the performance, the convolutional setup has shown an increase of 4% on the private test data, with the largest increase of 10% for the *FaceDetection* dataset. As the training of the *GSWGAN* requires a lot of time to converge, it is essential to find an architecture that works without a comprehensive grid search on the model architecture. In the experiment presented in Table 14.4, no additional architecture search was required.

However, for complex datasets such as the *FordA*, it is required to perform an architecture search. In the case of the *FordA*, different numbers of layers, filters, and latent dimension sizes were evaluated. The experiments cover only a subset of interesting architectures, emphasizing on the most relevant differences. Additional other setups were excluded as they did not provide any useful insights. In addition, in this setup, only convolutional architectures were tested due to two reasons. First, the previous analysis has shown that the convolutional setup achieves better results, and second, the dense setup did not converge to any meaningful model.

Table 14.5 shows the subset of different convolutional setups. The baseline performance of the InceptionTime classifier for this dataset was 94.92% and the best performing architecture achieved 92.20%. Therefore, with the correct architecture, the model dropped only 2.7% in performance. However, the results provide evidence that with a worse selection of the architecture, the performance

Table 14.4: Architecture impact: Shows the F1 score of InceptionTime classification using the GSWGAN [18] approach. Higher IS is better. Performances are given in percentage. InceptionTime classifier trained on private data (m-) and on Public data (m+).

Dataset	**GSWGAN-dense**			**GSWGAN-conv**		
	m- d+	**m+ d-**	**IS [121]**	**m- d+**	**m+ d-**	**IS [121]**
CharacterTraj.	98.07	98.95	18.1667	99.30	98.67	19.2626
ECG5000	92.36	89.39	1.8705	94.43	58.31	1.8722
FaceDetection	33.79	48.33	1.0206	48.67	58.51	1.0574
Wafer	97.55	95.76	1.2756	98.74	98.98	1.3491
Average	80.44	**83.11**	5.5834	**85.28**	78.61	**5.8853**

Table 14.5: Architecture search: Shows the weighted F1 scores for InceptionTime classifier trained on private data (m-) and on public data (m+). *GSWGAN* was used to generate the data. The baseline performance was 94.92%. Performances are given in percentage.

Network parameters			**m- d+**	**m+ d-**	**FID [54]**	**IS [121]**
z-dim	**Number of Filters**	**Size**				
32	256-256-256-256	7-5-3	70.43	72.10	16.5325	1.4347
32	**512-256-128-128-64-64**	**7-5-5-3-3**	**96.06**	**92.20**	**1.3951**	**1.7158**
32	512-256-128-64	7-5-3	77.81	78.04	16.9240	1.4585
48	512-256-128-64	7-7-7	73.00	71.25	15.0901	1.4678
48	512-512-512-512	7-7-7	96.44	81.94	4.0653	1.8208
64	512-256-128-64	7-5-3	73.08	74.73	16.3632	1.4303
64	512-256-128-64	7-7-7	70.58	41.49	15.7679	1.4339

dropped to 41.49%. Furthermore, the results indicate that the increase in the z-dimension did not provide any improvement. However, the filter size and filter number play an important role. As a result, the architecture search can be expensive, especially as training a generative model is costly. An important point is that the complexity of the data does not mandatory depend on the number of classes or channels. E.g. the *CharacterTrajectories* dataset has a larger number of channels and classes but less complex patterns.

14.2.4 *Impact of Noise Multiplier on Privacy-preserving Approaches*

As the goal of privacy-preserving machine learning is to achieve sufficient privacy using the budget, it is important to understand the impact of the budget on the outcome performance. The noise used in the DP learning process is one of the main parameters that affect privacy. The selection of the noise multiplier depends on multiple parameters such as the dataset size, model iterations during training, and desired privacy budget. To better understand the capabilities of the presented architectures, it is important to test their performance drop with an increasing noise multiplier. In this experiment, the noise multiplier was increased from 0.25 to 2.0. The DP classifier was also included, although it is not possible to directly compare the performance, as the number of iterations differs. An important point is that increasing the noise multiplier decreases the epsilon value that reflects privacy. A lower epsilon corresponds to better privacy. However, there is always a trade-off between privacy and accuracy.

In Table 14.6 the results are shown for a subset of datasets. The DP classifier shows that there is a significant difference based on the datasets. Whereas the performance drop for the *ECG500* dataset was less than 2% for all multipliers, for most of the other datasets the model fails at a certain point. Especially for the *CharacterTrajectories* dataset, a noise multiplier of 0.5 was enough to decrease the performance by more than 40%. The rest of the datasets validated the finding that DP classifiers show significant performance drops compared to the generative approaches. In comparison to that, the *DPWGAN* and *GSWGAN* were able to produce datasets that resulted in converging classifiers across all datasets. The *DPWGAN* has shown a similar performance decrease with an increasing noise multiplier. However, the *GSWGAN* achieved much higher performances across all the datasets when compared to the other two approaches. Except for the *FaceDetection* dataset, *GSWGAN* performed better in every setup. In the case of the *CharacterTrajectories* dataset, the performance drop for the noise multiplier of 2.0 was 25%. The performance of *GSWGAN* does not show significant performance drops up to a noise multiplier of 1.5. Overall, the noise multiplier experiments provide evidence that the *GSWGAN* is more robust to the noise added. This is the case as the discriminator does not suffer from privacy concerns and only the generator is affected by the noise. Therefore, it is possible to have a much stronger discriminator even with high noise values.

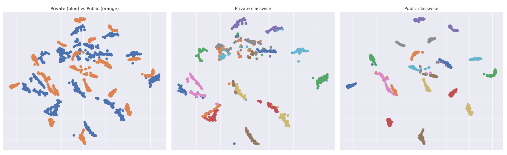
14.2.5 *T-SNE Visualization of Generated Data*

Understanding the actual dataset distribution is an important part, as in an ideal scenario, both datasets are indistinguishable. However, if the datasets are distinguishable, this does not mandatory result in a badly trained classifier. To understand the difference between the private and public data, T-SNE plots were created. The T-SNE approach transforms data into a two-dimensional space. The visualized plots show the difference between both sets. In Figure 14.1 the results of four datasets are shown. The first column of each sub figure shows the private train data and the public train data. Ideally, the data overlaps, as this states that the private and generated (public) dataset are indistinguishable. The *CharacterTrajectories* dataset shows this behavior for *DPWGAN* whereas, for the other three datasets, this is not the case. The second and third columns of the sub figure show the class distribution within the private and public train datasets. The results provide evidence that the generated data for the *CharacterTrajectories* dataset using the *GSWGAN* shares the same distribution as the original data (Figure 14.1b). The plots for the *DPWGAN* and the *CharacterTrajectories* dataset show a similar behavior (Figure 14.1a). Although the data shows a small offset, this did not hinder the performance by a large margin. Both models show a similar

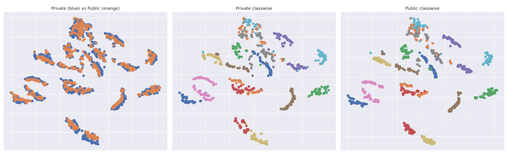
Table 14.6: Impact Noise: Shows the impact of the noise on the different architectures. The DP Baseline was trained on the private data, whereas accuracies of the GAN are computed using a classifier trained on the generated public datasets. Performances are given in percentage.

Dataset	**Method**	**Noise**				
		0.25	**0.5**	**1.0**	**1.5**	**2.0**
CharacterTraj.	DP [1]	81.14	39.86	18.24	8.83	6.82
	DPWGAN [151]	80.92	74.79	47.65	34.97	27.27
	GSWGAN [18]	**95.58**	**98.95**	**96.50**	**90.41**	**69.99**
ECG500	DP [1]	**89.61**	**89.49**	89.02	88.24	**87.65**
	DPWGAN [151]	27.39	85.45	44.90	43.02	9.53
	GSWGAN [18]	87.53	89.39	**89.44**	**89.58**	73.64
FaceDetection	DP [1]	**59.64**	**58.71**	**56.01**	33.77	33.53
	DPWGAN [151]	46.71	34.23	50.54	52.78	35.10
	GSWGAN [18]	44.68	48.33	51.14	**37.44**	**33.92**
Wafer	DP [1]	95.22	84.12	84.12	84.12	**84.12**
	DPWGAN [151]	85.33	84.12	2.10	65.07	81.18
	GSWGAN [18]	**96.92**	**95.76**	**85.52**	**96.02**	71.52

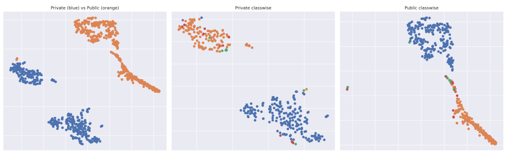
position of the 20 classes within the space. Furthermore, the results provide evidence that the dataset generated using *GSWGAN* shows a much better overlap compared to the private data. In addition, the class separation is much better compared to the *DPWGAN*. The results provide evidence that the generated data for the *CharacterTrajectories* and *FaceDetection* dataset using the *GSWGAN* share the same distribution as the original data (Figure 14.1b and Figure 14.1f). Figure 14.1c and Figure 14.1d show the T-SNE plots for the *ECG5000* dataset. This dataset consists of five classes. However, the first two classes cover 292 and 177 samples, leaving 31 samples for the remaining classes. This makes it impossible for the GANs to, correctly, learn all classes and results in generators that cover only the two main classes. In addition, the Figure 14.1c shows a large offset between the private and public data which is reflected in the accuracy as the public classifier cannot perform well on the private dataset. Furthermore, the two classes are not separated well. In contrast to that, Figure 14.1d shows that the generated data of the *GSWGAN*, although it is different from the original distribution, separated the two classes well. In addition, this model was able to perform well, although the distribution shows some differences. For the accuracy results related to this figure, the reader is referred to Table 14.2. The third dataset is visualized in Figure 14.1g and Figure 14.1h. Similar to the previous dataset, the *DPWGAN* produces a distribution with an offset, resulting in a bad performance. In contrast to that, the *GSWGAN* generated a dataset that shows a similar distribution within the T-SNE plot, resulting in high performance for the classifier trained on that dataset.



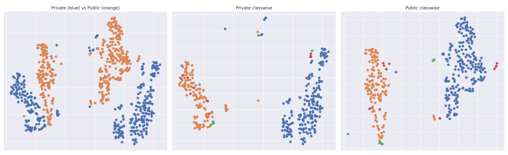
(a) CharacterTrajectories: DPWGAN [151]



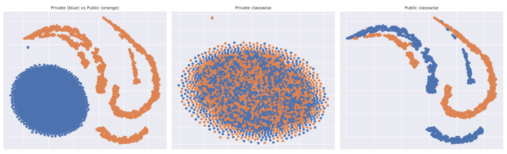
(b) CharacterTrajectories: GSWGAN [18]



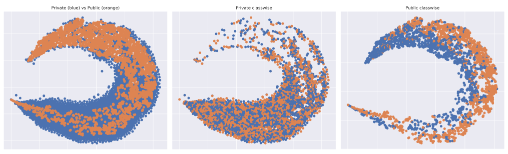
(c) ECG5000: DPWGAN [151]



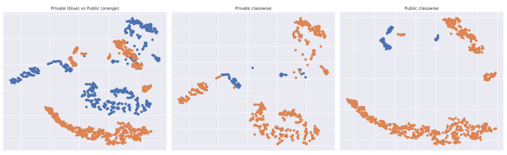
(d) ECG5000: GSWGAN [18]



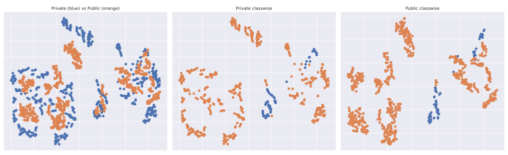
(e) FaceDetection: DPWGAN [151]



(f) FaceDetection: GSWGAN [18]



(g) Wafer: DPWGAN [151]



(h) Wafer: GSWGAN [18]

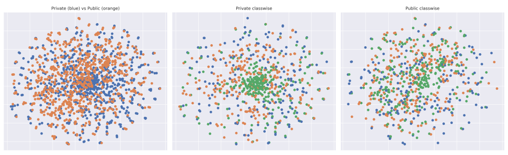
Figure 14.1: T-SNE Visualization (1/2): Subfigures show datasets generated using *DPWGAN* [151] and *GSWGAN* [18]. Plots within each subfigure: Left shows the difference between private and public train datasets. Middle: Class distribution of private train dataset. Right: Class distribution public train dataset.

One general finding across all datasets was that the *DPWGAN* produces data that is less similar to the original data. This can be explained by the fact that the differential privacy is applied to both the discriminator and generator, resulting in a worse performance of the discriminator. This is not the case for the *GSWGAN* as the privacy constraints are only applied to the generator. The stronger discriminator seems to improve the quality of the generated data.

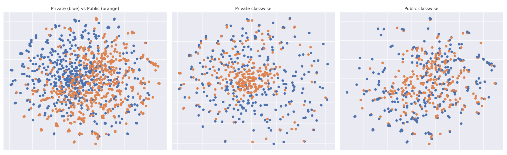
Figure 14.2 covers the remaining T-SNE plots for the *GSWGANs*. Except for the *HandOutlines* dataset, the dataset distributions of the real and the generated data show a high overlap. In addition, the class separation in the space is well for all the visualized datasets. Although the distribution of the *HandOutlines* dataset and the generated one does not perfectly fit, the results still show that the classifier trained on the generated dataset can classify the real data.

14.2.6 *Dataset Visualization - Private vs Generated (Public) Data*

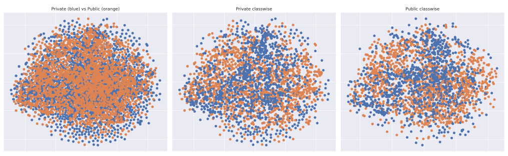
Breaking down the dataset using the T-SNE visualization shows that there is some discrepancy between the private and the generated data. However, understanding



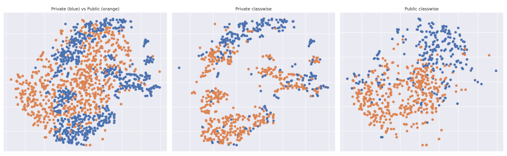
(a) AsphaltPavementType



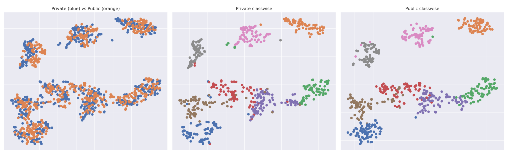
(b) AsphaltRegularity



(c) FordA



(d) HandOutlines

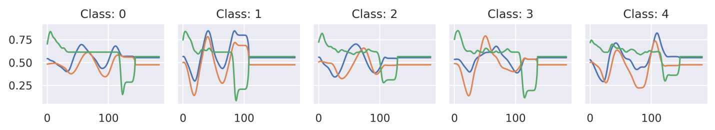


(e) UWaveGestureLibraryAll

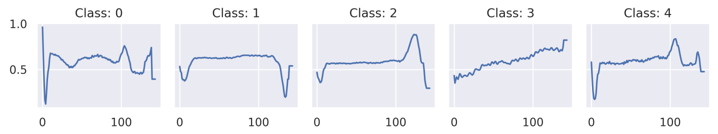
Figure 14.2: T-SNE Visualization (2/2): Sub-figures show the datasets generated using *GSWGAN* [18]. Plots within each sub-figure: Left shows the difference between private and public train datasets. Middle: Class distribution of private train dataset. Right: Class distribution public train dataset.

this discrepancy is key to understanding whether it is acceptable or not. Therefore, in Figure 14.3 real data and generated data are visualized to highlight the main differences. To create the samples for *GSWGAN-dense* and *GSWGAN-conv* the corresponding generator was used. The results indicate that the data generated using the generator preserves the dataset characteristics and has a similar shape when compared to the original data. The samples of the *FaceDetection* dataset were excluded, as visualizing that numerous channels do not provide any insights. One finding is that using *GSWGAN-conv* results in much smoother samples compared to *GSWGAN-dense*. Especially, *GSWGAN-dense* shows worse performance on the *Wafer* dataset, although the shape of those samples is not very complex. On the other side, *GSWGAN-conv* was able to produce very smooth samples for the *Wafer* dataset.

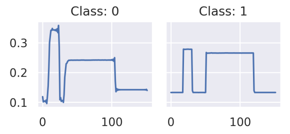
In the second part of this experiment, a sample generated using *GSWGAN-conv* is plotted against the real data to show how it fits into the original dataset. *GSWGAN-dense* was excluded, as the previous experiments indicate that it provides lower-quality samples. It is visible that the general shape of the generated sample is similar. In a latter experiment, the dataset statistics are evaluated to show that it is not a direct copy of the existing data, as it is not feasible to infer that from this visualization. For privacy, it is important that the generator produce new samples and does not memorize the existing ones, as this yields data leaks. Figure 14.4 shows the real and generated sample for the datasets. The plots provide evidence that the generated samples are not identical to the real data, but still preserve the same overall shape related to the classes.



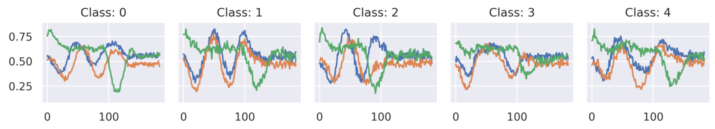
(a) CharacterTrajectories: Private data



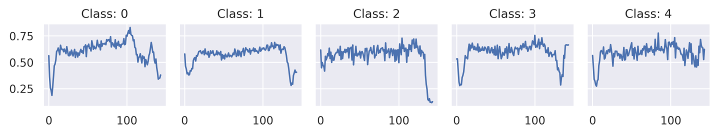
(b) ECG5000: Private data



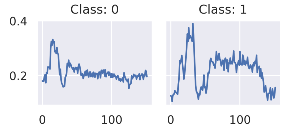
(c) Wafer: Private data



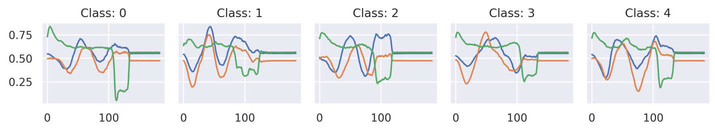
(d) CharacterTrajectories: Public data (GSWGAN-dense)



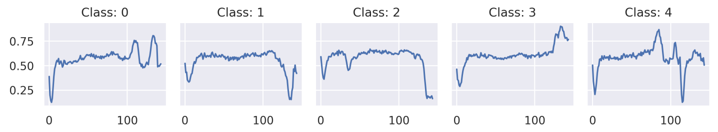
(e) ECG5000: Public data (GSWGAN-dense)



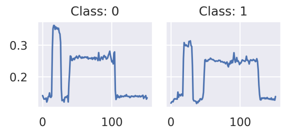
(f) Wafer: Public data (GSWGAN-dense)



(g) CharacterTrajectories: Public data (GSWGAN-conv)

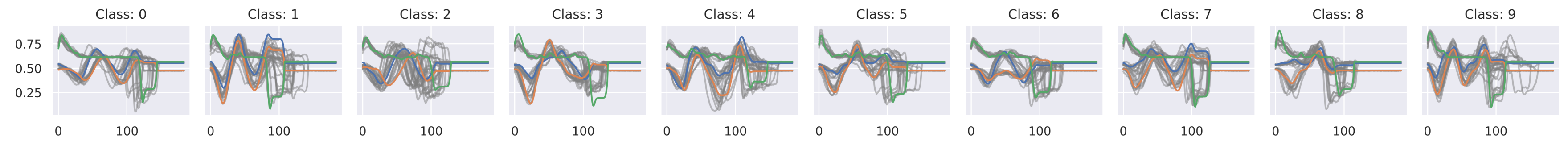


(h) ECG5000: Public data (GSWGAN-conv)

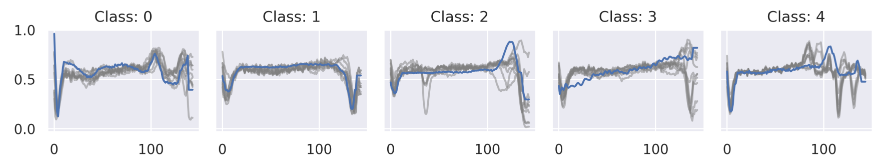


(i) Wafer: Public data (GSWGAN-conv)

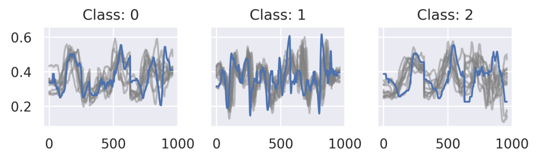
Figure 14.3: Dataset Visualization (1/2): Shows the private and the generated (public) data generated using *GSWGAN-dense* and *GSWGAN-conv*. The generated data of *GSWGAN-dense* shows a noisy behavior compared to the very realistic samples *GSWGAN-conv* [18].



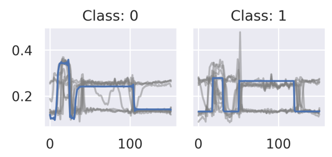
(a) CharacterTrajectories



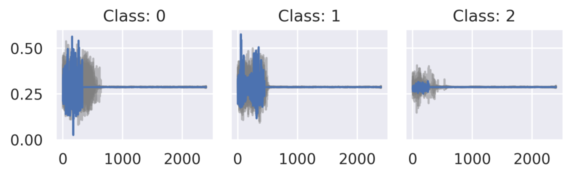
(b) ECG5000



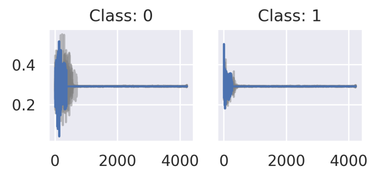
(c) UWaveGestureLibraryAll



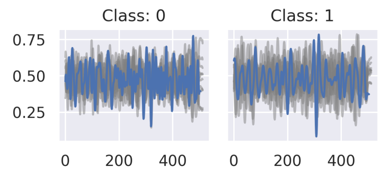
(d) Wafer



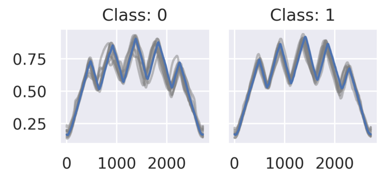
(e) AsphaltPavementType



(f) AsphaltRegularity

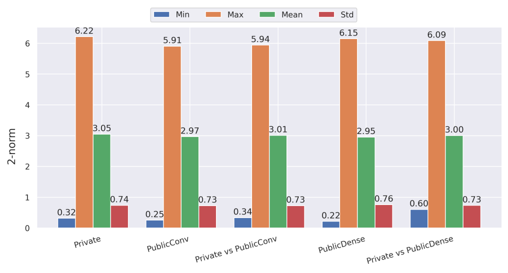


(g) FordA

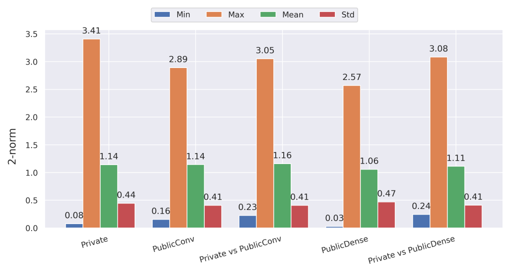


(h) HandOutlines

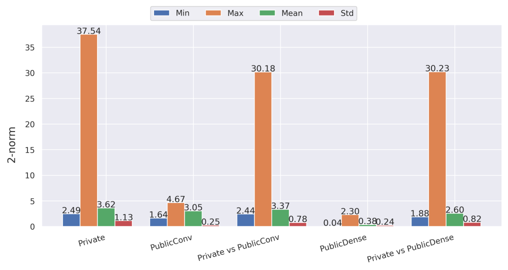
Figure 14.4: Dataset visualization (2/2): Shows datasets created using *GSWGAN-conv* [18]. Grey lines correspond to multiple original data samples. Blue corresponds to the generated data sample.



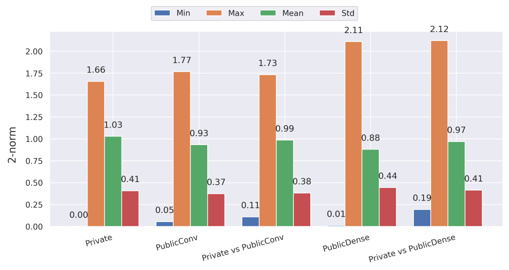
(a) CharacterTrajectories



(b) ECG5000



(c) FaceDetection



(d) Wafer

Figure 14.5: Dataset Distances (1/2): Shows the distance between the private and generated datasets using *GSWGAN-dense* and *GSWGAN-conv* and the samples within each dataset. To compare two datasets, the union of the samples was built and the L2-norm is used to compute the distance.

Especially, for complex datasets such as the *FordA* dataset, it is necessary to have very smooth and high-quality samples as the anomaly detection task is fragile to peaks and small changes in the data.

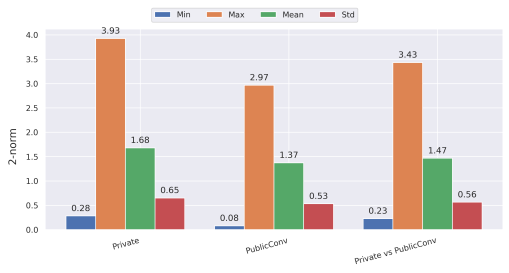
14.2.7 *Dataset Analysis - Computing the distance between samples*

To provide evidence that the generated dataset does not consist of copies of the real data but rather shows new and different samples, the distances between samples were computed. The goal was to show that the samples are different but share the same distribution, which cannot be stated by visual inspection. To analyze the differences and similarities between the private and the public datasets, the L2-norm was computed. Therefore, first, the distances within each dataset were computed. This helps to understand how diverse the data within the dataset is. However, to understand the connection between the datasets, the values were computed between the private and the public data. Therefore, the distance between every sample in the real dataset and the generated dataset was computed to identify changes in the minimal and maximal distance. Ideally, the distances should not change, highlighting that the data is neither a copy nor has the wrong distribution.

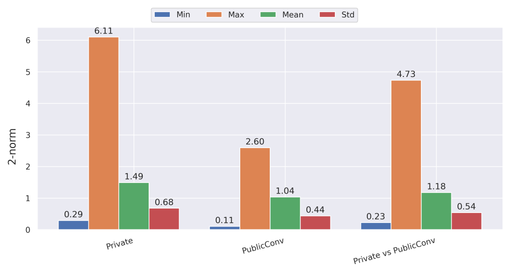
In Figure 14.5 the results are shown for four datasets which include the dense and convolutional setup. Especially, for the *CharacterTrajectories* the distances show that the distances within the private and the synthetic datasets are similar. The minimal distance value provides information about the clustering within the data, and ideally should be similar to the real private data. Otherwise, the synthetic samples could be centered around a single real data point. The max value provides insights about outliers. Comparing the values between the real and the generated dataset, the values are in the same range, highlighting that the data quality is good. A significantly lower minimum value would provide evidence that the generator copies existing data instead of creating new data. Accordingly, significantly higher maximum values correspond to outliers. The results show that except for the *FaceDetection* dataset, the distances provide evidence that the data quality is good. However, in Figure 14.5c the maximum distance within the synthetic datasets is much lower compared to its real data counterpart. This suggests that the generated data did not show a high variance. Furthermore, the mean distance for *GSWGAN-dense* was very low, which suggests that most of the data was very similar, and the synthetic data did not cover the complete distribution. In contrast to that, *GSWGAN-conv* has shown a similar mean distance as the real data. The comparison between the real and synthetic data provides further evidence that the generated data of the *GSWGAN-conv* generator successfully generates data that preserves a certain distance to the real data. For the remaining datasets in Figure 14.6, similar results can be observed. There are only small changes in the distance values, highlighting that the method successfully produces new data samples within the same space. For these datasets, only the convolutional setups are visualized, as these produced much better results.

14.3 DISCUSSION

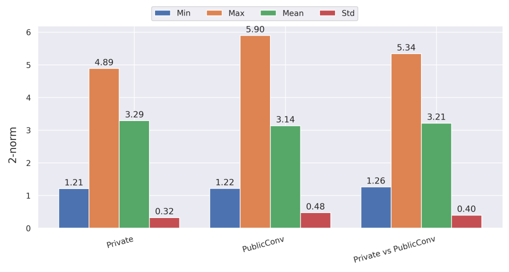
GSWGAN has shown a good performance across all the experiments. The comparison of the privacy approaches has shown that while a differential private classifier might apply to some cases, it limits not only the performance but is less stable than the *GSWGAN*. Furthermore, it can have impact methods applied on top of the classifier, such as interpretability methods. Another finding was that the *DPWGAN* has shown inferior performance for all experiments compared to the *GSWGAN*. The lower performance can be explained by the noise added to the complete architecture, whereas the *GSWGAN* only produces a private generator. To produce this private generator, privacy on the discriminator is not mandatory. A stronger public discriminator makes the network more stable. Furthermore, the experiments on the architecture strongly evidence that *GSWGAN-conv* outperforms *GSWGAN-dense* in the context of time series. The convolutional generator produced smoother samples and achieved higher accuracies. Especially, for anomaly detection tasks, it is important to use the convolutional setup for smoother samples. Furthermore, depending on the dataset, the successful application of *GSWGAN* requires an architecture search to find a suitable network. The T-SNE visualization has shown similar findings concerning the better convergence of *GSWGAN* compared to *DPWGAN*. The distribution of the generated datasets projected into a two-dimensional space is very similar to the original dataset. Finally, the dataset visualization and dataset statistics provide evidence that the generated samples are smooth and similar to



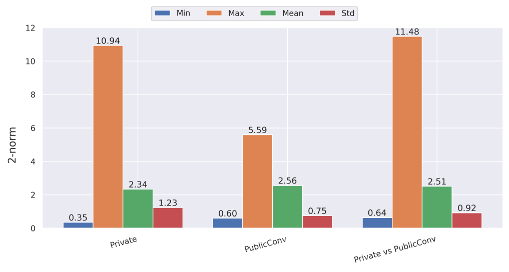
(a) AsphaltPavementType



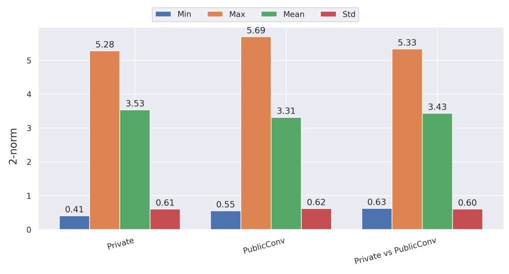
(b) AsphaltRegularity



(c) FordA



(d) HandOutlines



(e) UWaveGesture

Figure 14.6: Dataset Distances (2/2): Shows the distance between the private and generated datasets for *GSWGAN-conv* and the samples within each dataset. To compare two datasets, the union of the samples was built and the L2-norm is used to compute the distance.

the real data but still different enough from the real data in a way that the l2-norm between the real and generated data are similar.

14.4 CONCLUSION

This section benchmarked *DPWGAN* and *GSWGAN* in the context of time series classification. The results provide evidence that the gradient-sanitized approach is superior to the traditional *DPWGAN*. *GSWGAN* was able to achieve more stable performance across the datasets with the same amount of privacy. In addition, training a classifier directly, on private data does not provide better results across all datasets, highlighting that the generation of public data makes it possible to use any classifier without the limitation of differential privacy. The original *GSWGAN* uses a fixed number of iterations to train. However, the results indicate that the quality of the generator is not monotonic increasing and early stopping criteria such as the FID score resulted in superior performance for all datasets. Furthermore, the experiments show that *GSWGAN-conv* provides much smoother samples compared to *GSWGAN-dense*. In addition, the experiments provided evidence that an architecture search has a significant impact on the performance of the generative model. Concerning privacy, *GSWGAN* further shows better results when increasing the noise multiplier to achieve better privacy values. Finally, visual evidence for the correct dataset creation is provided and the distances within and between the real (private) and generated (public) data were computed, which validated the correctness of *GSWGAN* in the data generation process. The generator was able to provide high-quality data that share the same distribution as the real data.

Part VI

SUMMARY

15 CONCLUSION

This chapter discusses and concludes the findings of the interpretability and privacy components of *TimeFrame*. This covers the different perspectives that were evaluated in the context of this work and the novel approaches and conclusions that are valuable for the research. The discussion further highlights application scenarios and limitations of the approaches. For detailed numerical insights, the reader is referred to the corresponding chapter that covers the approach and all experiments.

15.1 POST-HOC INTERPRETABILITY

The Part II covered a comprehensive attribution benchmark (*Time to Focus*). The results of the benchmark have indicated that there is a need for a time series-specific attribution method that considers time series-specific features. Therefore, *TimeREISE* was proposed as a novel technique for time series attribution. Another aspect that was covered in the post-hoc part was the visualization of attribution methods and the network structure. *TSViz* was proposed as a framework that provides an explanation based on an attribution method and considers the user experience. In parallel, the problem of the complexity of time series data was tackled with *TSInsight*, a method that compresses the input data to reduce the cognitive load and focus only on data relevant for the prediction. Finally, as compression is not limited to an instance, influence functions were benchmarked in *Data Lens* to understand the relevance of samples during the training and inference.

15.1.0.1 *Time to Focus*

Time to Focus a comprehensive benchmark of existing attribution methods. A set of twelve state-of-the-art attribution methods was benchmarked across five different datasets concerning their Sensitivity, Infidelity, runtime, Continuity, insertion and deletion, and impact on accuracy. While some of the methods have shown to produce good results concerning some metrics, the analysis provided evidence that there is no superior method. Furthermore, the results provide evidence that some approaches are favored by some metrics. This is an important finding as most times attribution methods are evaluated without any information about the bias that is introduced by the used metric. E.g. perturbation-based approaches, naturally, show more robust results concerning metrics such as Continuity which evaluate the amount of noise. Furthermore, perturbation-based approaches showed better results for the Sensitivity and the insertion and deletion test. In contrast to that, gradient-based approaches are superior concerning the runtime, as they do not require a series of forward passes. Also, concerning the Infidelity, gradient-based approaches were superior. Especially the *Occlusion* showed the best performance for the perturbation-based approaches except for the runtime. Among the gradient-based approaches, the *Integrated Gradients* performed slightly better compared to others. Based on the category an attribution method belongs to, it is possible to predict its performance for the metrics. In addition, this helps to define the context in which it is possible to use the approach. E.g. real-time interpretability requires fast approaches which are mostly gradient-based, whereas if time is not a concern the perturbation-based approaches offer smoother results.

15.1.1 *TimeREISE*

TimeREISE is a novel attribution approach created to capture the characteristics of the time series domain. During the development of this approach, the findings of *Time to Focus* played a pivotal role. Motivated by a perturbation-based approach used in the image domain, the idea was adopted and led to an approach that produces high-quality results. In the insertion and deletion test *TimeREISE* outperformed five state-of-the-art methods on 17 datasets with an average rank of 2.0 and 1.6 across all datasets. A similar superiority was shown in the Sensitivity and Continuity. Concerning the Infidelity, there was no superior method. In addition, *TimeREISE* has shown to have a better runtime compared to methods such as *Occlusion* and other window-based approaches. *TimeREISE* has shown that an approach aligned to the time series data can produce superior results in almost all metrics, as it is optimized for the modality. Its superiority against the approaches introduced in the image domain and applied in the time series domain emphasizes on the importance of modality-specific evaluation of approaches. *TimeREISE* considers characteristics of time series such as the interaction between channels, and the effect of events spread over the whole time series. E.g. the interaction between channels is defined for the image domain and needs to be defined differently for the time series context. Furthermore, window-based approaches might not cover the effect of events across the whole time series. While in the image domain, most times concepts are close together if they belong to each other. This does not hold for time series. A change at the beginning of a series can take effect on the last value without showing a dramatic change in the values between.

15.1.2 *TSViz*

To bridge the lack of a visualization framework that considers the user experience to analyze the attribution and the network *TSViz* was developed. Previous benchmarks have shown that attribution methods are a valuable source of explanation, their presentation to a stakeholder requires a careful design. *TSViz* exemplary shows the possible visualization of a gradient-based approach for the time series context. In addition to the representation of the input saliency, it extends the idea to visualize the importance of the network architecture parts. During the experiments, it was shown great results on two datasets when pruning based on the importance values and the clustering of filters was performed. The pruning showed that removing the filters not relevant resulted in no increase concerning the loss, while removing the important filters increased the error by up to 8,000%. In addition, it highlights the limitations and remaining issues when it comes to time series interpretability. These limitations are mainly the amount of data that needs to be visualized in a suitable manner. While it provides the first network visualization for time series data, there is still space for improvement. However, the idea to cluster network parts and show only representative information is one step towards an improved network visualization. Furthermore, the restful application programming interface and the three-dimensional front-end provide easy access to the framework. In addition, the dashboard offers great opportunities concerning explanations aligned for different user groups. Finally, the gesture controlled virtual reality version of *TSViz* provides capabilities to interactively explore and understand the network flow.

15.1.3 *TSInsight*

TSInsight was proposed as a compression framework for time series data, as it compresses the data in a way that offers extended interpretability. Assuming that the input required by a trained network must be preserved while any other data point can be removed without dramatic changes, the results have confirmed that this is true for time series. Using a specific training procedure and loss, it was possible to produce a compression that preserved the shape of the relevant data points and suppressed the irrelevant data. Across eight datasets and compared against ten state-of-the-art approaches for, it was shown that this compression preserves the important features similar to the attribution methods. The performance dropped by less than two percentage points in the worst case. In addition, it was shown that *TSInsight* can be applied to a model without restrictions to the architecture. Furthermore, it offers a global interpretability, which is not the case for the attribution methods. In addition, the autoencoder trained to compress the data can be used to directly pre-process the data, making it more robust and to discard information that is not used by the model.

15.1.4 *Data Lens*

Data Lens covers a comprehensive influence function benchmark. Interpreting the models through the lens of data is not considered commonly in research. However, as the model is trained on a dataset and can leak information about this dataset, it is intuitive that the model stores some information relevant for the reasoning process. Using existing state-of-the-art approaches, a comprehensive benchmark was conducted. This benchmark has shown across three datasets that while advanced approaches lead to high-quality results for the mislabel detection in the image domain, it is possible to achieve similar results using direct measurements such as the *Loss*. Across the different datasets, the *Loss* showed the best mislabel detection in seven out of nine scenarios. Furthermore, the loss shows a pretty large discrepancy between correctly labeled and mislabeled data concerning the loss value. The *Representer Point* approach has shown the worst performance regarding the separation of correct and mislabeled data.The mislabel detection was selected in this context as it is related to the model and its generalization. Furthermore, this exposes interesting results concerning interpretability. The experiments show thanks to different *Influence Functions*, *Loss* and the combination of these approaches the bias of a model from a global perspective. Especially, the combination of the *Loss* and *Influence Functions* has shown to produce excellent results, identifying more than 95% of the mislabels. However, the use of the *Influence Functions* has shown a significant increase in the time consumption compared to the *Representer Points* and the *Loss*. Finally, the results indicate that the correction of the data based on the evaluated methods improved the network performance, whereas the deletion of the mislabeled data did not improve the performance.

15.2 INTRINSIC INTERPRETABILITY

The Part III covered two novel approaches for the time series analysis. *PatchX* realizes the idea of divide and conquer to reduce the complexity of a problem. Therefore, patches are used, and a hierarchical classification is put on top. Based on the findings of *PatchX* the second approach, *P2ExNet*, proposed a framework for patch-based prototypes. In addition to the divide and conquer principle, prototypes used to enable an easier interpretation based on similarity.

15.2.1 *PatchX*

With *PatchX* a novel interpretable reasoning approach was presented. The idea of dividing the data to produce smaller problems and later on combining the findings of these to produce an overall result has shown to work well in the time series context. Compared to two other state-of-the-art approaches, *PatchX* has shown to produce high-quality results for long sequences across five datasets. The accuracy results further show that the performance trade-off was marginal. In three out of five cases, *PatchX* outperformed the back-box model, whereas in the remaining cases the performance only dropped by about 1.5%. *PatchX* has shown that it is possible to produce predictions for the patches of a sample without the effort of data labeling. Only using the overall label, a novel patch transformation approach and a specific training led to impressive results that can also be used to label data. In addition, using *PatchX* it is possible to provide an explanation based on different levels which enables to go step-wise from the overall prediction down to each patch and understand the interaction. Finally, it has to be highlighted that the framework is modular, offering great applicability and the ability to extend it further. Concerning the additional time to produce this explanation, *PatchX* has shown to be faster than the SVM approach in the inference mode, while being slightly slower than the black-box model. During the training, *PatchX* requires additional time, making it slower than the SVM but still faster than the feature extraction approaches. Finally, the results indicate that it is not required to know the patch sizes precisely, as selecting multiple did not hinder the performance and quality of the *PatchX* results.

15.2.2 *P2ExNet*

P2ExNet was proposed as a novel prototype and patch-based reasoning approach. Using prototypes to point out the global behavior and produce instance-based explanations has been very prominent in the image modality. The results have shown that *P2ExNet* can produce prototypes for time series. Precisely, the prototypes are not for the complete sequence, but rather for patches. *P2ExNet* was inspired by the existing prototype-based approaches and has shown that with a minor drop in accuracy, it is possible to come up with a reasoning process based on the human reasoning behavior. *P2ExNet* was able to achieve a better performance in four out of eight datasets compared to a similar black-box CNN. The largest accuracy drop was about 5% and the largest gain was about 5%. The sanity check has shown that replacing the patches with the correct prototype preserves the accuracy, while replacing them with wrong prototypes results in a significant accuracy drop. While using the correct prototypes, the performance dropped about 6.5% in the worst case, compared to drops of more than 20% when a wrong prototype was used. *P2ExNet* produces the prototypes in the latent space, which further improves their quality as the network learns uses an enhanced representation instead of the input directly to produce the prototypes. To produce these high-quality prototypes, the decoder part is pivotal, as the results have shown that discarding it produces prototypes that are up to 6,000% further away from the selected representatives when comparing their latent representation to those of the prototypes using the decoder. Furthermore, the decoder helps to present the precise prototype instead of the closest patch within the dataset. Finally, the prototypes produced by *P2ExNet* are class dependent and have a class weight, which makes it possible to understand how relevant they are.

15.3 DIRECT PRIVACY

Part IV covered a comprehensive benchmark of existing privacy mechanisms and evaluated the impact of those on interpretability methods. *PPML* addresses the runtime and performance impact when the privacy is applied on top of the model during the training process. In contrast to that, *PPML x XAI* focuses on the change in explanations when interpretability methods are applied on top of a model that was trained in a private manner.

15.3.1 *PPML*

The *PPML* benchmark has shown across 16 datasets that differential privacy and federated learning to introduce minor up to major accuracy drops based on the hyperparameters, the network architecture and the dataset. The average performance using DP resulted in a decline of 12% across the datasets. The results of federated learning showed an average decline of about 20% whereas the federated ensemble approach showed an accuracy decline of only one percentage point. Furthermore, the results indicated that the performance of AlexNet was significantly better compared to LeNet, FCN, FDN, and LSTM. However, the behavior concerning the accuracy drops regarding the privacy method showed a similar behavior across the model architectures. It was found that while differential privacy works well for some datasets that are robust to noise, it is not possible to predict this behavior. Furthermore, the experiments emphasize on the careful combination of differential privacy and federated learning to improve the privacy. The accuracy combining the differential privacy and the federated ensemble did not show any further accuracy drop compared to the differential approach without the federated ensemble. While there are major accuracy drops when using both approaches, a hyperparameter tuning can resolve them. Another finding in this work was that the homomorphic encryption comes with significant overhead, making it unfeasible to use for time series data on a large scale. Training a network using the HE has shown to increase the training time by a factor of up to 17,000. However, the secure sharing works well and is a solution to transfer the data in a save manner.

15.3.2 *PPML x XAI*

PPML x XAI is a comprehensive analysis that has shown that the effect on interpretability methods can lead to significant changes. A broad range of nine different attribution methods was evaluated on five datasets concerning multiple aspects such as Sensitivity, Infidelity, Continuity, and others. For privacy, the most famous approaches such as the DP and FL as well as their combination were used. While some attribution methods have shown to be more robust to privacy, the overall finding was that the use of DP introduces noise to the attribution and lowers the Continuity significantly. In contrast to that, the use of FL in addition to the DP recovered the attribution results up to a certain point. Furthermore, it was shown that the gradient-based attribution methods show a significantly lower quality compared to perturbation-based approaches. Another finding was that the impact of privacy on the attribution method depends on the dataset. For some datasets, the noise significantly affects the attribution, whereas for other datasets the influence is marginal. This mainly depends on the definition of the problem and the features, as well as the pattern sizes ,and the vulnerability to noise.

15.4 INDIRECT PRIVACY

Part v covered the use of a generative approach was evaluated to understand if it is possible to exclusively work on synthetic data and transfer the findings. The goal of this was to provide an approach that does not affect the interpretability of the system and helps produce data that is easier to access.

15.4.1 *From Private to Public*

From private to public is a benchmark that covers the *DPWGAN* and the *GSWGAN* on nine different datasets. The main difference between those two approaches is that the *GSWGAN* does not apply any privacy constraints to the discriminator. The experiments have indicated that this has a huge impact, as the performance of the *GSWGAN* was significantly better compared to the *DPWGAN*. In addition, training a public classifier based on the data generated by the WGANs resulted in good performances if the GAN was optimized correctly. Further experiments emphasize that the architecture of the GAN influences the results a lot, and a hyperparameter tuning to get a suitable architecture is beneficial. The use of convolutional architectures has shown better visual performance for the generated data. However, training a convolutional network suffers from more training time compared to a dense setup. The visual and analytical inspection shows that the generated samples are close enough to the original samples to be used, but still preserve enough distance to not mimic the original data. While training the GAN is a challenging task, it comes with several advantages. The generated data can be used by third-parties without privacy concerns. In addition, it can help to overcome data shortage, as the generative model can generate the required amount of data if initially the amount of data was large enough to train the GAN. Another benefit is that the classifiers trained on the data can be shared and do not need privacy mechanisms, making it possible to use interpretability methods without the risk of misleading results due to the privacy approach.

16 FUTURE WORK

This section first shows the novel components developed in the scope of this thesis that can further be extended. Therefore, Table 16.1 shows the novel approaches that are included in *TimeFrame*, the proposed framework for interpretable and privacy-preserving time series analysis using deep neural networks. The explanation scope highlights whether the explanation provides information for each individual data point or only for the sample. The data scope defines whether the explanation is given for the instance individually or based on the whole dataset. It is visible that several different perspectives were addressed. In addition, to the proposed methods, existing techniques are included to evaluate and combine them with the proposed approaches.

Most of the approaches are developed for time series classification, and it is possible to extend them to the field of regression or forecast. In the case of *TimeREISE* the adaption to a regression task requires some changes in the score function. Instead of taking the classification confidence, it is possible to use the difference between the original prediction and the modified one. However, the basic framework to produce the attribution masks needs no further adjustments. Furthermore, it would be possible to enhance the perturbation technique used to replace masked input data points as it is currently, replacing them with the mean signal. One possible improvement could be to slightly adjust them towards the mean to have smaller changes within the signal and understand their impact. This way, the sample is more likely to be in distribution. As *TSViz* is a visualization framework that is based on gradient-based attribution methods, there are only minor changes required to support different attribution methods and regression. In addition, it is possible to enhance the visualization by supporting currently unsupported layer types such as attention layers and long short-term memory layers. Concerning *TSInsight*, *PatchX*, and *P2ExNet* only changes in the loss function are mandatory to enable their use on regression tasks. Besides the transfer to regression and layer support, one possible improvement for *TSViz* covers the better visualization with numerous channels. Also, future work could investigate in a better visualization, e.g. grouping of layers to make the network more compact in the visualization. Another idea is to group channels by their relevance to reduce the cognitive load for datasets with various channels. Concerning the main limitation of *TSInsight* are the hyperparameters which are tuned manually. Using an optimization algorithm, finding the best suitable parameters can be automated. However, this will produce a runtime increase. In addition to that, a metric to evaluate the quality of a compressed input does not exist so far. For *PatchX* one of the major limitations is that the training involves wrong labeled patches. Although the network can learn which patches are not class relevant, a possible improvement could be an advanced training mechanism that discards or weights the patches based on their class relevance. Finally, one possible improvement for *P2ExNet* covers a detailed analysis of

Table 16.1: Shows the novel approaches and their categories.

Approach	Explanation	Expl. Scope	Data scope
Post-hoc			
TimeREISE [85]	perturbation	local	instance
TSViz [91]	saliency	local & global	instance & global
TSInsight [90]	compression	global	instance & global
Intrinsic			
PatchX [84]	pattern	local & global	instance & global
P2ExNet [83]	prototypes	local	global

different autoencoder structures. While traditional autoencoders and variational autoencoders have shown similar performance, it might be that Wasserstein encoders can improve the quality of the prototypes.

BIBLIOGRAPHY

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. “Deep learning with differential privacy.” In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318 (cit. on pp. 16, 27, 28, 161, 165, 193, 198).

[2] A. Abdul, C. von der Weth, M. Kankanhalli, and B. Y. Lim. “COGAM: Measuring and Moderating Cognitive Load in Machine Learning Model Explanations.” In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–14 (cit. on p. 71).

[3] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. “Sanity checks for saliency maps.” In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. 4, 40, 59, 105).

[4] Z. Allam and Z. A. Dhunny. “On big data, artificial intelligence and smart cities.” In: *Cities* 89 (2019), pp. 80–91 (cit. on p. 3).

[5] D. Alvarez-Melis and T. S. Jaakkola. “On the Robustness of Interpretability Methods.” In: *arXiv e-prints* (2018), arXiv–1806 (cit. on p. 91).

[6] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al. “Privacy-preserving deep learning via additively homomorphic encryption.” In: *IEEE Transactions on Information Forensics and Security* 13.5 (2017), pp. 1333–1345 (cit. on p. 28).

[7] M. Arjovsky, S. Chintala, and L. Bottou. “Wasserstein generative adversarial networks.” In: *International conference on machine learning*. PMLR. 2017, pp. 214–223 (cit. on pp. 192, 193).

[8] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances.” In: *Data Mining and Knowledge Discovery* 31.3 (2017), pp. 606–660 (cit. on p. 3).

[9] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. *The UEA & UCR Time Series Classification Repository*. 2021. URL: www.timeseriesclassification.com (visited on 03/01/2021) (cit. on pp. 39, 61, 102, 113, 136, 150, 161, 175, 191).

[10] J. Benesty, J. Chen, Y. Huang, and I. Cohen. “Pearson correlation coefficient.” In: *Noise reduction in speech processing*. Springer, 2009, pp. 1–4 (cit. on p. 49).

[11] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: A review and new perspectives.” In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828 (cit. on p. 11).

[12] A. Bibal, M. Lognoul, A. De Streel, and B. Frénay. “Legal requirements on explainability in machine learning.” In: *Artificial Intelligence and Law* 29.2 (2021), pp. 149–169 (cit. on pp. 3, 12–14).

[13] T. M. Breuel. “The OCRopus open source OCR system.” In: *Proc.SPIE* 6815 (2008), pp. 6815 - 6815 –15. DOI: 10.1117/12.783598 (cit. on p. 3).

[14] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. "The secret sharer: Evaluating and testing unintended memorization in neural networks." In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 267–284 (cit. on p. 26).

[15] J. Castro, D. Gómez, and J. Tejada. "Polynomial calculation of the Shapley value based on sampling." In: *Computers & Operations Research* 36.5 (2009), pp. 1726–1730 (cit. on p. 21).

[16] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. "Differentially private empirical risk minimization." In: *Journal of Machine Learning Research* 12.3 (2011) (cit. on p. 28).

[17] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. "This looks like that: deep learning for interpretable image recognition." In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. 24, 147, 148, 152, 154, 155).

[18] D. Chen, T. Orekondy, and M. Fritz. "Gs-wgan: A gradient-sanitized approach for learning differentially private generators." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12673–12684 (cit. on pp. 29, 34, 192, 193, 195, 196, 198–202).

[19] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke. "The rise of deep learning in drug discovery." In: *Drug discovery today* 23.6 (2018), pp. 1241–1250 (cit. on p. 12).

[20] M. Coavoux, S. Narayan, and S. B. Cohen. "Privacy-preserving Neural Representations of Text." In: *arXiv e-prints* (2018), arXiv–1808 (cit. on p. 25).

[21] P. Cortez, M. Rio, M. Rocha, and P. Sousa. "Multi-scale Internet traffic forecasting using neural networks and time series methods." In: *Expert Systems* 29.2 (2012), pp. 143–155 (cit. on p. 78).

[22] J. Crabbé and M. Van Der Schaar. "Explaining time series predictions with dynamic masks." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 2166–2177 (cit. on pp. 23, 44).

[23] J. Cristian Borges Gamboa. "Deep Learning for Time-Series Analysis." In: *arXiv e-prints* (2017), arXiv–1701 (cit. on p. 3).

[24] G. Dahl, M. Ranzato, A.-r. Mohamed, and G. E. Hinton. "Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine." In: *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., 2010, pp. 469–477. URL: `http://papers.nips.cc/paper/4169-phone-recognition-with-the-mean-covariance-restricted-boltzmann-machine.pdf` (cit. on p. 3).

[25] A. Das and P. Rad. "Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey." In: *arXiv e-prints* (2020), arXiv–2006 (cit. on pp. 19, 20).

[26] K. Das and R. N. Behera. "A survey on machine learning: concept, algorithms and applications." In: *International Journal of Innovative Research in Computer and Communication Engineering* 5.2 (2017), pp. 1301–1309 (cit. on p. 11).

[27] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. “The UCR time series archive.” In: *IEEE/CAA Journal of Automatica Sinica* 6.6 (2019), pp. 1293–1305 (cit. on p. 3).

[28] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, et al. “Large scale distributed deep networks.” In: *Advances in neural information processing systems* 25 (2012) (cit. on p. 12).

[29] M. Denil, B. Shakibi, L. Dinh, N. De Freitas, et al. “Predicting parameters in deep learning.” In: *Advances in neural information processing systems*. 2013, pp. 2148–2156 (cit. on p. 83).

[30] X. Ding, L. Zhang, Z. Wan, and M. Gu. “A brief survey on de-anonymization attacks in online social networks.” In: *2010 international conference on computational aspects of social networks*. IEEE. 2010, pp. 611–615 (cit. on p. 27).

[31] F. K. Došilović, M. Brčić, and N. Hlupić. “Explainable artificial intelligence: A survey.” In: *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE. 2018, pp. 0210–0215 (cit. on p. 13).

[32] P. Esling and C. Agon. “Time-series data mining.” In: *ACM Computing Surveys (CSUR)* 45.1 (2012), pp. 1–34 (cit. on p. 3).

[33] L. M. Fagan, E. H. Shortliffe, and B. G. Buchanan. “Computer-based medical decision making: from MYCIN to VM.” In: *Automedica* 3.2 (1980), pp. 97–108 (cit. on p. 13).

[34] H. Fang and Q. Qian. “Privacy preserving machine learning with homomorphic encryption and federated learning.” In: *Future Internet* 13.4 (2021), p. 94 (cit. on p. 16).

[35] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean. “Inceptiontime: Finding alexnet for time series classification.” In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962 (cit. on pp. 62, 63, 176, 191).

[36] V. Feng. “An overview of resnet and its variants.” In: *Towards data science* (2017) (cit. on p. 176).

[37] L. Feremans, V. Vercruyssen, B. Cule, W. Meert, and B. Goethals. “Pattern-based anomaly detection in mixed-type time series.” In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 240–256 (cit. on p. 140).

[38] A. Fisher, C. Rudin, and F. Dominici. “All Models are Wrong, but Many are Useful: Learning a Variable’s Importance by Studying an Entire Class of Prediction Models Simultaneously.” In: *J. Mach. Learn. Res.* 20.177 (2019), pp. 1–81 (cit. on pp. 44, 68–70, 72).

[39] R. Fong, M. Patrick, and A. Vedaldi. “Understanding deep networks via extremal perturbations and smooth masks.” In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 2950–2958 (cit. on pp. 23, 102).

[40] R. C. Fong and A. Vedaldi. “Interpretable explanations of black boxes by meaningful perturbation.” In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3429–3437 (cit. on p. 66).

[41] M. Fredrikson, S. Jha, and T. Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures.” In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322–1333 (cit. on pp. 15, 26).

[42] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. “Privacy in pharmacogenetics: An End-to-End case study of personalized warfarin dosing.” In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 17–32 (cit. on p. 26).

[43] T.-c. Fu. “A review on time series data mining.” In: *Engineering Applications of Artificial Intelligence* 24.1 (2011), pp. 164–181 (cit. on p. 3).

[44] K. Fukuchi, Q. K. Tran, and J. Sakuma. “Differentially private empirical risk minimization with input perturbation.” In: *International Conference on Discovery Science*. Springer. 2017, pp. 82–90 (cit. on p. 28).

[45] A. H. Gee, D. Garcia-Olano, J. Ghosh, and D. Paydarfar. “Explaining deep classification of time-series data with learned prototypes.” In: *CEUR workshop proceedings*. Vol. 2429. NIH Public Access. 2019, p. 15 (cit. on pp. 24, 147, 148, 154, 155).

[46] D. Gentner and J. Colhoun. “Analogical processes in human thinking and learning.” In: *Towards a theory of thinking*. Springer, 2010, pp. 35–48 (cit. on p. 147).

[47] S. Ghosh and P. O. Kristensson. “Neural Networks for Text Correction and Completion in Keyboard Decoding.” In: *arXiv e-prints* (2017), arXiv–1709 (cit. on p. 12).

[48] R. Girshick. “Fast r-cnn.” In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448 (cit. on p. 3).

[49] S. P. Gochhayat, C. Lal, L. Sharma, D. Sharma, D. Gupta, J. A. M. Saucedo, and U. Kose. “Reliable and secure data transfer in IoT networks.” In: *Wireless Networks* 26.8 (2020), pp. 5689–5702 (cit. on p. 13).

[50] S. González-Carvajal and E. C. Garrido-Merchán. “Comparing BERT against traditional machine learning text classification.” In: *arXiv e-prints* (2020), arXiv–2005 (cit. on p. 12).

[51] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial nets.” In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 29).

[52] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and Harnessing Adversarial Examples.” In: *arXiv e-prints* (2014), arXiv–1412 (cit. on pp. 77, 87, 91).

[53] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang. “Towards efficient and privacy-preserving federated deep learning.” In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6 (cit. on pp. 16, 161).

[54] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 194–196).

[55] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, J. E. Davis, and J. H. Saltz. "Patch-based convolutional neural network for whole slide tissue image classification." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2424–2433 (cit. on p. 25).

[56] T. Huber, B. Limmer, and E. André. "Benchmarking Perturbation-based Saliency Maps for Explaining Atari Agents." In: *arXiv e-prints* (2021), arXiv–2101 (cit. on p. 40).

[57] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. "Adversarial examples are not bugs, they are features." In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 185).

[58] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren. "Secure, privacy-preserving and federated machine learning in medical imaging." In: *Nature Machine Intelligence* 2.6 (2020), pp. 305–311 (cit. on p. 4).

[59] H. Kannan, A. Kurakin, and I. Goodfellow. "Adversarial Logit Pairing." In: *arXiv e-prints* (2018), arXiv–1803 (cit. on p. 87).

[60] M. Karliuk. "Ethical and Legal Issues in Artificial Intelligence." In: *International and Social Impacts of Artificial Intelligence Technologies, Working Paper* 44 (2018) (cit. on p. 3).

[61] A. Karpathy and L. Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3128–3137 (cit. on p. 3).

[62] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)." In: *International conference on machine learning*. PMLR. 2018, pp. 2668–2677 (cit. on p. 135).

[63] J. Kim, A. Rohrbach, T. Darrell, J. Canny, and Z. Akata. "Textual explanations for self-driving vehicles." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 563–578 (cit. on p. 80).

[64] W. Knight. *MIT Technology Review: The Financial World Wants to Open AI's Black Boxes*. 2017. URL: `https://www.technologyreview.com/s/604122/the-financial-world-wants-to-open-ais-black-boxes/` (cit. on p. 3).

[65] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten. "CrypTen: Secure Multi-Party Computation Meets Machine Learning." In: *Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning*. 2020 (cit. on p. 170).

[66] P. W. Koh and P. Liang. "Understanding black-box predictions via influence functions." In: *International conference on machine learning*. PMLR. 2017, pp. 1885–1894 (cit. on pp. 23, 113–115, 119–121, 125, 126).

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (cit. on pp. 3, 40, 161).

[68] D. Kumar, G. W. Taylor, and A. Wong. “Opening the Black Box of Financial AI with CLEAR-Trade: A CLass-Enhanced Attentive Response Approach for Explaining and Visualizing Deep Learning-Driven Stock Market Prediction.” In: *ArXiv e-prints* (Sept. 2017). arXiv: 1709.01574 [cs.AI] (cit. on pp. 80, 103).

[69] A. Kurakin, I. J. Goodfellow, and S. Bengio. “Adversarial examples in the physical world.” In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112 (cit. on pp. 77, 87).

[70] A. Labrinidis and H. V. Jagadish. “Challenges and opportunities with big data.” In: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 2032–2033 (cit. on p. 11).

[71] M. Längkvist, L. Karlsson, and A. Loutfi. “A review of unsupervised feature learning and deep learning for time-series modeling.” In: *Pattern Recognition Letters* 42 (2014), pp. 11–24 (cit. on p. 3).

[72] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Backpropagation applied to handwritten zip code recognition.” In: *Neural computation* 1.4 (1989), pp. 541–551 (cit. on p. 165).

[73] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. “Visualizing the loss landscape of neural nets.” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 105).

[74] O. Li, H. Liu, C. Chen, and C. Rudin. “Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018 (cit. on pp. 14, 24).

[75] X. Li and J. Lin. “Linear time complexity time series classification with bag-of-pattern-features.” In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2017, pp. 277–286 (cit. on p. 140).

[76] Y. Li and T. Yang. “Word embedding for understanding natural language: a survey.” In: *Guide to big data applications*. Springer, 2018, pp. 83–104 (cit. on p. 17).

[77] Z. W. Lim, M. L. Lee, W. Hsu, and T. Y. Wong. “Building trust in deep learning system towards automated disease detection.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 9516–9521 (cit. on p. 12).

[78] Z. C. Lipton. “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.” In: *Queue* 16.3 (2018), pp. 31–57 (cit. on p. 19).

[79] X. Liu, L. Xie, Y. Wang, J. Zou, J. Xiong, Z. Ying, and A. V. Vasilakos. "Privacy and security issues in deep learning: A survey." In: *IEEE Access* 9 (2020), pp. 4566–4593 (cit. on pp. 4, 15, 25).

[80] S. M. Lundberg and S.-I. Lee. "A unified approach to interpreting model predictions." In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 22, 44, 145).

[81] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. "Communication-efficient learning of deep networks from decentralized data." In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282 (cit. on p. 28).

[82] D. A. Melis and T. Jaakkola. "Towards robust interpretability with self-explaining neural networks." In: *Advances in Neural Information Processing Systems*. 2018, pp. 7775–7784 (cit. on pp. 3, 90, 91).

[83] D. Mercier, A. Dengel, and S. Ahmed. "P2exnet: Patch-based prototype explanation network." In: *International Conference on Neural Information Processing*. Springer. 2020, pp. 318–330 (cit. on pp. 147, 218).

[84] D. Mercier, A. Dengel, and S. Ahmed. "PatchX: Explaining Deep Models by Intelligible Pattern Patches for Time-series Classification." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8 (cit. on pp. 133, 218).

[85] D. Mercier, A. Dengel, and S. Ahmed. "TimeREISE: Time Series Randomized Evolving Input Sample Explanation." In: *Sensors* 22.11 (2022). DOI: 10.3390/s22114084 (cit. on pp. 57, 218).

[86] D. Mercier, A. Dengel, S. Ahmed, et al. "From Private to Public: Benchmarking GANs in the Context of Private Time Series Classification." In: *arXiv preprint arXiv:2303.15916v2* (2023) (cit. on p. 191).

[87] D. Mercier, A. Lucieri, M. Munir, A. Dengel, and S. Ahmed. "PPML-TSA: A modular privacy-preserving time series classification framework." In: *Software Impacts* (2022), p. 100286 (cit. on p. 161).

[88] D. Mercier, S. Saifullah, A. Lucieri, A. Dengel, and S. Ahmed. "Privacy Meets Explainability: A Comprehensive Impact Benchmark." In: *arXiv preprint arXiv:2211.04110* (2022) (cit. on p. 175).

[89] D. Mercier, S. A. Siddiqui, A. Dengel, and S. Ahmed. "Interpreting deep models through the lens of data." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8 (cit. on p. 113).

[90] D. Mercier, S. A. Siddiqui, A. Dengel, and S. Ahmed. "TSInsight: A Local-Global Attribution Framework for Interpretability in Time Series Data." In: *Sensors* 21.21 (2021), p. 7373 (cit. on pp. 97, 218).

[91] D. Mercier, S. A. Siddiqui, M. Munir, A. Dengel, and S. Ahmed. "Tsviz: Demystification of deep learning models for time-series analysis." In: *IEEE Access* 7 (2019), pp. 67027–67040 (cit. on pp. 39, 40, 61, 62, 77, 101, 102, 113, 114, 136, 150, 175, 176, 218).

[92] D. Mercier., J. Bhatt., A. Dengel., and S. Ahmed. "Time to Focus: A Comprehensive Benchmark using Time Series Attribution Methods." In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, INSTICC. SciTePress, 2022, pp. 562–573. DOI: 10.5220/0010904400003116 (cit. on p. 39).

[93] T. Miller. "Explanation in artificial intelligence: Insights from the social sciences." In: *Artificial intelligence* 267 (2019), pp. 1–38 (cit. on p. 176).

[94] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh. "Privacy in Deep Learning: A Survey." In: *arXiv e-prints* (2020), arXiv–2004 (cit. on pp. 15, 25–27).

[95] I. Mironov, K. Talwar, and L. Zhang. "Rényi Differential Privacy of the Sampled Gaussian Mechanism." In: *arXiv e-prints* (2019), arXiv–1908 (cit. on p. 167).

[96] R. Mitchell, J. Cooper, E. Frank, and G. Holmes. "Sampling permutations for Shapley value estimation." In: (2022) (cit. on p. 44).

[97] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. "Pruning Convolutional Neural Networks for Resource Efficient Inference." In: *arXiv e-prints* (2016), arXiv–1611 (cit. on p. 90).

[98] L. Myers and M. J. Sirois. "Spearman correlation coefficients, differences between." In: *Encyclopedia of statistical sciences* 12 (2004) (cit. on p. 49).

[99] R. R. Nadikattu. "The emerging role of artificial intelligence in modern society." In: *International Journal of Creative Research Thoughts* (2016) (cit. on p. 11).

[100] M. Naehrig, K. Lauter, and V. Vaikuntanathan. "Can homomorphic encryption be practical?" In: *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 2011, pp. 113–124 (cit. on p. 28).

[101] I. E. Nielsen, D. Dera, G. Rasool, R. P. Ramachandran, and N. C. Bouaynaya. "Robust Explainability: A tutorial on gradient-based attribution methods for deep neural networks." In: *IEEE Signal Processing Magazine* 39.4 (2022), pp. 73–84 (cit. on pp. 16, 40).

[102] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu. "Using of Jaccard coefficient for keywords similarity." In: *Proceedings of the international multiconference of engineers and computer scientists*. Vol. 1. 6. 2013, pp. 380–384 (cit. on p. 49).

[103] S. J. Oh, B. Schiele, and M. Fritz. "Towards reverse-engineering black-box neural networks." In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 121–144 (cit. on p. 25).

[104] F. Paas, J. E. Tuovinen, H. Tabbers, and P. W. Van Gerven. "Cognitive load measurement as a means to advance cognitive load theory." In: *Educational psychologist* 38.1 (2003), pp. 63–71 (cit. on p. 142).

[105] S. Palacio, J. Folz, J. Hees, F. Raue, D. Borth, and A. Dengel. "What do deep networks like to see?" In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3108–3117 (cit. on pp. 23, 99, 100, 103, 107).

[106] S. Palacio, A. Lucieri, M. Munir, S. Ahmed, J. Hees, and A. Dengel. "Xai handbook: Towards a unified framework for explainable ai." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3766–3775 (cit. on p. 15).

[107] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data." In: *arXiv e-prints* (2016), arXiv–1610 (cit. on p. 28).

[108] M. Perc, M. Ozer, and J. Hojnik. "Social and juristic challenges of artificial intelligence." In: *Palgrave Communications* 5.1 (2019), pp. 1–7 (cit. on p. 3).

[109] R. S. Peres, X. Jia, J. Lee, K. Sun, A. W. Colombo, and J. Barata. "Industrial artificial intelligence in industry 4.0-systematic review, challenges and outlook." In: *IEEE Access* 8 (2020), pp. 220121–220139 (cit. on p. 3).

[110] V. Petsiuk, A. Das, and K. Saenko. "RISE: Randomized Input Sampling for Explanation of Black-box Models." In: *arXiv e-prints* (2018), arXiv–1806 (cit. on pp. 23, 57, 59, 60).

[111] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. "Variational autoencoder for deep learning of images, labels and captions." In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 29).

[112] T. E. Raghunathan et al. "What do we do with missing data? Some options for analysis of incomplete data." In: *Annual review of public health* 25.1 (2004), pp. 99–117 (cit. on p. 12).

[113] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang. "Membership Inference Attack against Differentially Private Deep Learning Model." In: *Trans. Data Priv.* 11.1 (2018), pp. 61–79 (cit. on p. 15).

[114] G. D. P. Regulation. "Regulation EU 2016/679 of the European Parliament and of the Council of 27 April 2016." In: *Official Journal of the European Union. Available at: http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf (accessed 20th Sep. 2017)* (2016) (cit. on pp. 4, 13, 15, 31).

[115] M. T. Ribeiro, S. Singh, and C. Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier." In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144 (cit. on pp. 17, 21, 22, 44, 65, 68–70, 72, 100, 145).

[116] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. "Contractive auto-encoders: Explicit invariance during feature extraction." In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress. 2011, pp. 833–840 (cit. on p. 107).

[117] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez. "Explainable Artificial Intelligence (XAI) on TimeSeries Data: A Survey." In: *arXiv e-prints* (2021), arXiv–2104 (cit. on p. 19).

[118] M. Al-Rubaie and J. M. Chang. "Privacy-preserving machine learning: Threats and solutions." In: *IEEE Security & Privacy* 17.2 (2019), pp. 49–58 (cit. on p. 4).

[119] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors." In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. URL: http://dx.doi.org/10.1038/323533a0 (cit. on p. 78).

[120] M. Sabt, M. Achemlal, and A. Bouabdallah. "Trusted execution environment: what it is, and what it is not." In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. IEEE. 2015, pp. 57–64 (cit. on p. 29).

[121] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. "Improved techniques for training gans." In: *Advances in neural information processing systems* 29 (2016) (cit. on pp. 194–196).

[122] R. Saluja, A. Malhi, S. Knapič, K. Främling, and C. Cavdar. "Towards a Rigorous Evaluation of Explainability for Multivariate Time Series." In: *arXiv e-prints* (2021), arXiv–2104 (cit. on p. 4).

[123] S. Salvador and P. Chan. "FastDTW: Toward accurate dynamic time warping in linear time and space." In: *Intelligent Data Analysis* 11.5 (2007), pp. 561–580 (cit. on p. 83).

[124] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller. "Evaluating the visualization of what a deep neural network has learned." In: *IEEE transactions on neural networks and learning systems* 28.11 (2016), pp. 2660–2673 (cit. on p. 177).

[125] W. Samek, T. Wiegand, and K.-R. Müller. "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models." In: *arXiv e-prints* (2017), arXiv–1708 (cit. on p. 3).

[126] M. Schermann, H. Hemsen, C. Buchmüller, T. Bitter, H. Krcmar, V. Markl, and T. Hoeren. "Big data." In: *Wirtschaftsinformatik* 56.5 (2014), pp. 281–287 (cit. on p. 11).

[127] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. "Towards a rigorous evaluation of xai methods on time series." In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 4197–4201 (cit. on p. 13).

[128] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. Van Laerhoven. "Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection." In: *Proceedings of the 20th ACM International Conference on Multimodal Interaction*. ICMI '18. Boulder, CO, USA: ACM, 2018, pp. 400–408. DOI: 10.1145/3242969.3242985 (cit. on p. 102).

[129] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626 (cit. on p. 103).

[130] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. "Membership inference attacks against machine learning models." In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18 (cit. on p. 25).

[131] A. Shrikumar, P. Greenside, and A. Kundaje. "Learning important features through propagating activation differences." In: *International conference on machine learning*. PMLR. 2017, pp. 3145–3153 (cit. on pp. 20, 22, 44).

[132] D. F. Silva, R. Giusti, E. Keogh, and G. E. Batista. "Speeding up similarity search under dynamic time warping by pruning unpromising alignments." In: *Data Mining and Knowledge Discovery* 32.4 (2018), pp. 988–1016 (cit. on p. 3).

[133] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." In: *arXiv e-prints* (2013), arXiv–1312 (cit. on pp. 20, 44).

[134] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. "Striving for Simplicity: The All Convolutional Net." In: *ICLR (workshop track)*. 2015 (cit. on pp. 20, 44, 65, 68–70, 72, 103).

[135] T. Sudkamp and R. J. Hammell. "Interpolation, completion, and learning fuzzy rules." In: *IEEE Transactions on Systems, Man, and Cybernetics* 24.2 (1994), pp. 332–342 (cit. on p. 12).

[136] M. Sundararajan, A. Taly, and Q. Yan. "Axiomatic attribution for deep networks." In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328 (cit. on pp. 21, 44, 65, 68–70, 72, 103).

[137] L. Sweeney. "k-anonymity: A model for protecting privacy." In: *International journal of uncertainty, fuzziness and knowledge-based systems* 10.05 (2002), pp. 557–570 (cit. on p. 27).

[138] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *Thirty-first AAAI conference on artificial intelligence*. 2017 (cit. on p. 62).

[139] S. C. Tan, K. M. Ting, and T. F. Liu. "Fast anomaly detection for streaming data." In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. 2011, p. 1511 (cit. on p. 102).

[140] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim. "Privacy-Preserving Deep Learning on Machine Learning as a Service—a Comprehensive Survey." In: *IEEE Access* 8 (2020), pp. 167425–167447 (cit. on p. 4).

[141] H. C. Tanuwidjaja, R. Choi, and K. Kim. "A survey on deep learning techniques for privacy-preserving." In: *International Conference on Machine Learning for Cyber Security*. Springer. 2019, pp. 29–46 (cit. on p. 25).

[142] L. Theis, I. Korshunova, A. Tejani, and F. Huszár. "Faster gaze prediction with dense networks and Fisher pruning." In: *arXiv e-prints* (2018), arXiv–1801 (cit. on p. 90).

[143] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. "Wasserstein Auto-Encoders." In: *arXiv e-prints* (2017), arXiv–1711 (cit. on p. 29).

[144] R. Tomsett, D. Harborne, S. Chakraborty, P. Gurram, and A. Preece. "Sanity checks for saliency metrics." In: 34.04 (2020), pp. 6021–6029 (cit. on p. 4).

[145] T. Vermeire, T. Laugel, X. Renard, D. Martens, and M. Detyniecki. "How to choose an Explainability Method? Towards a Methodical Implementation of XAI in Practice." In: *arXiv e-prints* (2021), arXiv–2107 (cit. on p. 54).

[146] M. M.-C. Vidovic, N. Görnitz, K.-R. Müller, and M. Kloft. "Feature Importance Measure for Non-linear Learning Algorithms." In: *arXiv e-prints* (2016), arXiv–1611 (cit. on p. 101).

[147] S. Wagh, D. Gupta, and N. Chandran. "SecureNN: Efficient and private neural network training." In: *Cryptology ePrint Archive* (2018) (cit. on pp. 28, 29).

[148] P. Wang, E. Fan, and P. Wang. "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning." In: *Pattern Recognition Letters* 141 (2021), pp. 61–67 (cit. on p. 12).

[149] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck. "Evaluating explanation methods for deep learning in security." In: *2020 IEEE european symposium on security and privacy (EuroS&P)*. IEEE. 2020, pp. 158–174 (cit. on p. 14).

[150] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." In: *arXiv e-prints* (2016), arXiv–1609 (cit. on p. 3).

[151] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou. "Differentially Private Generative Adversarial Network." In: *arXiv e-prints* (2018), arXiv–1802 (cit. on pp. 29, 192, 193, 198, 199).

[152] M. Yan, C. W. Fletcher, and J. Torrellas. "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures." In: *29th USENIX Security Symposium (USENIX Security 20)*. 2020, pp. 2003–2020 (cit. on p. 26).

[153] Q. Yang, Y. Liu, T. Chen, and Y. Tong. "Federated machine learning: Concept and applications." In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19 (cit. on p. 28).

[154] Q. Yang and X. Wu. "10 challenging problems in data mining research." In: *International Journal of Information Technology & Decision Making* 5.04 (2006), pp. 597–604 (cit. on p. 3).

[155] C.-K. Yeh, C.-Y. Hsieh, A. Suggala, D. I. Inouye, and P. K. Ravikumar. "On the (in) fidelity and sensitivity of explanations." In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on pp. 14, 39, 45, 69, 177).

[156] C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, T. Pfister, and P. Ravikumar. "On completeness-aware concept-based explanations in deep neural networks." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20554–20565 (cit. on pp. 14, 16).

[157] C.-K. Yeh, J. Kim, I. E.-H. Yen, and P. K. Ravikumar. "Representer point selection for explaining deep neural networks." In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. 23, 113–115, 119–121).

[158] X. Yi, R. Paulet, and E. Bertino. “Homomorphic encryption.” In: *Homomorphic encryption and applications*. Springer, 2014, pp. 27–46 (cit. on p. 28).

[159] M. D. Zeiler and R. Fergus. “Visualizing and understanding convolutional networks.” In: *European conference on computer vision*. Springer. 2014, pp. 818–833 (cit. on pp. 22, 44, 60, 65, 68–70, 72).

[160] Q.-s. Zhang and S.-C. Zhu. “Visual interpretability for deep learning: a survey.” In: *Frontiers of Information Technology & Electronic Engineering* 19.1 (2018), pp. 27–39 (cit. on p. 4).

[161] M. Zheng, D. Xu, L. Jiang, C. Gu, R. Tan, and P. Cheng. “Challenges of privacy-preserving machine learning in IoT.” In: *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*. 2019, pp. 1–7 (cit. on p. 25).

[162] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger. “Evaluating the quality of machine learning explanations: A survey on methods and metrics.” In: *Electronics* 10.5 (2021), p. 593 (cit. on p. 176).

[163] T. Zhu, D. Ye, W. Wang, W. Zhou, and P. Yu. “More than privacy: Applying differential privacy in key areas of artificial intelligence.” In: *IEEE Transactions on Knowledge and Data Engineering* (2020) (cit. on p. 15).

[164] Y. Zhu and Y.-X. Wang. “Poission subsampled renyi differential privacy.” In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7634–7642 (cit. on p. 28).

[165] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. “Visualizing Deep Neural Network Decisions: Prediction Difference Analysis.” In: *arXiv e-prints* (2017), arXiv–1702 (cit. on p. 80).

INDEX

ACADEMIC CURRICULUM VITÆ: DOMINIQUE MERCIER

PERSONAL INFORMATION

Affiliation:	German Research Center for Artificial Intelligence
Department:	Smart Data & Knowledge Systems
Phone:	+49 631 20575 3521
Email:	dominique.mercier@dfki.de
University Email:	mercier@rhrk.uni-kl.de
Google Scholar:	9CLJlYUAAAAJ
ORCID:	0000-0001-8817-2744
LinkedIn:	dominique-mercier-177227220

SKILLS & INTERESTS

Python

Tensorflow, Keras, Pytorch, Pandas, Numpy, Seaborn, Matplotlib, Sklearn

Web Development

HMTL, CSS, JavaScript

Unity Engine

Development of 3d and virtual reality environments including C++ scripts for interaction.

Further Computers Skills

Bash, Linux, SSH, Cluster computing

Language Skills

German and English

Interests

Time Series Analysis, Document Analysis, Natural Language Processing, Deep Learning, Convolutional Neural Networks, Transformer, Interpretability, Privacy

EDUCATION

RPTU Kaiserslautern-Landau

05/2018 - 09/2023

PhD Student / Researcher, Knowledge-based Systems Group, Prof. Dengel

Main areas: XAI, Privacy, Time Series, Deep Learning, CNNs, GANs

PhD Topic: *TimeFrame: A Novel Framework for Interpretable and Privacy-Preserving Deep Learning for Time Series Analysis*

TU Kaiserslautern

04/2016 - 05/2018

Master of Science in Computer Science

Master Thesis: *Towards Understanding Deep Networks for Time Series Analysis*

TU Kaiserslautern

10/2012 - 05/2016

Bachelor of Science in Computer Science

Bachelor Thesis: *Publicaiton Sentiment Analysis*

EXPERIENCE

since 06/2017	German Research Center for Artificial Intelligence (DFKI) Smart Data and Knowledge Services Department, Prof. Dengel Researcher
06/2017 - 05/2018	German Research Center for Artificial Intelligence (DFKI) Smart Data and Knowledge Services Department, Prof. Dengel Student Assistant

SELECTED PUBLICATIONS

Sensors 2022	TimeREISE: Time-series Randomized Evolving Input Sample Explanation D. Mercier, A. Dengel, S. Ahmed
ICAART 2022	Time to focus: A comprehensive benchmark using time series attribution methods D. Mercier, J. Bhatt, A. Dengel, S. Ahmed
IJCNN 2021	Patchx: Explaining deep models by intelligible pattern patches for time-series classification D. Mercier, A. Dengel, S. Ahmed
IEEE 2021	Evaluating privacy-preserving machine learning in critical infrastructures: A case study on time-series classification D. Mercier, A. Lucieri, M. Munir, A. Dengel, S. Ahmed
IEEE 2019	Tsviz: Demystification of deep learning models for time-series analysis S. A. Siddiqui, D. Mercier, A. Dengel, S. Ahmed