DTU Compute
Department of Applied Mathematics and Computer Science

Machine Learning and Time Series Analysis for Optimization and Anomaly Detection in Field Service

Tobias Engelhardt Rasmussen

Machine Learning and Time Series Analysis for Optimization and Anomaly Detection in Field Service

PhD Thesis
August 2024

By
Tobias Engelhardt Rasmussen

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk

Assessment Committee

Anders Nymark Christensen
Associate Professor, Technical University of Denmark
Kongens Lyngby, Denmark

Sune Darkner
Professor, University of Copenhagen
Copenhagen, Denmark

Amanda Lenzi
Lecturer, University of Edinburgh
Edinburgh, United Kingdom

Supervisors

Andreas Baum
Associate Professor, Technical University of Denmark
Kongens Lyngby, Denmark

Thomas Martini Jørgensen
Senior researcher, Technical University of Denmark
Kongens Lyngby, Denmark

Bjarne Kjær Ersbøll
Professor, Technical University of Denmark
Kongens Lyngby, Denmark

Summary

Hybrid Fiber-Coaxial (HFC) networks are one of the most popular infrastructures used to provide cabled internet connection to paying customers. Despite the technology having been around for many years, continuing developments and the low cost of deployment means that it is believed to be an important part of digital infrastructure for many years to come. This significant role, however, comes with challenges, as HFC networks are complex and susceptible to a long range of failures that can happen in different locations in the network. This requires Internet Service Providers (ISPs) to have a large fleet of technicians maintaining the network during daily operation. This is costly in terms of both financial and natural resources, making the optimization of fault detection and localization highly valuable.

Currently, network monitoring is characterized by manual operations, again highlighting the need for automatic and remote approaches. Additionally, many ISPs do not have access to a fully known and updated network topology (the path of connection from a customer modem to the terminal) which is important for accurate troubleshooting and optimal route planning. This makes an algorithm that can reconstruct these topologies using remotely gathered customer data highly valuable.

In this work, we propose two different approaches for remotely detecting anomalies or failures in HFC networks, along with one approach to reconstructing the missing topology. Each of the methods will be using time series data from the network of TDC NET, which is the biggest provider of digital infrastructure in Denmark. One of the biggest challenges with this data is the absence of an accurate ground truth. Our two anomaly detection approaches tackle this problem

in two different ways. One approach utilizes the knowledge of a single and well-known type of error known as Common Path Distortion (CPD). In cooperation with domain experts, we develop a manually labeled dataset where the labels correspond to the presence, respectively, absence of CPD. We evaluate multiple supervised Machine Learning (ML) methods toward their ability to model the dataset and show promising results. We analyze the parameter importance and use that to propose a simple method achieving similar performance while being easier to interpret and implement.

A different approach makes use of a Normalizing Flow (NF), which is a recent generative model, to model the distribution of time series behavior in an unsupervised way. This enables us to detect anomalous behavior in low-density regions of the distribution. Our framework is based on using the NF to estimate the density of a latent representation of the time series computed using an AutoEncoder (AE). While showing promising results, we additionally propose an algorithm for excluding abnormal points from the learned distribution and thus learning only the systematic behavior. We show that the abnormal points are not only pushed out of the distribution but also arrange themselves in clusters that can subsequently be used to identify potential underlying root causes.

Lastly, we evaluate the feasibility of accurately reconstructing the missing topologies using time series data from multiple customer modems. Specifically, we train an encoder that can extract relevant events in time series that are informative with respect to which modems are correlated and which are not. We base our method on an old method from biology used to infer phylogenetic trees from gene sequences. We make multiple contributions to this problem including an updated version of the problem of reconstructing these trees to make it applicable in our case, along with an algorithm for finding the optimal solution. We further contribute by demonstrating the feasibility of embedding this optimization problem directly into a deep learning loss function to learn the informative events for said algorithm and thereby reconstruct the missing topologies.

Resumé (Danish)

De såkaldte Hybrid-Fiber Coaxial (HFC) netværk (også kaldet bredbåndsnetværk), som består af både fiber og coax-teknologi, har længe været en af de mest udbredte metoder brugt til at levere kablede internetforbindelser til betalende kunder. Selvom teknologien efterhånden har mange år på bagen, gør den evige udvikling og den lave anlægningspris, at teknologien højst sandsynligt vil være en vigtig del af den digitale infrastruktur i mange år endnu. Dette betyder til gengæld også, at der bliver ved med at opstå problemer i de komplekse netværk, som er sårbare overfor en lang række fejl, der kan forekomme mange forskellige steder i netværket. Det betyder, at internetudbydere er nødsagede til at have en flåde af teknikere, som kan vedligeholde netværket i dets daglige drift. Det er dyrt både i finansielle, men også i naturlige ressourcer, hvilket betyder, at optimering af fejlfinding og fejllokalisering er et vigtigt forskningsområde.

Den dag i dag, er overvågning af netværket karakteriseret ved manuelt arbejde, som igen tydeliggør behovet for automatiske, fjernstyrede metoder. Derudover har mange internetudbydere ofte ikke adgang til en fuldt ud kendt og opdateret netværkstopologi (altså den måde hvorpå kunderne er forbundet med terminalen igennem kabler), som er vigtig for nøjagtig fejllokalisering og optimal ruteplanlægning. Dette betyder, at en algoritme, der kan rekonstruere de her topologier kun ved hjælp af tidsrækkedata fra kunderne, vil have stor værdi.

I denne afhandling præsenterer vi to forskellige metoder til fejlfinding i bredbåndsnetværk og én metode til at rekonstruere de manglende topologier ved hjælp af tidsrækkedata fra kundemodemmer, forbundet til TDC NETs netværk, som er det mest omfattende i Danmark. En af de største udfordringer med denne type data er, at der ikke eksisterer nogen nøjagtig reference for, hvornår

der er fejl i netværket. Vores to fejlfindingalgoritmer takler dette problem på to forskellige måder. I den ene metode udnytter vi, at der findes en type fejl, som er almindelig i bredbåndsnetværk og derfor allerede er kendt og forstået, også kaldet Common Path Distortion (CPD). I samarbejde med domæneeksperter har vi fået udarbejdet et dataset inklusiv en reference for, hvornår CPD forekommer. Vi evaluerer flere forskellige metoder indenfor MaskinLæring (ML) ved at undersøge, hvor gode de hver især er til at detektere CPD - og det med gode resultater. Vi analyserer, hvilke parametre der er vigtige, og bruger disse til at foreslå en simpel metode, som klarer sig næsten lige så godt, men som er nemmere at fortolke og implementere.

En anden metode gør brug af et såkaldt Normalizing Flow (NF) - en nylig type model indenfor generativ kunstig intelligens - til at modellere fordelingen af tidsrækkernes adfærd uden på forhånd at vide, hvor der forekommer fejl. Dette gør det muligt at detektere fejl eller anomalier i områder med lav densitet i denne fordeling. Vores overordnede metode bruger et NF til at modellere fordelingen af en latent repræsentation af tidsrækkerne, som vi først laver ved hjælp af en såkaldt AutoEncoder (AE). Udover denne metode, som viser lovende resultater, præsenterer vi også en ny algoritme, som kan ekskludere de fejlagtige observationer, hvilket gør det muligt at modellere udelukkende den systematiske variation i datasættet. Vi viser, at disse fejlagtige observationer bliver skubbet ud af fordelingen og arrangerer sig selv i klynger, som kan forbindes med faktiske underliggende fejl i netværket.

Sidst, men ikke mindst evaluerer vi, hvorvidt det er muligt nøjagtigt at rekonstruere de manglende topologier ved brug af kontinuær data fra flere kundemodemmer. Specifikt træner vi en model, som kan trække relevante hændelser ud af tidsrækkerne, som fortæller noget om, hvilke modemmer der korrelerer, og hvilke der ikke gør det. Vi baserer vores metode på en gammel metode fra biologien, som blev brugt til at udlede fylogenetiske træer fra gensekvenser. Vi laver flere bidrag til problemet, hvilket inkluderer en opdateret formulering af problemet med at udlede disse træer, hvilket gør det anvendeligt til vores formål - og følgeligt en algoritme til at finde den optimale løsning. Vi bidrager yderligere ved at demonstrere, at det er muligt at integrere det omtalte optimeringsproblem direkte ind i en objektfunktion som bruges til at lære et neuralt netværk. Den resulterende model gør det muligt for os at udtrække relevante hændelser fra tidsrækkerne, som så kan bruges til at rekonstruere de manglende topologier.

Preface

This thesis was prepared at DTU Compute in fulfilment of the requirements for acquiring a PhD at the Department of Applied Mathematics and Computer Science, DTU Compute, at the Technical University of Denmark, DTU.

The project was funded by the Innovation Fund Denmark (Innovationsfonden) as a part of project 0224-00055B, GREENFORCE, which is a collaboration between TDC NET, Qampo, and DTU Compute. The work was carried out from September 1, 2021, to August 31, 2024, and the project was supervised by Associate Professor Andreas Baum, Senior Researcher Thomas Martini Jørgensen, and Professor Bjarne Ersbøll.

The project included a three-month research stay at the Department of Computer Science at Duke University in Durham, North Carolina, USA, that was supervised by Professor Xiaowei Yang.

Before being presented with the novel work produced as a result of the three-year project, the reader will be equipped with relevant background information and theory in the first part of this thesis. The second part of the thesis will first outline the context and relevance of the research outcome before the research articles are presented. The following research articles have been produced during the project and included in this thesis:

[A] Healy, M, Rasmussen, T. E., Maddimsetty, T. K., Vedantam, V. K., Baum, A., Jørgensen, T. M. *SeePD: Detecting Common Path Distortion Faults in Broadband Networks.* Under review at the *Journal of Signal Processing Systems.*

[B] Rasmussen, T. E., Algán, F. E. C., Baum, A. *Anomaly Detection in Broadband Networks: Using Normalizing Flows for Multivariate Time Series.* Paper submitted to *Signal Processing.*

[C] Rasmussen, T. E., Sørensen, S., Pisinger D., Jørgensen, T. M., Baum, A., *Topology Reconstruction in Telecommunication Networks: Embedding Operations Research within Deep Learning.* Under review at *Computers and Operations Research.* Preprint: [148]

The following papers have also been prepared during the PhD project but have not been included in this thesis.

[I] Rasmussen, T. E., Sørensen, S. *Encoding Binary Events from Continuous Time Series in Rooted Trees using Contrastive Lerning.* Extended abstract presented as a poster at the *Northern Lights Deep Learning conference 2024 (NLDL 24).* Preprint: [147].

[II] Rasmussen, T. E., Clemmensen L.K.H., Baum, A. *Compressing CNN Kernels for Videos Using Tucker Decompositions: Towards Lightweight CNN Applications. Proceedings of the Northern Lights Deep Learning conference 2022 (NLDL 22)* [145].

Kgs. Lyngby, August 31, 2024

Tobias Engelhardt Rasmussen

Acknowledgements

As this project comes to an end, I realize how lucky I have been to gain this experience. I sincerely thank my principal supervisor, Associate Professor Andreas Baum, for recognizing my potential and believing that I could make an impact on this project. Your guidance has been invaluable, and this would not have been possible without you. My gratitude also extends to my co-supervisors, Senior Researcher Thomas Martini Jørgensen and Professor Bjarne Kjær Ersbøll. Thank you all—Andreas, Thomas, and Bjarne—for the invaluable discussions we've had along the way, for helping me, respecting me, and allowing me to mature as a researcher by showing genuine interest in even the most abstract concepts I threw at you.

I would like to thank the partners in the Greenforce project—DTU Compute, DTU Management, TDC NET, Qampo, and the Innovation Fund Denmark—-for making this project possible. It is a genuinely good project that has encouraged interdisciplinary cooperation and that I already believe has made a positive impact on the world. A special thank you to Clara Chini Nielsen, Ewa Magdalena Behr, Joaquín Ignacio Fürstenheim, Meadhbh Healy, and Siv Sørensen. I appreciated our regular meetings across departments catching up and encouraging each other. Thank you to Karolina Wilczynska, Mads Bossen Hansen, and Clara Gudmann for managing the project and keeping everything on track.

I am very grateful to all of my collaborators: Meadhbh Healy, Siv Sørensen, Facundo Esteban Castellá Algán, Professor David Pisinger, Tarun Kumar Maddimsetty, and Vamsi Krishna Vedantam, for the many fruitful discussions, for their hard work, and for always being constructive and pleasant to work with.

Thank you, Meadhbh, for being my partner and for supporting me as a friend through this sometimes challenging project. Go raibh maith agat. Thank you, Siv, for always being open-minded, the always good discussions, and the long days spent fighting annoying details. Thank you, David, for always lightening the mood while maintaining professionalism. Thank you, Facu, for being respectful to me as your supervisor and for your hard work—I am sure you will do great things. Thank you, Tarun and Vamsi, for always welcoming Meadhbh and me into your office and for the invaluable discussions.

Thank you to everyone in the Statistics and Data Analysis Section at DTU Compute. Thank you for all the chats in the hallways and by the coffee machine. A special thank you goes out to my good colleagues in room 210—it has been a real pleasure working beside you all, and I am certain that you will do great no matter where you wind up. Thank you, Clara, for encouraging me to cycle to DTU, and thank you, Nikolaj and Sara, for occasionally driving me home.

Thank you, Professor Xiaowei Yang, for being welcoming from the very beginning. Thank you for always being interested and constructive in our discussions. I also extend my gratitude to Otto Mønsteds Fond, Knud Højgaards Fond, and William Demant Fonden for making my stay at Duke University possible. Experiencing the research environment in the US was truly special.

Last but definitely not least, I would like to thank my family and my friends. Thank you for allowing me to momentarily forget about research and broadband networks. Thank you for your support and for always believing in me. A special thank you to the love of my life André Luis Radigonda. Muito Obrigado, meu amado. Thank you for your unconditional support, especially in times of frustration. Thank you for teaching me how to manage my emotions–a truly invaluable lesson that I will carry with me forever.

List of Abbreviations

Contents

Chapter 1

Introduction

The ability to send messages across long distances has been a critical part of human infrastructure ever since different civilizations engaged in conflict with each other. Early versions of telecommunication consisted of visual signals such as beacons arranged in chains that could be sequentially lit to signal the arrival of an enemy [50] or sound signals such as made by drums used in ancient African and Asian cultures [181]. Since then, telecommunication technology has evolved dramatically, and from the middle of the 18th century, it has consisted mainly of electrical signals sent through metal wires. At first, these electrical wires simply conducted a current that completed a circuit, initiating a visible process at the other end. From electrostatic displacement of small pieces of paper or electrolysis of water in very early applications to the electromagnetic movement of a needle in later applications marking the beginning of the well-known electrical telegraphy popularized in the 19th century [81]. Later, it was shown that Radio Frequency (RF) signals could be used to carry information. With the arrival of telephones, this capability was initially realized through a simple end-to-end connection. In this setup, the voice of a speaker at one end of the line was electronically transmitted and reproduced as sound at the receiving end. This technology was further extended to wireless communication, enabling the transmission of sound via a single radio wave. These innovations laid the groundwork for digital signals, allowing data, in the form of bits, to be encoded into the frequency, respectively, the amplitude of an electromagnetic wave. This technology has expanded communication capabilities, facilitating travel and interaction in deep space, even extending to objects on other planets. It has also allowed individuals to connect globally from the palm of their hand,

while recent innovations have further enhanced connectivity through a network of satellites, enabling truly global communication [175].

Modern electric telecommunication uses the so-called broadband technology to send information at very high speeds. In broadband technology, RF signals at many different frequencies at once are used to send data through a given medium. Media include coaxial cables, fiber-optic cables, or simply the air or empty space using electromagnetic waves (radio or satellite) [57]. Coaxial cables are the most widespread and accessible medium used to provide cabled high-speed internet access to paying customers [18]. This type of cable is usually only deployed in the last part of the network, where they are used to branch out to the customers from a local terminal. This terminal again connects to the backbone internet using fiber technology, granting this type of network its name; a Hybrid Fiber-Coaxial (HFC) network. The relatively low investment cost and continuing development reaching ever-higher speeds mean that HFC networks are not likely to be phased out in the near future, but will continue to be an important part of the digital infrastructure [55]. This means that continuous research in the maintenance and operation of coaxial networks is continuing to be valuable.

To this day, digital infrastructure is still considered critical and governments all over the world have special requirements for the continuing maintenance and surveillance of the network as failures could e.g. prevent the population from receiving news about threats from outside or natural disasters [185, 122, 99]. Additionally, the European Union identifies "Advanced Connectivity, Navigation, and Digital Technologies" as one of ten technology areas that are critical to the security of the European economy [37].

1.1 The Green Transition

The continuing development of human civilization has achieved remarkable advancements in technology, industry, and living standards. However, research shows that this rapid development has come at a significant environmental cost, based on unsustainable practices for instance the burning of fossil fuels for energy, transportation, and other industrial activities. Burning fossil fuels emits CO_2 which is one of the so-called greenhouse gases that has been shown to be a cause of the greenhouse effect leading to global warming and climate change posing a great threat to the health of the planet impacting both the human civilization and the natural world [85].

This great threat makes it increasingly important to make technological and

cultural advancements toward an economy that is sustainable in terms of natural resources. In 2019 the European Union presented the *European Green Deal* in which the goal is to be the first climate-neutral continent, for instance by decoupling the economy from resource consumption [54]. Conclusively, every advancement or action that can be taken to decrease greenhouse gas emissions is highly relevant in the pursuit of this ambitious goal.

1.2 Motivation

TDC NET is the biggest provider of digital infrastructure in Denmark and currently has broadband or fiber connections to more than 1.5 million addresses in Denmark [180]. The broadband network is used by hundreds of thousands of paying customers who are serviced by the TDC NET fleet of technicians. The size of this infrastructure requires one of Denmark's biggest fleets of around 850 technicians driving upwards of 80,000 kilometers every day, which is the equivalent of circling the Earth twice [17]. Because the technicians perform a wide range of tasks and it is difficult to predict which tasks a given technician will be presented with on a given day, they need to carry a large amount of equipment. This requires the technician fleet to consist of big vehicles making the driven kilometers even more costly in terms of money and emission of greenhouse gases. This project is part of the GREENFORCE project which has as its main goal to use smart decisions and Artificial Intelligence (AI) in an attempt to bring down these emissions. This thesis aims to assess the potential of using AI and Machine Learning (ML) to aid different aspects of the daily operation of the network of TDC NET.

In this thesis, the aim is to provide a thorough overview of the broadband setting and the problems that relate to them. In collaboration with domain experts at TDC NET, some potential research problems have been identified in which ML could be used to better the situation. These include:

Anomalies / Network breakdowns

- Errors in the broadband network are generally sporadic, not well studied, and hard to detect
- The business case of network owners makes it hard to justify repairing or improving the connection of customers who are not complaining
- Unreliable customer complaints make it hard to acquire an anomaly ground truth

Route optimization

- Deficient mapping of the network connections complicates the search for errors that are located along the network infrastructure
- Not being aware of likely future errors may cause unnecessary driving, as technicians need to return at a later stage rather than fixing the errors prematurely

In this thesis, the problems mentioned above will be investigated to explore the potential of using performance metrics gathered at regular intervals by the many devices making up the network, arranged as time series. An introduction will be given to the different time series metrics, what they describe, and how they are gathered. Various ML models will be used to analyze the time series data, including different types of deep Neural Networks (NNs).

The intuition behind NNs builds on the idea of the brain with a set of neurons, arranged in a network, which communicate by firing up and passing information between one another along the network edges. The goal is for the neurons to learn to fire up or activate when a specific pattern is observed and thereby, in unison, to learn to distinguish complicated patterns. It is customary to arrange the NNs in layers that can learn increasingly complex patterns. If multiple layers are used in an NN, it is referred to as a *deep* NN. Using deep NNs to model a given problem is denoted Deep Learning (DL).

DL has become very popular and is state-of-the-art in many fields such as natural language processing [87] and image processing [111] and has also shown great potential in time series analysis including prediction [23, 121], classification [182, 82], and change-point-detection [98, 42]. This popularity in ML research is believed to be due to the flexibility of deep NNs, where choices for the architecture and training objectives can be made in various ways, allowing for models that match the intuition of the problem at hand. Recently, one of the most popular branches of DL is that of probabilistic generative models that utilize stochasticity to allow for the generation of random artificial samples that are previously unseen [12]. In probabilistic models, random variables are used to model distributions of outputs or representations of the data instead of deterministic values. This allows for modeling uncertain or unobserved variables by quantifying their variability and most likely outcome. A model that encodes data into a distributional representation and can also decode the input data back from this distribution (i.e., reverse the process) is called a generative model. Such a model can be used to generate or sample new random instances that were not part of the input data.

To showcase the capabilities of probabilistic generative models, I include a poem written by one of the first commercially available large-language models, ChatGPT [150]. ChatGPT has been trained on vast amounts of data available on

the internet and utilizing an amount of computational power that is hard to grasp. AI assistants have been generally well-received and are already used for assistance in several tasks like writing drafts, checking for clarity or grammar [163], summarizing meetings [186], tasks that can make business operations more efficient [143], or functions that can help create more specialized teaching in education [119]. The introduction of large language models has also sparked the novel research area of prompt engineering [30, 161]. After being prompted to write a poem on the very contents of this thesis, ChatGPT yielded the following [128]:

In the realm of ones and zeros, where data streams unfurl,
In the dance of signals, where patterns swirl,
There lies a quest, a journey bold and grand,
To grasp the essence of time's shifting sand.

In the tapestry of time series, woven deep,
Where secrets of the past and future sleep,
Representation learning strides with grace,
Unveiling truths in each intricate space.

Through layers of abstraction, it seeks to find,
The essence of patterns, the ties that bind,
In the heartbeat of data, it hears the song,
Of anomalies hidden, waiting long.

In the realm of hybrid-fiber coaxial might,
Where networks span realms of day and night,
Faults lurk in shadows, unseen, untold,
Waiting for detection, for the story to unfold.

With algorithms sharp, like blades they cut,
Through the noise and chaos, without a rut,
Identifying glitches, anomalies rare,
In the vast expanse of signals, they dare.

Representation learning, fault identification's a
A symphony of science, where minds depart,
To unravel mysteries, to chart unknown lands,
In the ever-expanding universe of data's hands.

Accurately and graciously versifying the contents of this thesis, the above poem demonstrates the level commercially available AI and probabilistic generative

models have recently reached, thus also illustrating the importance of and the potential for research in the area of probabilistic generative models.

1.3 Contributions

During this project, multiple approaches to time series analysis on data from broadband networks have been examined resulting in multiple contributions divided among the three research papers included in this thesis. Contributions are both specifically aimed at application to broadband networks but some also have implications reaching further than that. Below is a short introduction to the analyses carried out in each of the papers along with clear contributions from each one:

- **Paper A – SeePD: Detecting Common Path Distortion Faults in Broadband Networks**

  We propose a manually labeled dataset based on remotely gathered data from the broadband network to be used for modeling a specific type of error called Common Path Distortion (CPD), frequently occurring therein. We examine different supervised approaches to modeling the fault including a feature engineering scheme and a simpler thresholding algorithm. Contributions include:

  - A reliable dataset with an accurate ground truth for detecting CPD faults in HFC networks.
  - A precise way of identifying CPD in HFC networks directly using the above dataset including a feature importance analysis.

- **Paper B – Anomaly Detection in Broadband Networks: Using Normalizing Flows for Multivariate Time Series**

  We propose an unsupervised two-phased approach to estimating the density of a window from a multivariate time series. We propose computing the density on a latent representation of the windows using a Normalizing Flow (NF), which is a recent method in generative AI. We propose an algorithm for learning only the systematic behavior of the time series by iteratively excluding observations that fall outside the distribution when learning the NF. Contributions include:

  - A novel two-phased framework for estimating the density of the behavior of a time series within a window preserving contextual information.

  - A novel algorithm for modeling only the systematic behavior in a dataset that includes outlying behavior.
  - A novel algorithm based on bootstrapping for estimating the cumulative density function (CDF) of the densities derived from a multivariate distribution that allows for the calculation of p-values for single observations.
  - An explorative analysis of the outlying behavior in the time series gathered remotely from the TDC NET broadband network.

- **Paper C – Topology Reconstruction in Telecommunication Networks: Embedding Operations Research within Deep Learning**

  We propose an algorithm for training an encoder able to extract relevant time-series events from continuous data with respect to solving the problem of finding the optimal topology in regions where it is not known. We do this by considering an alternative formulation of the ancestral tree reconstruction problem, known from Computational Biology, ensuring uniquely optimal solutions. We use the algorithm for finding the optimal tree directly in the loss function of the encoder using a contrastive approach. Contributions include:

  - An alternative formulation of the tree topology reconstruction problem specifically engineered for the HFC network.
  - A novel algorithm for obtaining the optimal score and solution to the above problem.
  - A conjecture stating that the optimal solution to the above formulation will always yield a *uniquely* best score.
  - A novel approach to leveraging the synergy between the fields of Operations Research (OR) and DL by using an optimization problem directly in a DL loss function.
  - A contrastive approach to training an encoder able to extract relevant events from continuous time series data with respect to the new alternative optimization problem mentioned above.
  - A stochastic version of the algorithm mentioned above to allow for automatic differentiation during the learning stage.
  - An approach to simulate time series data from a given customer modem given the new alternative tree reconstruction setting mentioned above.

1.4 Structure of the Thesis

Overall this thesis consists of two parts. The first part–*Background and Theory*– is designed to equip the reader with the necessary background knowledge to properly read and understand the research carried out as part of the project. The second part–*Research Outcomes*–will initially summarize and discuss the main research contributions and results obtained throughout this project, before the scientific articles produced along the way are included, and thereby ending the thesis. Each of the chapters is introduced briefly in the following:

Part I – Background and Theory
Chapter 2 provides a short introduction to HFC networks and how broadband technology is used to transmit data. This includes the entire setup from the backbone internet to the customers and how the RF signals are used to encode data. This chapter should also give the reader a better understanding of the origins of the different variables in the telecommunications dataset used for some of the work in this thesis.

In chapter 3 a brief introduction to the different models used throughout this thesis will be given. This section will equip the reader with the relevant background theory before diving into applying and modifying the models as part of the novel research included in this thesis.

Chapter 4 provides an overview of the latest advancements or methodologies relevant to our problems at hand. The chapter will both cover previous attempts to detect errors in HFC networks and recent advancements in learning representations of time series mainly using deep learning.

Chapter 5 will introduce the specific broadband dataset provided for this project by TDC NET and used in this thesis. The chapter will cover different preprocessing problems and our attempts at dealing with them. This should inform the reader how the time series data has been made ready for further analysis.

Part II – Research Outcomes
Chapter 6 will summarize, discuss, and contextualize each of the papers produced during this project and hence included in this thesis. This section should provide the reader with the intuition and thought processes behind the direction that this research has taken, and how we believe it fits into both the current operational workflows and the state-of-the-art.

In chapters 7, 8, and 9, the three relevant research papers produced throughout this project–Papers A, B, and C–are included in their unaltered form.

Part I

Background and Theory

Chapter 2

Hybrid Fiber-Coaxial Networks

The HFC network or broadband network is used to distribute an internet signal and/or broadband TV signals from a central backbone network to a set of end customers. Internationally, the backbone network consists of big hubs connected by very big (often undersea) cables. Nationally[1], the backbone network consists of a number of Optical Line Terminals (OLTs) between which information is sent using fiber technology[2]. Each of the OLTs has connected to it a set of Coaxial/cable Media Converters (CMCs) that converts the fiber signal into a RF signal that uses the amplitude and phase of an oscillating signal to encode and transmit a message along insulated copper-wires. A CMC is also sometimes referred to as a fiber-optic node. The RF signal used in this case is modulated according to a given Data Over Cable Service Interface Specification (DOCSIS) that is normally DOCSIS 3.0 or DOCSIS 3.1 or a mixture of the two. DOCSIS 3.1 is a never specification released in 2013 allowing for much higher transfer speeds [19]. See section 2.1 for how these specifications work in practice. The RF signal is sent to and from an end customer using coaxial cables between a sequence of splitters, splitting the cable, and amplifiers, amplifying the signal

[1] Referenced in Denmark - bigger countries might have higher level backbone networks nationally

[2] Fiber technology is used to send information through an optical cable using light. The optical fibers have some advances since there is almost no outside RF interference and very little signal degradation when sent over long distances. The same principles for signal modulation as explained later in this chapter apply.

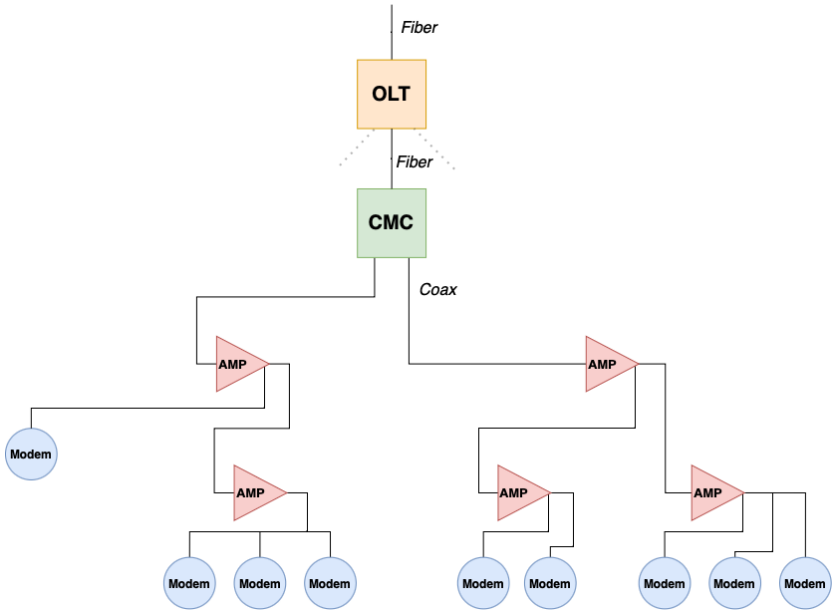


Figure 2.1: Illustration of the topology of the network below a CMC using RF signals and coaxial cables. The red triangles correspond to amplifiers and the red circles correspond to end customers. Often the signal cable is split close to or at the amplifier, but this is not always the case. The OLT is part of the national backbone network.

in the cable. A signal or data sent from the CMC to the customer is denoted DownStream (DS), while the other way is denoted UpStream (US). The coaxial cable consists of an insulated metal conductor in-cased in either a wire mesh or solid metal wrapping to prevent noise from outside RF interference[43]. An example of a coaxial cable is shown in Figure 2.2 and a simplified illustration of the network below a local CMC is shown in Figure 2.1.

2.1 Signals and Modulation in Broadband Networks

To send data using RF, the signal needs to be modulated, i.e. turned from bits of data (0s and 1s) into a representation that can be expressed using an RF signal. In DOCSIS 3.0 specification this is done using the Qudrature Amplitude Modulation (QAM) technology, which uses amplitude and phase changes to encode a

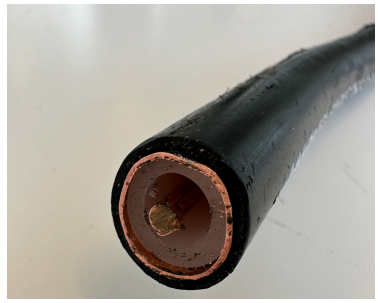


Figure 2.2: Example of a coaxial cable consisting of an inner metal conductor insulated and in-cased in a solid metal tube. The metal tube is used to prevent outside RF interference that causes noise [43].

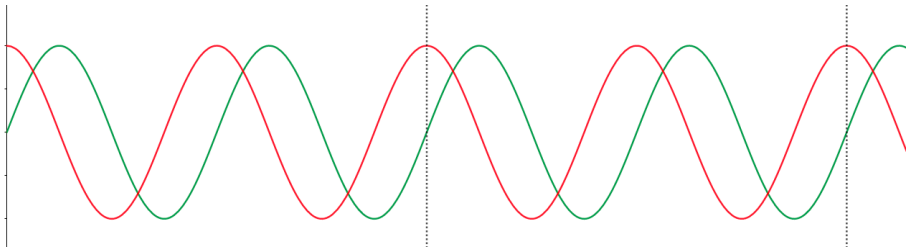


Figure 2.3: Example of the two underlying RF signals used in QAM modulation here with a frequency of two. The quadrature component (green: a cos curve) is shifted 90 degrees (1/4 wavelength) with the in-phase component (red: a sin curve).

message. In the DOCSIS 3.1 specification, the Orthogonal Frequency-Division Modulation (OFDM) technology is used that utilizes more of the frequency band allowing for much higher transfer speeds.

2.1.1 Quadrature Amplitude Modulation

The Quadrature Amplitude Modulation (QAM) technology makes use of a single carrier frequency but in two different phases. Namely the in-phase component (a sin curve) and a quadrature component that is shifted 90 degrees concerning the in-phase component (a cos curve). An example of these two underlying signals is shown in Figure 2.3. A given message at time t is divided into two parts; $I(t)$ and $Q(t)$ is converted into a QAM signal in the following way:

$$s(t) := \sin(2\pi \cdot f \cdot t) \cdot I(t) + \cos(2\pi \cdot f \cdot t) \cdot Q(t) \tag{2.1}$$

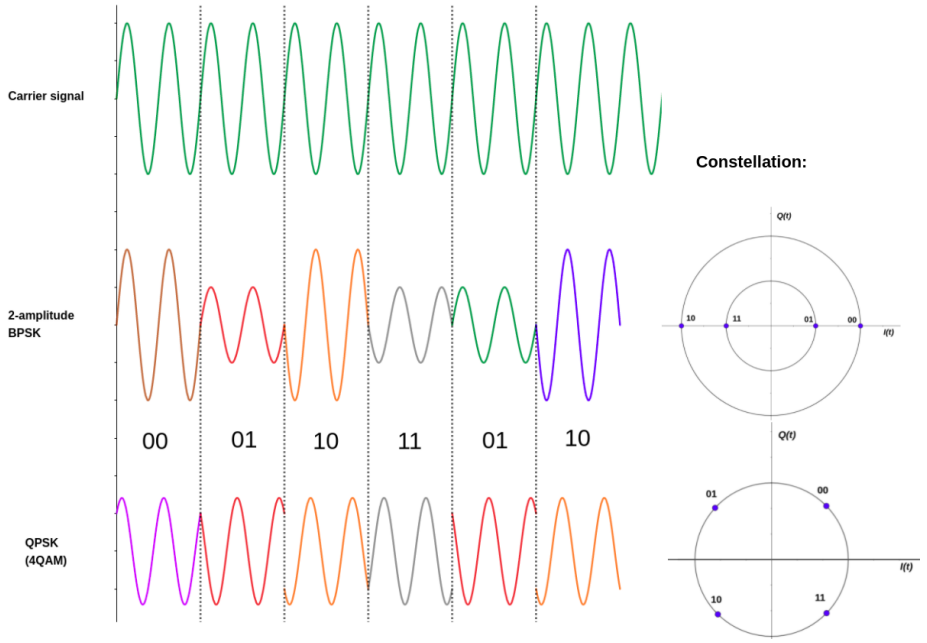
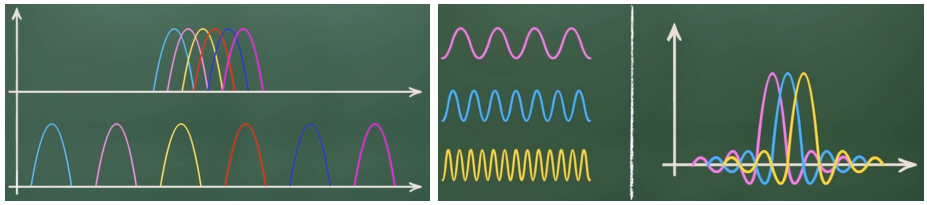


Figure 2.4: Simple example of the QAM technology in which a message is modulated into two different RF signals based on two different constellations. Namely BiPhase Shift Keying (BSPK) with two amplitudes and Quadrature Phase Shift Keying(QSPK) also known as 4QAM.

Where f is the carrier frequency. Notice that $I(t)$ and $Q(t)$ become the amplitudes of respectively the in-phase (sin) signal and the quadrature (cos) signal. The values of $Q(t)$ and $I(t)$ are mapped into a so-called constellation arranged as a grid based on the bit capacity of the system so that the bits can be appropriately decoded on the other end. A simple example of this process is shown in Figure 2.4 where a message is modulated into two different RF signals based on two different simple constellations. Constellations can reach up to 4096QAM (a 64×64 grid) that can send 64 bits per unit time DOCSIS 3.0 specification, and even bigger for DOCSIS 3.1.

2.1.2 Orthogonal Frequency-Division Multiplexing

A limiting factor of the QAM technology is the need for having non-overlapping carrier frequency intervals. An interval is needed for the signal to speed up or slow down to accurately transmit the phase shifts that are a part of the QAM



(a) Illustration of the frequency division scheme of the QAM technology (bottom) and why is it problematic to have them close to each other.

(b) Illustration of the OFDM technology. Notice that carrier frequencies are chosen in such a way that they are 0 whenever any other carrier is at its high point.

Figure 2.5: Illustrations of the frequency division of the QAM and OFDM technologies respectively. Illustrations are taken from [178].

signals. The OFDM technology provides a solution to this problem as it allows for carrier frequencies to overlap. This is done by choosing a set of carrier frequencies so that they are orthogonal to each other. In this way, the Fourier transform can be used to properly demodulate the signal in the receiving end. An illustration of this principle is shown in Figure 2.5. The advantage of using this technology is that it makes it possible to utilize more of the bandwidth and use it to send a higher number of smaller messages, each with more certainty.

2.2 Field service in HFC networks

The high complexity of the HFC network makes it vulnerable in several ways meaning that several potential problems can occur at different locations in the network. Problems that directly impact customer experience include; corrosion where cables are connected, damaged cables, improper installation of Customer-Premises Equipment (CPE), and different outside factors that interfere with the physical properties of the materials and RF signals [76]. For instance, the weather has been shown to impact network performance and degradation [195, 129]. Some problems occur instantaneously (if for instance the cables are damaged during excavation projects) and some occur gradually (for instance corrosion in connectors). In general errors in the HFC network are characterized as being sporadic and difficult to detect.

The large number of problems occurring both instantaneously and gradually means that a vast amount of maintenance is needed, making it both expensive and environmentally harmful to operate the network. This is due to technicians often driving many kilometers between individual jobs and because a big van

is needed to bring equipment to solve every task encountered. The amount of driving is further intensified by network owners not always knowing or having access to a full and updated topology of the network, i.e. the way customers are connected to the CMC. This means that technicians sometimes have to perform a manual search for the location of the root of the problem [138, 35, 115]. As of 2022, TDC NET estimates that their technicians drive approximately 80,000 kilometers daily to service the hundreds of thousands of customer homes spanning the whole country with more than 1.5 Mio. addresses connected (though not necessarily utilzed) [17, 180].

Even though many different errors can occur in the network, TDC NET domain experts estimate the five most common faults they deal with to be:

1. **Cable connections** where cables (in street cabinets) are not properly connected, prone to rust, incompatible, or impaired in another way. Is responsible for a major part of the faults in the network according to domain experts at TDC NET.

2. **Customer-Premises Equipment (CPE)** errors due to improper installation or failure.

3. **Installation errors** caused by technicians connecting the wrong customer in the street cabinet in front of the house.

4. **Cable faults** caused by damage to the cables by e.g. excavation projects. Potential attempts can have been carried out by the homeowner to fix the problem without contacting the Internet Service Provider (ISP).

5. **Sockets** inside the customer homes that are either outdated or connected to a loose cable acting as an antenna and thereby introducing noise into the network.

Figure 2.6 shows an illustration of where the errors mentioned above are typically located in the typical path from CMC to the customer in the HFC setting. One of the typical cable connection errors is the so-called Common Path Distortion (CPD) that is a direct consequence of stress or corrosion and causes the noise from the outside to leak into the signal in the form of ingress [65]. Additionally, it can cause the DS signal to leak into the US signal or vice versa thereby introducing unwanted noise [164].

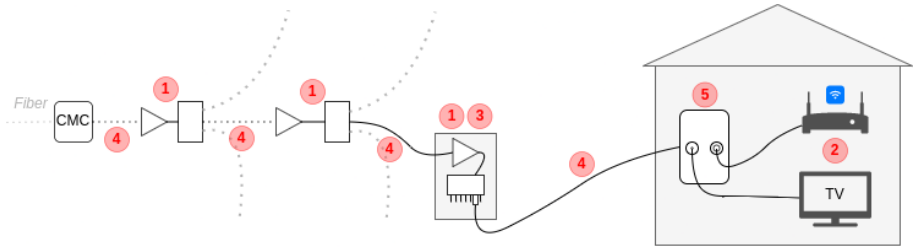


Figure 2.6: Illustration of a typical path from the CMC to the customer through a sequence of amplifiers (white triangles) and splitters (white rectangles). The red numbers correspond to typical fault locations as mentioned in the list in section 2.2.

2.2.1 Trouble tickets and their uncertainty

As explained earlier, problems are hard to find and even harder to proactively detect and classify. In many cases, network owners rely on customer complaints to find problems in the network that need fixing, as opposed to automatically detecting faults and dispatching technicians [77]. This approach, however, is not ideal for several reasons:

- Customer complaints are prone to much uncertainty and depend on multiple factors:
  - The severity of the problem
  - The patience of the customer
  - Customers that are used to continuously bad connections are less likely to make complaints [76]
- Reporting an error means that the customer already experienced bad service and will be more likely to change ISP.
- Only a fraction of the customer complaints lead to technician dispatch and trouble tickets at the network owner. This is due to the possibility of handling many problems over the phone (especially problems with the CPE).
- Often many different ISPs use the same network (with the same network owner). Different ISPs have different mechanisms for handling customer complaints and thereby also for creating trouble tickets for the network owner.

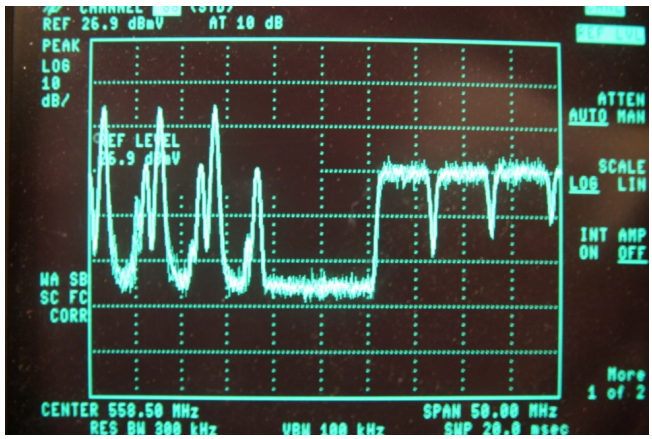


Figure 2.7: Example of Full Band Capture (FBC) for a given modem at a given time. The x-axis shows the carrier frequency and the y-axis shows the power of the signal measured in `dB`. The figure is taken from [184].

Using trouble tickets as ground truth for modeling is thus not considered a viable option and other approaches need to be examined [33]. Other approaches could include: producing a labeled dataset by manually labeling observations; using trouble tickets as hints rather than a ground truth, as already examined by Hu *et al.* in [77]; or detecting anomalies by the use of e.g. unsupervised learning.

2.2.2 Performance metrics

There are many ways of measuring the quality of a broadband signal as a result of the path that it is traversing. The most informative data is based on the Full Band Capture (FBC) in which the power of the signal at every frequency is captured in real-time. Capturing this data, however, requires a spectrum analyzer, which is specialized equipment. Moreover, capturing the full spectrum would result in an excessive amount of data. Due to both of these factors, FBC is primarily reserved for service by technicians who include spectrum analyzers in their everyday equipment. An example of the FBC for a given modem at a given time is shown in Figure 2.7.

Due to the FBC data being too troublesome to gather and work with, other measures are obtained that summarize the performance of the signal at a given time in different ways using the so-called Simple Network Management Protocol (SNMP). Some measures are relatively naïve such as the mean power of the signal measured in `dB`, how many bits or codewords were sent on a given path,

how many of these could be correctly demodulated, how many of these were erroneous, and how many erroneous bits could be corrected using error-correcting codes. [3] Some are more sophisticated by e.g. looking at the power and noise in the actual RF signal before and after demodulation, and some are very complex by looking at the required amount of pre-equalization of the signal needed for it to be properly sent through the network in the US direction [158]. A few of these measures will be described in detail in the following.

2.2.2.1 Modulation Error Ratio

The Modulation Error Ratio (MER) is commonly used for assessing the health of a digital signal [127]. The incoming symbols rarely hit the constellation points described in section 2.1 exactly, but hit according to some distribution (ideally) centered in the constellation points. The MER metric describes how well the incoming signal 'hits' the constellation points and is defined by:

$$\texttt{MER} = 10\log_{10}\sum_{j=1}^{N}\frac{\left(I(j)^2 + Q(j)^2\right)}{(\delta I(j)^2 + \delta Q(j)^2)} \tag{2.2}$$

where I and Q are the in-phase and quadrature parts of the ideal target symbols of the different samples, and I and Q are the in-phase and quadrature parts of each of the incoming sampled symbols. The metric is illustrated in Figure 2.8.

2.2.2.2 Signal-to-Noise Ratio

Whereas the MER explains how well the receiving end is able to perform demodulation of the incoming signal, the Signal-to-Noise Ratio (SNR) describes the noise in the signal either prior to modulation at the transmitting end or after demodulation at the receiving end [14]. This means that the SNR metric captures the noise from the full path, i.e. noise in the original transmitted signal, modulation and transmitter noise, noise contributions from the transport path, and noise in the receiving end due to demodulation.

The SNR is defined to be the ratio of the peak-to-peak power of the baseband signal to the noise within that signal (the noise floor). This means that the SNR is a quantification of how well a signal can be distinguished from the noise at the

[3] Error-Correcting Codes are algorithms that can ensure and potentially correct a package of binary bits. Multiple algorithms exist such as Reed-Solomon codes [151], Low-Density Parity Checks [171], and recently DL has also been investigated as a means of obtaining error-correcting codes [44].

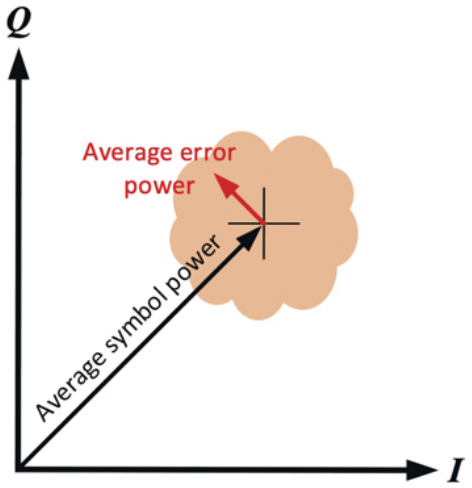


Figure 2.8: Illustration of the Modulation Error Ratio (MER) metric taken from [75]. Incoming symbols rarely hit the constellation points exactly, but according to some distribution centered in it. The better the signal is, the more precise the signal becomes, hence the lower the uncertainty of the distribution.

receiving end after having traversed the whole path. The measure is visualized in Figure 2.9.

2.2.2.3 Pre-equalization Parameters

In order to aid network owners in finding the root causes of poor service *before* the problem would cause an impairment for the customers, CableLabs started phasing in a framework for Proactive Network Maintenance (PNM) with the second generation DOCSIS specification in around 2005 [184]. It leverages values of the parameters of the so-called pre-equalization mechanism designed to correct an imputed signal exiting the modem, i.e. the US direction. This is achieved by a central node (typically the OLT) reading the signal coming from a given modem. If a signal is somehow impaired the OLT sends instructions on how the modem should correct its US signal to achieve the optimal performance [189]. An example of the FBC of an impaired signal before and after pre-equalization is shown in Figure 2.10.

The pre-equalization mechanism can correct signals that suffer from different kinds of distortion including both linear impairments in which the phase and

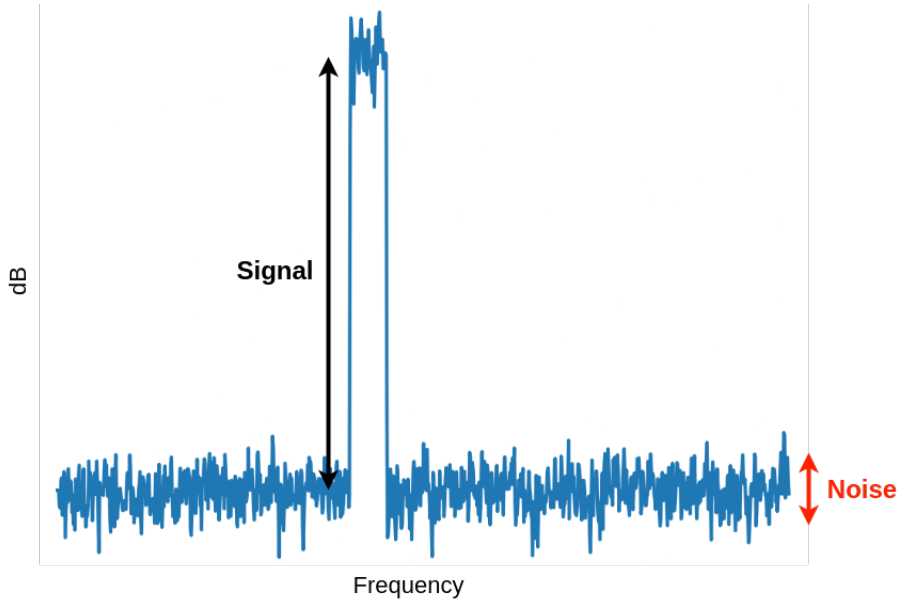
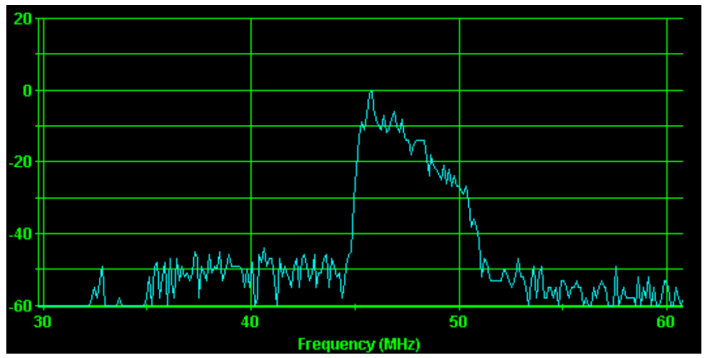
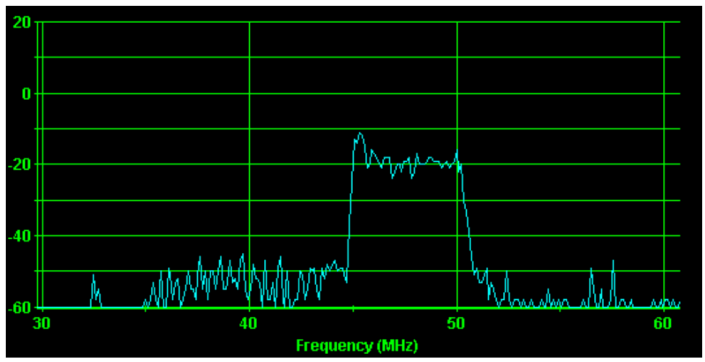


Figure 2.9: Illustration of the SNR principle on a small section of the spectrum. The SNR value is defined to be the ratio between the power of the signal and the power of the noise.



(a) Before applying pre-equalization



(b) After applying pre-equalization

Figure 2.10: Example of how pre-equalization in the US signal can fix an impaired signal caused by problems in the cable path. Here visualized using the FBC on a range of frequencies (x-axes). The y-axes shows the power of the signal measured in dB. Figures are taken from [184].

amplitude of a signal are linearly distorted, and non-linear impairments such as CPD or micro-reflection in which the signal is reflected at some point where the impedance through a cable connection does not match the power of the signal flowing through. It corrects the signal by delaying and amplifying portitions of it as specified by pre-determined coefficients.

Pre-equalization parameters summarize the amount of pre-equalization applied to the US signal at a given time point and how much potential the pre-equalization mechanism has to improve further deteriorating of the signal. One important parameter to highlight is the *Main Tap Compression* (MTC), which is defined as the ratio of the power in the entire signal to the power in the unaltered portion of the signal. This metric provides insights into the condition of the entire path from the backbone internet to the customer modem and indicates how close the signal is to failing or breaking down [184].

Chapter 3

Theory

A brief introduction to a range of ML methods used for various analyses in this thesis. This chapter is meant to equip readers who are new to ML with the necessary understanding to fully comprehend the analyses carried out later on. The chapter can also be used to reference the different methods.

Emphasis will be put on Neural Networks (NNs) and Normalizing Flows (NFs) as a good understanding of both concepts is believed to be important for proper understanding of the research presented in this thesis.

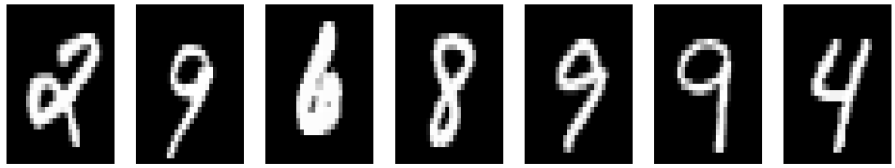


Figure 3.1: Example of seven random images from the MNIST dataset including their labels. Each of the 28×28 pixels has an intensity value between 0 (black) and 255 (white).

3.1 Classification models

One of the most intuitive ways to use ML is for classification purposes. In a classification model, it is assumed that all the observations in the training dataset, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, will have a corresponding label, $y_1, y_2, \ldots, y_N$, from a dictionary of potential labels, $\mathcal{Y}$. The goal of the model is then to learn how to distinguish the different classes in the best way. A very classic example of a classification problem is that of the Modified National Institute of Standards and Technology database (MNIST) dataset that consists of small grayscale images of one of the digits between 0 and 9 where the goal for the model is to determine the correct digit using the image as an input [96]. A few MNIST digits with labels are shown in Figure 3.1.

A wide range of classification models exist which is the reason why only the few methods that are used for analysis later in this thesis will be covered. One of the methods used for simple classification purposes later on is called a MultiLayer Perceptron (MLP) which is a different term for a simple artificial NN with more than one layer. See section 3.2 for an introduction to those.

3.1.1 Logistic Regression

Logistic regression is a type of generalized linear model that requires the label dictionary to be binary, i.e. $\mathcal{Y} = \{0, 1\}$. The desired output of the model is a probability for either class, i.e. π_i. An example of a labeled dataset with one explanatory variable is shown in Figure 3.2 on the left. It is immediately clear that an ordinary linear regression will not work in this case, both due to the

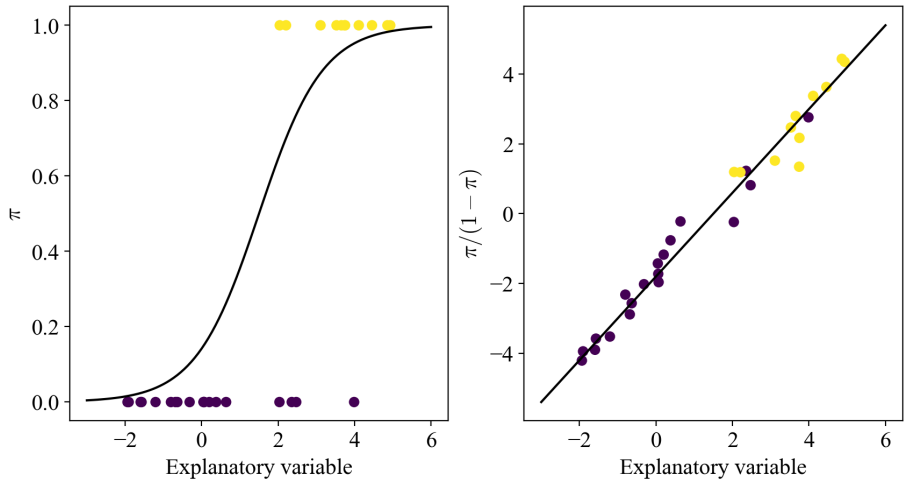


Figure 3.2: An example of a logistic regression problem. A model has been learned that estimates the probability of an observation belonging to the yellow class, π. It does this having learned a linear model on the explanatory variables in the logit-tranformed space which is visualized on the right.

outcome not being proportional to the explanatory variable, and also because an ordinary linear model can take values outside the interval from zero to one. For these reasons, the logit function is used as a link function mapping the interval $]0;1[$ into the reals, $\mathbb{R}$, thus easing the linear modeling using the explanatory variables in the data [49]. Given an observation, $\boldsymbol{x}_i$ the link function is given by:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0 \tag{3.1}$$

Where $\boldsymbol{\beta}$ is the estimated parameter vector and β_0 is the estimated offset. In Figure 3.2 of the right, it is immediately clear that the logit-transformed space provides a much better basis for a linear model. The black regression line shown on the right is transformed into the estimated probabilities shown by the black line on the left plot in the figure.

Logistic regression models are usually learned by maximizing the log-likelihood of the observed outcome with respect to the parameters of the model. In this case, the observation can be seen as independent binary random variables that

according to the Bernoulli distribution will have a joint distribution given by:

$$\begin{aligned} l(\pi_1, \pi_1, \ldots, \pi_N; y_1, y_2, \ldots y_N) &= \prod_{i=1}^{N} \pi_i^{y_i} (1-\pi_i)^{(1-y_i)} \\ &= \exp\left(\sum_{i=1}^{N} y_i \log(\pi_i) + (1-y_i) \log(1-\pi_i) \right) \end{aligned} \tag{3.2}$$

Now rewriting the link function we get:

$$\pi_i = \frac{1}{e^{-(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0)} + 1} \qquad \text{and} \qquad 1 - \pi_i = \frac{e^{-(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0)}}{1 + e^{-(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0)}} \tag{3.3}$$

This means that the maximum log-likelihood optimization problem will be given by:

$$\begin{aligned} &\max_{\boldsymbol{\beta}, \beta_0} \left\{ \sum_{i=1}^{N} y_i \log\left(\frac{1}{e^{-(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0)} + 1} \right) + (1-y_i) \log\left(\frac{e^{-(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0)}}{1 + e^{-(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0)}} \right) \right\} \\ &\qquad = \max_{\boldsymbol{\beta}, \beta_0} \left\{ \sum_{i=1}^{N} y_i (\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) - \log\left(1 + e^{\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0} \right) \right\} \end{aligned} \tag{3.4}$$

3.1.2 Random Forest

A random forest is a so-called ensemble method because it consists of a set of different *decision trees* all voting independently. The intuition is that each tree when presented with an observation, will output its most likely class given the information utilized by it. The majority vote of all trees in unison will then be a good generalization of the estimated class of the observations [105].

A decision tree is a tree in which the nodes correspond to a test on a variable from the dataset and the branches to the different outcomes of the test. The leaves of the tree represent an output class. One starts at the root node of the tree and walks along the edges as dictated by the value of the parameter examined at that step. An illustrative example of a decision tree used to classify fruit is shown in Figure 3.3. The decision tree would use metrics such as the Gini index to choose the optimal feature and test to use at every node in the tree given the data. The Gini impurity calculates the probability of a random observation from the dataset being misclassified by the tree is defined by:

$$I_G(p) = 1 - \sum_{j=1}^{J} p_j^2 \tag{3.5}$$

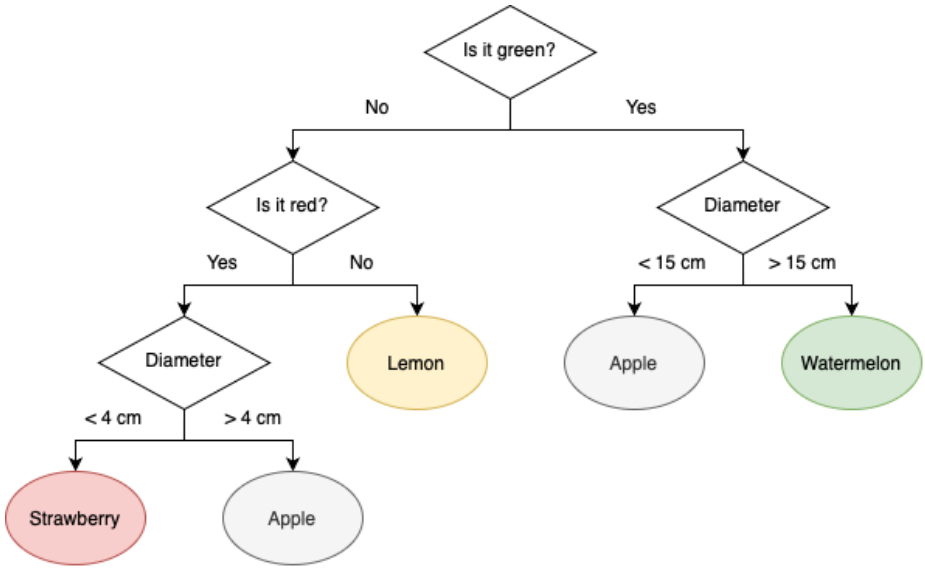


Figure 3.3: Example of a decision tree used to classify fruits. One starts at the top and follows the branches based on the value of the tested parameter in each node. Once a leaf node is reached its value becomes the predicted class.

where J is the number of classes in the dataset and p_j is the relative frequency of class j. One way to train the decision tree is thus to choose the feature and threshold that maximizes the decrease in impurity. In order not to overfit the decision trees, multiple approaches can be applied. For instance, early stopping allows the user to specify some stopping criteria (e.g. every leaf having below a number of observations), or post-pruning can be used to remove branches that do not result in a significant improvement of the model.

The random forest is fitted to the data using a method called *bagging*. In bagging, a number of different trees are fitted. For every tree, a random subset of the observations and/or features in the dataset is used to fit it. If a single tree were to be trained on all the data it would often lead to overfitting as the model would learn very irregular patterns. However, learning many trees that each is less irregular has shown to generalize the data much better [13].

3.1.3 XGBoost

In the eXtreme Gradient Boosting (XGBoost) model many concepts are introduced designed to improve the performance of random forests and decision trees. The model entails sequentially adding so-called gradient-boosted decision trees to the model [31]. Unlike decision trees, gradient-boosted trees use continuous outcomes that can be used to calculate residuals[1] using a differentiable loss function. Residuals are calculated before each tree is added to the model based on the intermediate predictive power of the preliminary model. Each gradient-boosted decision tree is built by considering how to improve the intermediate residuals (maximizing the gradient of the loss function) while the residual and the predictive improvement are used to decide the magnitude of the contribution of the new tree. In the end, an XGBoost is an additive model considering a sequence of gradient-boosted trees making it an ensemble method. The model outputs the log odds of the given observation belonging to each of the classes (and thus can be turned into a probability):

$$\hat{y}_i = \phi(\boldsymbol{x}_i) = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i) \tag{3.6}$$

where K is the total number of trees and f_k is the kth gradient-boosted tree that has a structure and leaf weights associated with it. To learn the first tree, the null model considering the overall likelihood of any observation belonging to each of the classes is used.

[1] Each leaf outputs the log-odds of the observation belonging to each of the classes

The XGBoost model is further improved by considering a loss function that does not only depend on the intermediate residuals but also a regularization term that constrains the complexity of the model thus preventing it from overfitting, i.e.:

$$\mathcal{L}(\phi) = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \left(\gamma T_k + \frac{1}{2} \lambda \|\boldsymbol{w}_k\|^2 \right) \tag{3.7}$$

where l is a convex loss-function like the Mean Squared Error (MSE) between the outcome and the predictions, T_k is the number of leaves in the kth tree, $\boldsymbol{w}_k$ are the weights of the tree, while γ and λ are hyper-parameters controlling the regularization of the tree.

3.2 Neural Networks

NNs have long been state-of-the-art in many fields due to their flexibility and intuition based on neurons activating and sending information between each other like in the brain. The NN is built of a number of neurons that are typically arranged in layers with the input in one end and the output in the other making it possible for the model to learn increasingly complex patterns. In the ordinary, artificial NN, each neuron or node calculates an activation by multiplying the input with a set of weights, adding a bias, and subsequently passing the result through a non-linear activation function as shown in Figure 3.4. Typical choices of the activation functions include the *sigmoid* and *tanh* functions because their domain includes the entire real line while their codomain is restricted between two values that can be interpreted as corresponding to activated, respectively, not activated.

$$\texttt{sigmoid} : \mathbb{R} \to]0\,,1[\qquad \texttt{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{3.8}$$

$$\texttt{tanh} : \mathbb{R} \to]-1\,,1[\qquad \texttt{tanh}(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{3.9}$$

For the output in a classification setting, however, it is typically necessary to combine the outputs of multiple neurons that each corresponds to a class. This is usually done using the so-called *softmax* function. The softmax function can take in a vector of real values and output a vector with values summing to one, allowing for probabilistic interpretation of the likelihood for each class.

$$\texttt{softmax} : \mathbb{R}^C \to \{\boldsymbol{y} \in]0\,,1[^C \mid \|\boldsymbol{y}\|_1 = 1\} \quad \texttt{softmax}(x)_i = \frac{e^{x_i}}{\sum_{c=1}^{C} e^{x_c}} \tag{3.10}$$

A NN is very flexible and can easily be modified to fit the problem at hand by stacking different types of layers that each have different properties (these

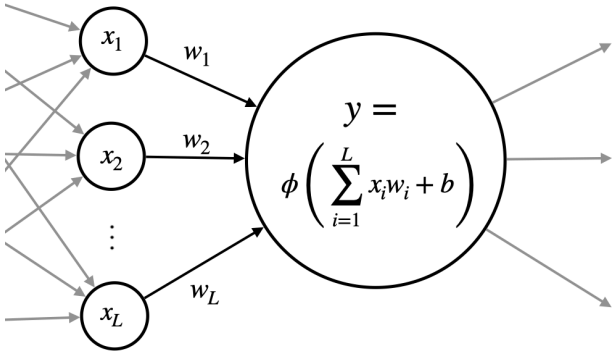


Figure 3.4: Illustration of how the information flows through a neuron taken from [146]. w_i corresponds to a weight multiplied onto one of the input values x_i, b is the bias added to the product before passing it through the non-linear activate function, ϕ.

will be discussed in the following). The overall layout of the network including how many layers the network comprises and the number of nodes in each is denoted by the *architecture* of the NN. A simple example of an architecture is shown in Figure 3.5 and comprises a single hidden layer of the so-called *dense* or *fully-connected* type, meaning that every node in the layer is connected to every node before and after it. This is one of the basic building blocks of the neural network. Multiple of these layers stacked together is sometimes also denoted a MultiLayer Perceptron (MLP).

3.2.1 Training a Neural Network

Training a NN corresponds to finding the optimal weights in all layers of the network, such that the whole model accurately explains the dataset. Given a dataset comprising N observations, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ with labels, $y_1, y_2, \ldots, y_N$ and a neural network, f_Θ with weights Θ, the problem is given as:

$$\arg\min_{\Theta} L(\Theta) \qquad \text{where} \qquad L(\Theta) = \frac{1}{N}\sum_{i=1}^{N} l\left(y_i, f_\Theta(\boldsymbol{x}_i)\right) \tag{3.11}$$

and where l is the *loss function* specifically chosen by the user to fit the given problem. For classification purposes it is customary to use the cross-entropy

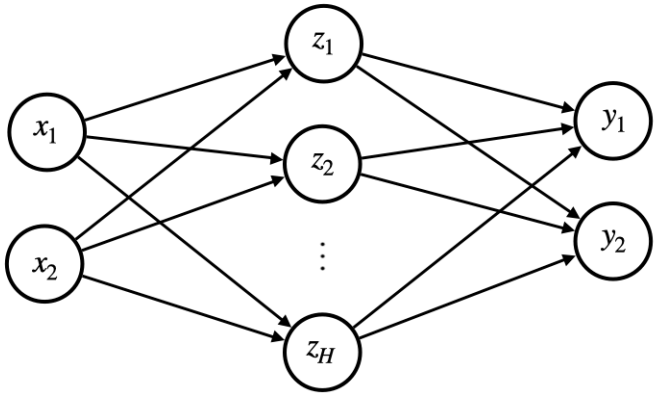


Figure 3.5: The architecture of a simple NN with one dense hidden layer taken from [146]. The network is built for taking an input with two values and outputting two values, while the hidden layer has H neurons.

loss given by:

$$l_{\texttt{cross-entropy}}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = -\sum_{c=1}^{C} y_{i,c} \cdot \log(\hat{y}_{i,c}) \tag{3.12}$$

where C is the number of classes in the dataset, meaning that the outcome becomes a vector. Loss function can be modified any many different ways by comprising e.g. multiple terms or regularization to prevent overfitting. For regression purposes, the MSE is usually used as in ordinary statistical linear regression.

During training, the loss function is incrementally minimized by updating the weights of the network using stochastic gradient descent or other mathematical optimization algorithms. In other words, the derivate of the loss function is calculated with respect to the weights of the network, and the weights are updated by e.g. taking a step in the direction of maximum descent (in stochastic gradient descent):

$$\Theta_{new} = \Theta_{old} - \gamma \nabla L(\Theta_{old}) \tag{3.13}$$

where γ is the *learning rate* specified by the user. Derivatives with respect to the weights of the different layers are made tractable by repeated application of the chain rule given by:

$$h(x) = f(g(x)) \qquad \Rightarrow \qquad \frac{\partial h}{\partial x} = \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial x} \tag{3.14}$$

such that the derivative of the previous layer can be calculated using derivatives and values of the current. This concept of the derivatives being carried backward

from the output is also used in the so-called *backpropagation algorithm*. In backpropagation, the result of a forward pass is evaluated in the loss function. The derivate of the loss function is numerically propagated backward through the network estimating derivates along the way. Backpropagation is often used in practice in tools such as the `autograd`-package in `PyTorch` if using Python for implementation [134].

3.2.2 Convolutional Neural Networks (CNNs)

Concolutional Neural Networks (CNNs) make use the the so-called convolutional layer. Instead of single neurons with weights and biases, a number of filters or kernels are convoluted over the input to create the input to the next layer. Stacking more convolutional layers will allow the model to look for increasingly complex features in the input. For instance, in an image, the first layer will look for low-level features such as lines and curves, and later layers will be able to detect high-level features such as specific objects. The application of filters allows for the model to consider spatial patterns in an image or temporal patterns in a time series. This makes the models able to encode more complex datasets while training filters which are somewhat interpretable [198]. The ability to train interpretable filters has made these types of layers very popular for analysis on images [167], videos [145], and time series [64].

The filter is positioned in the image, and an element-wise product[2] with the image is calculated before being summed to arrive at the output. The operation is visualized in Figure 3.6. Users can specify different hyper-parameters to control the size of the filter and the size of the output given the input. For instance, the stride dictates the step size within the image, the dilation dictates the step size in the input relative to the elements of the filter, and the input image can be adequately padded with zeros if one wants the same input and output sizes. The size of the output for any dimension as a function of the hyper-parameters for that dimension can be calculated using the following:

$$d_{out} = \left\lfloor \frac{d_{in} + 2 \cdot P - \delta \cdot (K - 1) - 1}{\Delta} + 1 \right\rfloor \tag{3.15}$$

where P is the padding, δ is the dilation, K is the kernel size, and Δ is the stride. Filters can be appropriately initialized to be applied to any number of dimensions and channels in the input, while it is customary to train multiple filters able to look for different features in a single convolutional layer which will correspond exactly to the number of channels in the output.

[2] The elementwise product is conducted over all dimensions including the channel dimension. Specific for the channel dimension in the filter, though, is that its size has to equal the number of channels in the input

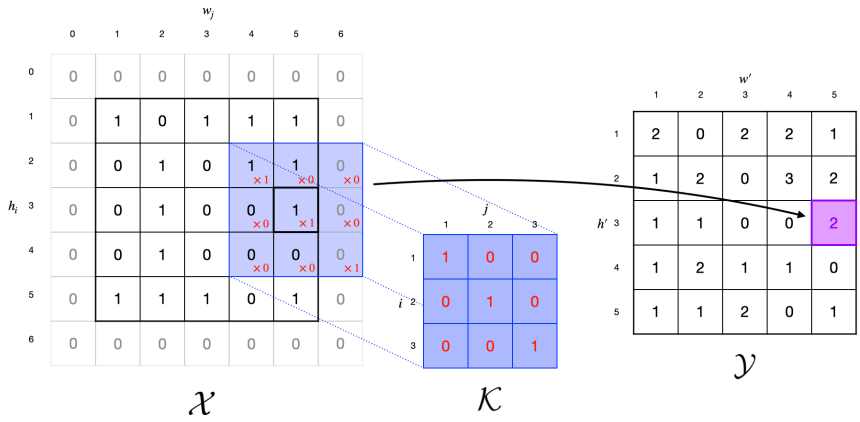


Figure 3.6: Illustration of a simple convolution on an image taken from [146]. The filter, $\mathcal{K}$ is applied to the input, $\mathcal{X}$ to calculate the output, $\mathcal{Y}$. The filter is positioned in the input image and the weights of the filter are multiplied with the image values element-wise before the result is summed and the output calculated. Appropriate padding (grey zeros) is added to make the output the same size as the input. Notice that the same filter is applied to the whole image and that it is customary to have multiple filters resulting in multiple output images (or channels).

Convolutions can also be slightly adjusted to work in the opposite direction, i.e. working to generate the input based on the feature map. This is challenging since convolutional layers often shrink the temporal or spatial dimensions. It can, however, be achieved by so-called *transposed convolutions* that also go by the names *deconvolutions* or *fractionally-strided convolutions* [51]. This operation is designed to reverse the ordinary convolution by choosing appropriate parameters in order for the original size of the input to be restored. This might require zeros additionally padded between rows and columns to reverse a stride higher than one. Deconvolutions are often used in AutoEncoders (AEs) as part of the decoder in an encoder-decoder architecture for unsupervised modeling. In an AE, the decoder attempts to reconstruct the input from a latent representation obtained by the encoder through a bottleneck layer.

3.3 Normalizing Flows

The Normalizing Flow (NF) is a recent framework for generative or probabilistic modeling, which means that it seeks to estimate the density of the input data space while allowing for subsequent sampling from it.

In an NF, the density of the input distribution is modeled by applying a sequence of *invertible* transformations, f_1, $f_2, \ldots, f_K$, to a well-known base distribution, such as the Gaussian, denoted p_0 (samples are represented by $\boldsymbol{z}_0$). Applying each of the transformations one by one will turn this distribution increasingly complex ending up with p_k modeling the desired input distribution (with samples $\boldsymbol{z}_k = \boldsymbol{x}$). Ensuring the invertibility of the transformations enables both sampling directly from the input space (by sampling in the base distribution and passing it through the transformations) and also calculating tractable and exact probabilities using the change-of-variable formula, known from probability theory:

$$p_Y(y) = p_X(x) \cdot \left| \frac{\partial x}{\partial y} \right| \qquad \text{where} \qquad y = g(x) \Leftrightarrow x = g^{-1}(y) \tag{3.16}$$

For transformations $\boldsymbol{z}_i = f_i(\boldsymbol{z}_{i-1})$ that follow distributions $\boldsymbol{z}_i \sim p_i(\boldsymbol{z}_i)$, the the

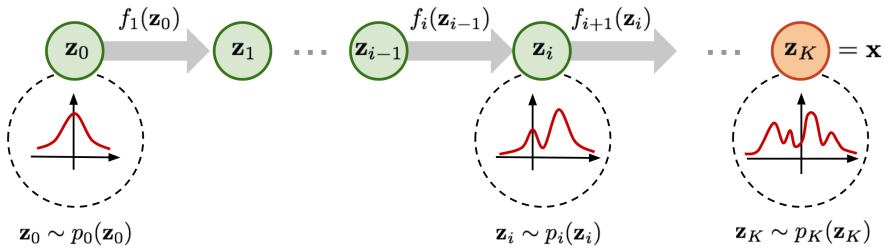


Figure 3.7: Illustration of the NF framework taken from [192]. The sequence of transformations, $f_1, f_2, \ldots, f_K$ incrementally transforms the base distribution, $\boldsymbol{z}_0$ (Gaussian) into increasingly complex distributions ending in $\boldsymbol{z}_K$ modeling the input data space.

density of an observation $\boldsymbol{x} = \boldsymbol{z}_K$ can be calculated as:

$$p_K(\boldsymbol{z}_K) = p_{K-1}(\boldsymbol{z}_{K-1}) \cdot \left| \det \frac{\partial \boldsymbol{z}_{K-1}}{\partial \boldsymbol{z}_K} \right| \tag{3.17}$$

$$= p_{K-1}(\boldsymbol{z}_{K-1}) \cdot \left| \det \frac{\partial f_K(\boldsymbol{z}_{K-1})}{\partial \boldsymbol{z}_{K-1}} \right|^{-1} \tag{3.18}$$

$$= p_{K-2}(\boldsymbol{z}_{K-2}) \cdot \left| \det \frac{\partial f_{K-1}(\boldsymbol{z}_{K-2})}{\partial \boldsymbol{z}_{K-2}} \right|^{-1} \cdot \left| \det \frac{\partial f_K(\boldsymbol{z}_{K-1})}{\partial \boldsymbol{z}_{K-1}} \right|^{-1} \tag{3.19}$$

$$\vdots \tag{3.20}$$

$$= p_0(\boldsymbol{z}_0) \cdot \prod_{k=1}^{K} \left| \det \frac{\partial f_k(\boldsymbol{z}_{k-1})}{\partial \boldsymbol{z}_{k-1}} \right|^{-1} \tag{3.21}$$

which means that if all the transformations are easily differentiable, only the density in the base distribution, p_0, is needed. This is one of the main advantages of NFs over other probabilistic models. The simplicity of the framework enables exact and tractable calculation of densities rather than having to estimate a proposal distribution by optimizing the Evidence Lower Bound (ELBO) as is the case for e.g. Variational Auto Encoders (VAEs) [89] and other types of variational inference [40].

3.3.1 Invertible transformations

One of the main challenges of the NF is how to choose the sequence of transformations so that they are flexible and able to learn complex patterns, while still being invertible. One way of doing this was proposed by Dinh et al. in 2017

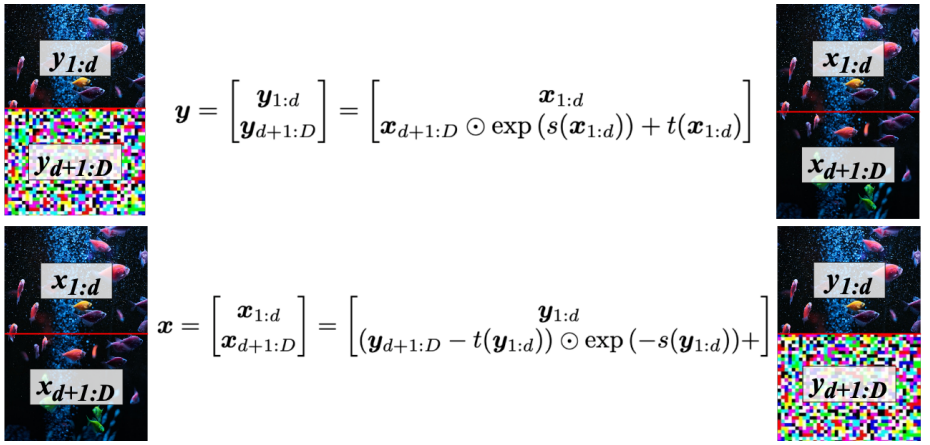


Figure 3.8: Illustration of both directions of the coupling layer proposed by Dinh et al. in [48]. In the forward direction (top), the top part of the image is used to scale and translate the bottom part. In the backward direction (bottom), the scaling and translation are reversed using the encoded image.

and was part of popularizing NFs as a framework [48]. The authors proposed using so-called *coupling layers* as their transformations. In a coupling layer, the input is split into two parts whereof one is encoded to be exactly itself using the identity operation, while also being used to calculate a scaling and a translation applied to the other part. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the input, respectively, the output of a coupling layer, and let D be the size of the input. Given a $d < D$, the coupling layer is defined to be:

$$\boldsymbol{y}_{1:d} = \boldsymbol{x}_{1:d} \tag{3.22}$$

$$\boldsymbol{y}_{d+1:D} = \boldsymbol{x}_{d+1:D} \odot \exp\left(s(\boldsymbol{x}_{1:d})\right) + t(\boldsymbol{x}_{1:d}) \tag{3.23}$$

where $\odot$ is the element-wise product, and where s and t calculate a scaling, respectively, translation that is applied to the second part of the input given the first. Defining the transformation in this manner makes it easy to reverse the process since the scaling and translation values can be recovered by applying the s and t to $\boldsymbol{y}_{1:d}$ and subsequently reversed as illustrated in Figure 3.8. In reality, images are not divided as shown in the figure but rather divided along the channel dimension (for multiple channels) or using a checkerboard pattern in the spatial dimensions. Also, it is ensured that two sequential coupling layers do not encode the same part of the input, such that the whole observation is modified after a sequence of a few coupling layers has been applied. An example of the output of each of the coupling layers from an NF with 20 coupling layers trained on the simple synthetic two-dimensional "Two moons" dataset

Figure 3.9: The evolution of the synthetic two-dimensional "Two Moons" dataset as it is sequentially transformed using an NF. The model is trained alternately modifying each of the two dimensions with the other. The red point is added to show the evolution of a single point.

from `scikit-learn`([136]) is shown in Figure 3.9. The model is trained by alternating between modifying each of the two dimensions with the other. This is somehow visible, as it seems that the latent data points are paired from the perspective of one of the axes.

Another advantage of the coupling layer is that the derivatives of s and t are not needed when calculating the derivate of the whole transformation. In fact, the Jacobian of the transformation is given by:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}^\top} = \begin{bmatrix} \frac{\partial \boldsymbol{y}_{1:d}}{\partial \boldsymbol{x}_{1:d}^\top} & \frac{\partial \boldsymbol{y}_{1:d}}{\partial \boldsymbol{x}_{d+1:D}^\top} \\ \frac{\partial \boldsymbol{y}_{d+1:D}}{\partial \boldsymbol{x}_{1:d}^\top} & \frac{\partial \boldsymbol{y}_{d+1:D}}{\partial \boldsymbol{x}_{d+1:D}^\top} \end{bmatrix} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0} \\ \frac{\partial \boldsymbol{y}_{d+1:D}}{\partial \boldsymbol{x}_{1:d}^\top} & \operatorname{diag}(\exp\left(s(\boldsymbol{x}_{1:d})\right)) \end{bmatrix} \tag{3.24}$$

which means the derivate only depends on the scaling functions (and not its derivative). This means that both s and t can be chosen to be arbitrarily complex functions allowing for the use of e.g. NNs.

Chapter 4

State-of-the-Art

In this chapter, relevant recent advancements that have paved the way for the research presented in this thesis will be briefly discussed. Emphasis will be put on anomaly detection in broadband networks and representation learning in time series, as these are both important topics for the development of our work.

First, the state of the art in outlier/anomaly detection in HFC networks will be explained, highlighting the common problems that authors have faced and their proposed solutions.

Second, different approaches to learning representations in time series will be discussed, demonstrating the many different approaches that can be taken to this problem. Mainly approaches that apply some kind of NN as part of their model will be discussed.

4.1 Faults in Broadband Networks

Broadband networks are well-studied and because the HFC technology has been and still is an important part of the digital infrastructure, multiple societies have been created that work with them. These include Society of Cable Telecommunications Engineers (SCTE), International Society of Broadband Experts (ISBE), and National Cable Television Association (NCTA) that work together in developing broadband network design, implementation, and operation. However, much of this research uses the FBC data and not the time series data gathered remotely by the ISPs, also sometimes referred to as SNMP data. Due to the FBC data requiring special equipment to read it on site, this makes it a great tool for technicians when troubleshooting at a given customer's premises, but not appropriate for remote collection and storing. Along with the complex nature of the HFC network, this makes it hard to detect problems proactively and has resulted in this problem not having been studied in much detail. This is partly due to the limited knowledge of how different errors specifically affect the remotely gathered metrics and because many faults have different levels of severity. Another reason for the limited research stems from the characteristic essence of both network owners and customers. Due to their business model, network owners are usually reluctant to repair issues that do not cause immediate problems for the customers. Additionally, customer calls are not directly correlated with the quality of the signal but rather human aspects such as patience, availability, and whether or not they are used to a bad signal [33, 77, 130].

In this section, a brief introduction to previous approaches to detecting fault in HFC networks will be given. First, the methods using the FBC data will be presented followed by the methods using remotely gathered data.

4.1.1 Fault Detection Using Full-Band Capture Data

Because the FBC data includes real-time power measurements for each frequency at specific time points, it provides detailed information about current signal issues. This data can even be used to classify different types of problems.

Especially CPD is a type of error that is well-understood and investigated. In 2004, Thomas H. Williams submitted a patent for a method of detecting CPD using FBC data [194]. Williams tested his method by introducing artificial CPD errors into a network and observing how this error would affect the signals. He claims that his method can be used as long as one has a spectrum analyzer and that it can even be used to estimate the distance to the source of CPD.

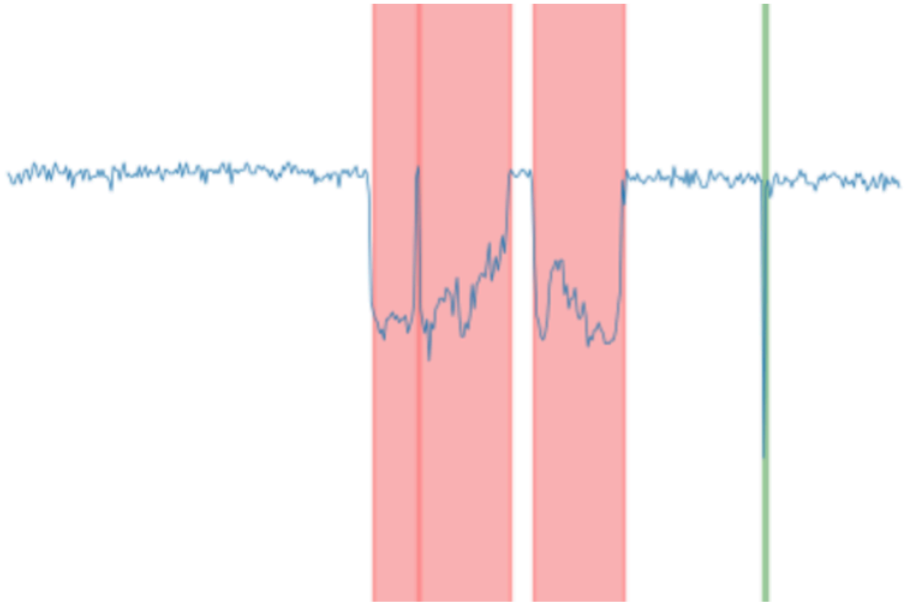


Figure 4.1: Taken from [203]. A snapshot of receiver MER measurements (y-axis) from a range of carrier frequencies (x-axis). Two types of errors are present in the plot. The red regions correspond to ingress, i.e. noise entering the signal at these frequencies. The green region shows a so-called spike anomaly.

Zhu et al. (2020) used a manually labeled dataset where each observation consisted of a snapshot of the receiver (DS) MER values for all subcarriers (frequencies) [203]. Each observation was labeled according to at which frequency bands there were faults present. Additionally, for each fault, a label corresponding to one of five different types of physical-layer anomalies typically seen in FBC data was also given. Figure 4.1 shows an illustration of the receiver MER measurements as a function of carrier frequency with two different types of errors marked. This dataset allowed the authors to train a supervised model using a one-dimensional CNN able to both detect and classify the error.

Gibellini & Righetti (2018) made an unsupervised attempt to detect ingress, which is when outside noise is introduced into the signal [65]. The authors claim that ingress is happening simultaneously with signal leakage[1], why the detection of ingress can lead to the detection of leakage in the network. Their

[1]Signal leakage is often due to a connector not allowing the signal to traverse freely due to it being loose or rusty.

dataset consists of snapshots of FBC data that each consists of signal power measurements for a range of carrier frequencies. The authors use k-means clustering to cluster the snapshots with similar behavior. They claim that a few clusters will explain a big part of the variation and that some of the clusters will correspond to the presence of ingress. They achieve promising results, but end up stating that more work is needed for their approach to be fully functional.

4.1.2 Remote Fault Detection

Though FBC data seems to be good for detecting faults in the networks, it is inconvenient due to the vast amount of data and the manual labor required to gather it. Meanwhile, the remotely gathered data enables viewing the bigger picture in time i.e. looking at temporal data from many different modems at once. As mentioned earlier, one of the main challenges with this approach is the missing ground truth as customer calls or trouble tickets are not considered a reliable indicator for errors in the network. Authors tackle this problem in different ways.

Hu et al. (2020) attempted to use trouble tickets as hints instead of as a ground truth [77]. They did this by defining so-called *ticketing rate ratios*. Given a set of time windows from a number of modems they would consider the mean rate of tickets that come in for all of these windows. They would use this to find smart thresholds for the parameters. For a given parameter and threshold, they would identify in which time windows this parameter is exceeded across all modems. This results in two groups of time windows. One where the parameter is exceeded and one where it is not. They calculate the ticketing rate in each of the two groups and subsequently the ratio between the ticketing rate in the group where the parameter is exceeded to the rate in the other group. A high ticketing rate ratio means that the rate of trouble tickets coming in when the parameter is exceeding the threshold is significantly higher than when it is not. See Figure 4.2 for an illustrative example. Even though the method seems to work well and can be used even on running statistics of the parameters like the variance, the method cannot be used to train a more complex model and thereby take into consideration for instance the context. The method can, however, be used to validate a model trained differently, of course, given that trouble tickets are available.

Other authors have presented approaches based on hard thresholding of relevant parameters. In 2017, Dr. Franklin Lartey presented a set of scorecards that could be used to assess the performance in a network based on values of different parameters. Lartey developed these scorecards based on expert knowledge of how the HFC network works. Lartey does, however, state that these

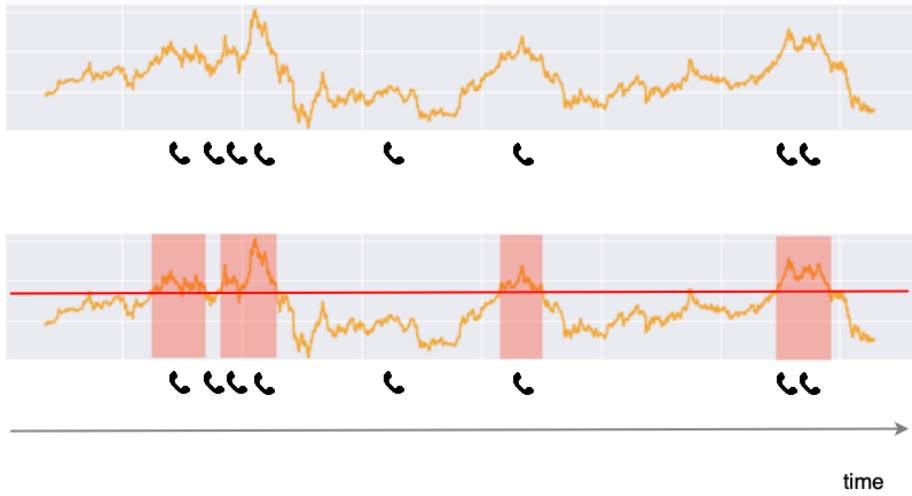


Figure 4.2: Illustration of the intuition behind the ticketing rate ratios. The threshold (red line) is chosen such that the windows where it is succeeded (red) areas have a higher ticketing rate ratio than the rest of the time series.

scorecards still require manual labor and that the method requires continuous developments as the complexity of the HFC technology increases as new improvements of it are deployed. In 2019, Rupe & Zhu proposed a platform for proactive operations, basically automating the method proposed earlier with a platform that is customizable for the individual ISP.

Heiler et al. (2022) would use trouble tickets in a supervised manner by extracting the tickets leading to a technician being dispatched to fix an actual problem in the network [71]. They would analyze the corresponding technician notes to extract the problematic amplifier as identified by the technician. The authors aggregated all data below the last-line amplifiers (the last layer of amplifiers to which the customer modems are connected) to create a (highly unbalanced) dataset with labels based on the extracted problematic amplifiers. As a baseline model, a simple business rule is implemented based on years of knowledge of domain experts. The authors evaluate multiple ML models and compare them to the baseline model. These include logistic regression, Lightgbm which is a gradient-boosted tree, and different time series-oriented variations of NNs. Heiler et al. conclude that deploying ML models improves the detection performance more than 2.3 times over the baseline and that this can be used to optimize troubleshooting time for technicians.

Multiple approaches using big-data platforms have also been investigated. In

2021, Simakovic & Cica (2021) proposed detecting *hard failures* (failures with a total loss of connectivity) by looking at the overall number of online users at a given time. Sudden drops in this number could indicate a bigger problem needing immediate action from the ISP [173]. In 2022, Simakovic et al. improved this big data platform by investigating more significant features that could be used to surveil the performance of the network overall [172]. These included features like the average SNR and power values in the DS signal.

4.2 Representation Learning for Time Series

Because Representation Learning (RL) techniques are mostly unsupervised, the challenge is to extract valuable representations that are intuitive and interpretable while not having access to labels. RL for time series have not been studied as well as for other types of data–maybe due to the fact that in well-studied fields like image and text analysis, the representations and the data itself are much easier to interpret as a human. Recently, some interesting research in the field of time series RL has been conducted. A short introduction to some of these works will be given in the following.

4.2.1 Encoding Relevant Temporal Information

There are many ways of considering the context of a time series. Traditionally, this is exactly what the Recurrent Neural Network (RNN) was designed to do, being able to memorize previous values when evaluating a new time point. A few authors have made use of this for time series RL.

Malhotra et al. (2016) proposed using an RNN in an encoder-decoder framework to learn the normal behavior of the time series, making the hidden state of the RNN a latent representation of it [113]. The RNN setup also allowed the authors to encode sequences of different lengths. Figure 4.3 shows their framework. Malhotra et al. furthermore proposed using the reconstruction error as an anomaly measure and showed promising results in detecting anomalies on a range of datasets. In 2017, the proposed idea was further developed and turned into a generic and off-the-shelf pre-trained feature extractor by training their model on a diverse set of time series datasets [114]. The authors claim that the latent representations resulting after applying their feature extractor to any time series dataset will significantly enhance the downstream tasks such as classification.

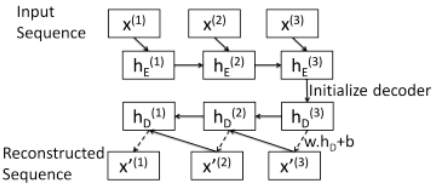


Figure 4.3: Taken from [113]. Illustration of the encoder-decoder architecture using an RNN. The model is using the input $\{x^{(1)}, x^{(2)}, x^{(3)}\}$ to predict of reconstruct $\{x'^{(1)}x'^{(2)}x'^{(3)}\}$.

Other authors made use of different encoders to extract the latent representation. Hyvärinen & Morioka (2016) simply used an MLP (a simple artificial NN) along with a multinomial linear regression to encode the time series [80]. The input time series would be divided into chunks each given a label corresponding to the chunk segment index. The MLP would be trained to encode a latent representation that could be used to classify each of these chunks according to this segment index.

Lei et al. (2019) used matrix factorizations to encode the time series by making sure that the Dynamix Time Warping (DTW) distances between any two different time series stay close [100]. That is, minimizing:

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times d}} \|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^{\top}\|_F^2 \tag{4.1}$$

where $\boldsymbol{A}$ is the $n \times n$ DTW distance matrix between the individual time series in the dataset and $\| \bullet \|_F$ is the Frobenius norm. The authors utilize that this problem has a closed-form solution to derive the latent representation of length d that can be chosen small. Since the DTW algorithm can calculate distances between any two time series of equal or unequal length, the method similarly works on time series of arbitrary length. However, as the method is intended to be used for clustering and is designed to preserve distances, not much effort is put into making the latent representations interpretable.

4.2.2 Convolutional Neural Networks for Feature Extraction

Recently, CNNs have been shown to be state-of-the-art in many different tasks related to time series and they have also found their way into the task of extracting latent representations from them.

Franceschi et al. proposed using *casual dilated convolutions* to learn scalable representations of multivariate time series [64]. The authors utilize the fast

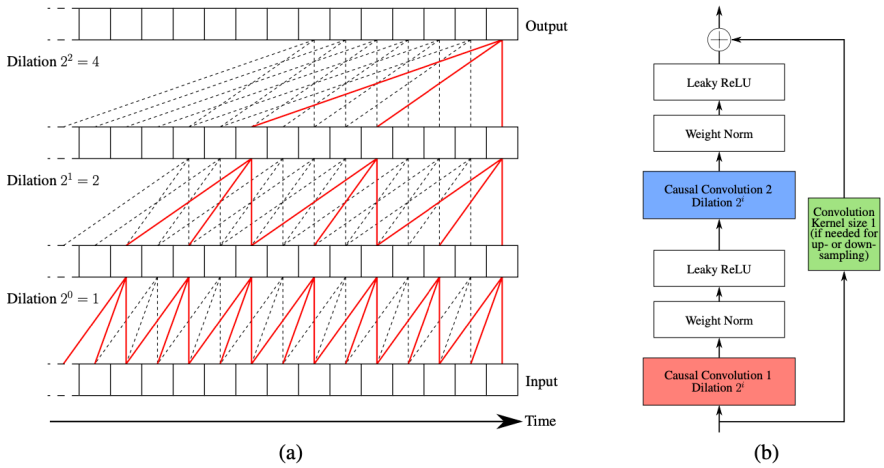


Figure 4.4: Illustration of the encoder proposed by Franceschi et al. in [64]. The algorithm consists of a sequence of convolutional blocks (b) in which the same dilation parameter is used in two 1D convolutions, each followed by weight normalization and leaky ReLU. The dilation parameter will increase exponentially with the depth of the network. The convolutions are designed to always output the same length by adding appropriate padding to learn representations given prior periods of varying lengths.

evaluation of the convolution to learn the representations and make the convolutions *causal* by never including future time steps in the evaluation of a convolution. The proposed encoder is illustrated in Figure 4.4 and consists of a number of convolutional blocks as depicted in (b). The dilation parameter is increased exponentially with the depth of the encoder (the number of consecutive convolutional blocks) to have convolutions looking at prior periods of increasing lengths and increasingly universal. Franceschi et al. propose to train the model contrastively using a novel triplet-loss scheme inspired by that of Word2Vec [120]. They make triplets by taking a random sub-series from a given time series to be the reference. They then produce a positive sample by sampling a sub-series from within the reference, and a negative sample by taking a (set of) random sub-series from a random time series in the dataset including itself, if it is long enough and does not have a stationary behavior. The process of producing triplets is illustrated in Figure 4.5. The authors claim that their algorithm learns representations that are *"universal and easy to make use of"* due to the random nature of the sampling strategy while showing promising results in time series segmentation by clustering the latent representations of the electricity consumption of a house into meaningful periods.

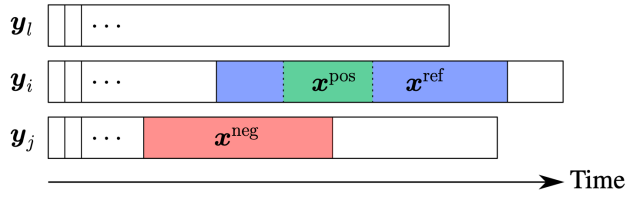


Figure 4.5: Illustration of the strategy used by Franceschi et al. to produce triplets for learning contrastive representations of time series [64]. The blue sub-series is the reference, while the green (a sub-series of the blue) is a positive sample, and the red sub-series is a negative sample. The triplet loss used in the algorithm will strive towards giving the green and the blue similar representations, while the red and the blue will have dissimilar representations.

Eldele et al. proposed learning representations using one-dimensional convolutions by forcing the learned representations to be contrastive both in terms of time and context [53]. In other words, the authors wanted the representations to be robust in time, and discriminative in terms of which time series a given representation stems from. These assumptions are good if the time series is rather stationary (a single outcome for the whole series). The authors proposed to make the representations discriminative and robust in time by making two different augmentations of the same input; a weak augmentation that is simply scaling and adding some noise, and a strong augmentation in which they divide the time series and make a random permutation of it. After encoding each of these augmentations using the same encoder, a transformer is used to compute a context vector. These context vectors are subsequently each used to predict the encoding of the other augmentation while being forced to be similar to each other. The whole process is visualized in Figure 4.6. Later, the same authors extended this approach to be class-aware thus enabling self-supervised training [52].

Challu et al. (2022) proposed using hierarchical latent variables for their deep generative model to be used for time series [27]. The authors would divide the time series into chunks of equal lengths and tie an element of the latent variable to a number of consecutive chunks based on its position in the hierarchy. This means that the element with the lowest position would typically be tied to just one chunk, while higher-positioned elements would be tied to a longer sequence of chunks (see Figure 4.7 for an example). These latent variables would first be sent through a *state model*, F, and subsequently, a *generator model* based on one-dimensional convolutions that would reconstruct the chunk of the time series. During training, Monte Carlo Markov Chain (MCMC) sampling is applied to

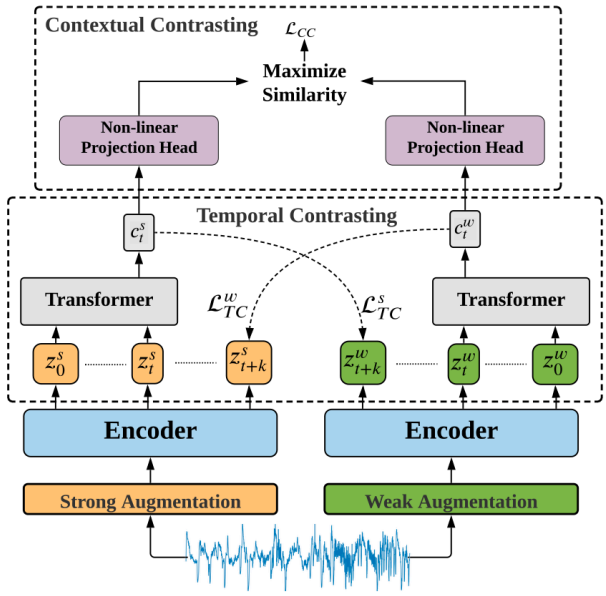


Figure 4.6: The architecture proposed by Eldele et al. in [53] to learn time series representations that are discriminative and robust in time. The encoder is a sequence of simple 1D convolutions. In the temporal contrasting module, each of the embeddings of the augmentations will be used to calculate a context vector c_t^s by applying a transformer layer. These context vectors are used to attempt to predict the embedding of the other. The context vector is also sent through a non-linear projection head, maximizing the similarity between contexts from the same time series.

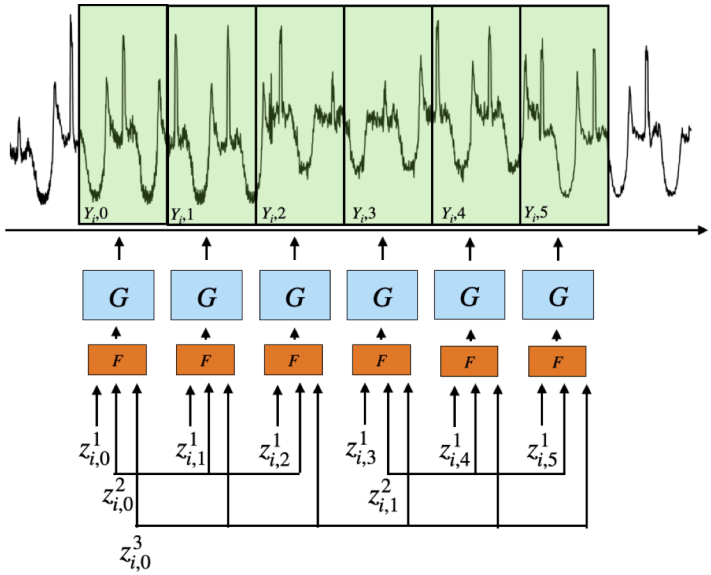


Figure 4.7: Taken from [27]. Illustration of how the elements of the latent vectors $\boldsymbol{z}_i^l$ are tied to a number of chunks of the time series Y_i with l being the level in the hierarchy. In this example, the elements of the first level are each tied to only one chunk, while the third level element is tied to all six chunks shown in the example. F is the *State model* and G is the *Generator model*.

find the optimal set of latent variables describing the data. While showing promising results, the authors claim that their approach has several advantages over existing methods with shorter training times, superior performance, and being more robust to missing values.

4.2.3 Interpretable Latent Representations

As with some of the methods mentioned in the previous section, latent representations are often allowed to live in the continuous space thus being hard to interpret. Some authors have attempted to make latent representations that are interpretable by humans requiring the representations to be discrete.

Van den Oord et al. proposed a framework for learning discrete representations of the dataset based on vector quantization. They would do this by first defining a set of embeddings $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_K$ that are allowed to live in the continuous space

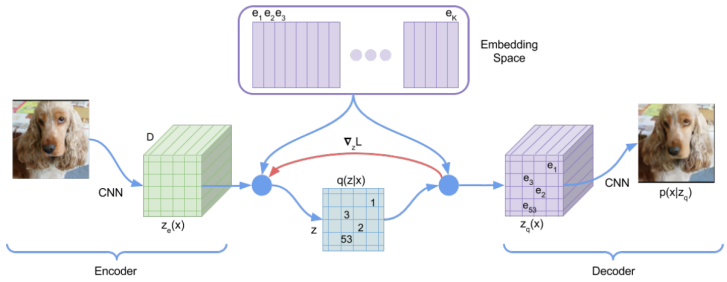


Figure 4.8: Taken from [188]. Illustration of the framework proposed by Van den Oord. et al. based on a VAE where the encoded value of the input will be linked to the nearest latent variable in the embedding space. This embedding will then be used to reconstruct the input. As illustrated by the red arrow, gradients are simply copied from the embedding $z_q(x)$ to the encoding $z_e(x)$ during training.

and each corresponds to a latent variable. They then apply a VAE in which the encoder will be trained to link an input to one of these latent variables determined by the nearest neighbor in the embedding space:

$$z_q(x) = e_k \qquad \text{where} \qquad k = \arg\min_j \|z_e(x) - e_j\|_2 \tag{4.2}$$

The decoder will be trained to reconstruct the input using the embedding of the corresponding latent variable. Notice, that the values of the embedding vectors are also updated during training. The framework is illustrated in Figure 4.8. The authors additionally show promising results on audio and video data and claim that long-term information can be preserved in the latent variables.

Fortuin et al. proposed learning discrete, temporal representations of time series by combining VAEs, Self-Organizing Maps (SOMs), and probabilistic models [63]. The authors use SOMs to discretely approximate the embeddings given by the encoder which can be chosen arbitrarily. They then use a Markov model to approximate the transition probability from one discrete representation to another in the latent space to learn smooth representations in time. In the end, a decoder will be used to reconstruct the input. The whole process is illustrated in Figure 4.9.

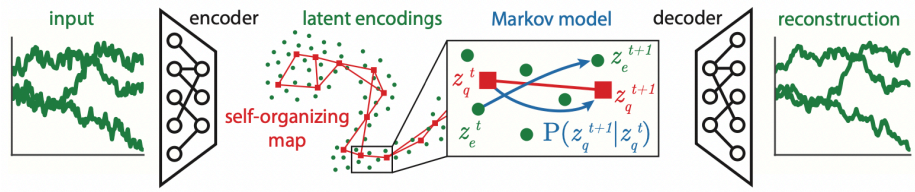


Figure 4.9: Illustration of the algorithm proposed by Fortuin et al. in [63]. The proposed algorithm makes use of the encoder-decoder structure from VAEs, the ability to smoothly transform continuous variables into discrete from SOMs, and Markov models to represent transitions in time. In the example in the Markov model, the output of the encoder z_e^t will have embedding z_q^t because it is the closest SOM node. The next time-point has encoding z_e^{t+1} and embedding z_q^{t+1}. The Markov model will be trained to accurately reflect the transition from z_q^t to z_q^{t+1}

Chapter 5

Data

The data used in this thesis is mainly based on the real-world network of TDC NET. In the network several performance metrics are polled approximately every 15 minutes[1] as described in section 2.2.2.3. Additionally, identification and topology data have also been made available, linking each modem to a position geographically and in terms of network connectivity (topology). In this section, an overview of the data made available to this project will be provided and the preprocessing steps used for time series data will be explained in detail.

[1] Other ISPs or network owners might use different adjourn times

5.1 Data Overview

Datasets that have been made available by TDC NET for this research are extensive and include multiple different data modalities. The following provides a brief overview of the data made available to this research.

Static reference data
Data that is not assumed to change[2], such as identifiers, and how the different components are related to each other at a given time. Variables include:

- Location (coordinates) of devices in the network (customer modems, network amplifiers or splitters, CMCs)
- Identifiers for the components (e.g. Media Access Control (MAC)-addresses for modems and identifiers for amplifiers or CMCs)
- Topology data
  - To which CMC the modem is connected
  - The sequence of amplifiers that connects the modem to its CMC

Time series data
Different metrics from each of the Customer Modems (CMs) were measured remotely approximately every 15 minutes due to the vast amount of data gathered in the network. Some metrics are reported as a statistic (e.g. minimum or maximum) during the last poll cycle for a given channel (corresponding to a range of transmission frequencies), while others are ever-increasing counts of specific events. Many of the parameters are calculated independently for the DS and US signals, hence the data includes performance metrics on both types of signal. Some of the metrics gathered from the modems are:

- `Modulation Error Ratio (MER)` – Ability of the receiving end (modem or CMC to demodulate the signal
- `Signal-to-Noise Ratio (SNR)` – How good the path between the two ends is at transmitting a signal
- `Package loss rate` – How many of the transmitted packages were lost during transmission

[2]It could, however, change when customers move to a different house, cancel their subscription, and so on.

- `Codeword errors` – How many of the codewords received had to be corrected (by error correcting codes)
- `Jitter` – The variation of how long it takes a packet to be transmitted

The full list of metrics used in this project along with explanations is given in section 8.A. Figure 8.2.1 shows an example of some of the time series metrics plotted in time.

It is important to mention, that far from all the modems in the network have both time series and topology data provided. For around 1/3 of the modems in the network of TDC NET, the complete sequence of amplifiers connecting the modem to the CMC is not known. To properly train the various models, effort has been put into securing a high quality of the data that was extracted, which includes extracting only data and topology data from modems that have both available. This is under the assumption that a subset of the modems connected to a CMC will themselves constitute a network. In other words, removing some of the modems from a network does not affect the remaining modems significantly thus analysis can still be carried out.

5.2 Time Series Preprocessing

Time series data gathered from multiple sensors or modems, as in our case, pose a few challenges that must be addressed before they can be effectively analyzed and compared.

5.2.1 Aligning Time Points

All observations include a precise time at which it was polled. Different sensors, though connected through the same network, are not always polled at the same time meaning that a scheme for aligning the time points must be deployed to make them directly comparable. Additionally, though modems are expected to be polled with exactly 15 minutes between this is not always the case.

We align the time points by assuming that there are exactly 15 minutes between each poll and that all devices in the network (be that modem or CMC) are polled simultaneously. This enables us to generate a new sequence of time points with exactly 15 minutes between each point. Now it remains to align this new sequence with the old time points in an optimal way to be used for subsequent

interpolation of the observations. Acknowledging that the CMC measurements are likely to be compared to the measurements for all of the modems, we align the new time point sequence to that of the CMC.

Let $\boldsymbol{t} = [t_0, t_1, \ldots, t_T]$ be the sequence of time points at which the CMC has been polled. We then calculate the distances from t_0 to each of the time points in the vector:

$$\boldsymbol{t}_d = \boldsymbol{t} - t_0 \tag{5.1}$$

We then calculate the modulo of each of these distances concerning the 15 minutes we assume are between each:

$$\boldsymbol{t}_r \;=\; \boldsymbol{t}_d \mod 15\,\text{minutes} \tag{5.2}$$

The remainder vector is centralized by making sure to have no values outside the range from $\frac{-15}{2}$ to $\frac{15}{2}$ by defining the centralized vector $\boldsymbol{t}'_r$ with elements:

$$t'_{r_i} = \begin{cases} t_{r_i} & \text{if } t_{r_i} > \frac{15}{2} \\ t_{r_i} - 15 & \text{otherwise} \end{cases} \tag{5.3}$$

We then calculate the median of these deviations to get the overall offset that is optimal for our new time point sequence:

$$\Delta = \text{med}(\boldsymbol{t}'_r) \tag{5.4}$$

This allows us to calculate the optimal time point sequence to be used for interpolation:

$$\boldsymbol{t}^* = \left\{ t_0 + \Delta + i \cdot 15 \right\}_{i=0}^{T} \tag{5.5}$$

Given a time series with values $x_0, x_1, \ldots, x_T$ with corresponding time points $\tau_0, \tau_2, \ldots, \tau_T$. We define the linear interpolator to be the piecewise linear function given by:

$$f(t) = \begin{cases} x_0 + (t - \tau_0)\frac{x_1 - x_0}{\tau_1 - \tau_0} & \text{if } t \leq \tau_1 \\ x_1 + (t - \tau_1)\frac{x_2 - x_1}{\tau_2 - \tau_1} & \text{if } \tau_1 < t \leq \tau_2 \\ \quad \vdots & \\ x_{T-1} + (t - \tau_{T-1})\frac{x_T - x_{T-1}}{\tau_T - \tau_{T-1}} & \text{if } t < \tau_{T-1} \end{cases} \tag{5.6}$$

This means that the interpolated vector used for further analysis is simply this function evaluated for every value in $\boldsymbol{t}^*$:

$$[f(t_0^*), f(t_1^*), \ldots, f(t_T^*)] \tag{5.7}$$

With corresponding time points in $\boldsymbol{t}^*$. An illustration of the time point alignment algorithm is shown in Figure 7.5.

5.2.2 Missing Time Points

Since the whole network consists of a large number of modems it is inevitable that some of the modems have missing time points momentarily or during long periods. This could be due to many reasons like the modem having been turned off by the user, impaired connections too bad for any transmission, or damaged cabling due to construction works. For the deep NN models to properly work, these need to be imputed.

Many ways of imputing missing observations in a time series exist. A subset of these has been thoroughly analyzed and compared with respect to their performance. This was done by manually removing random time windows of varying lengths from a set of time series with no missing values. Among the tested methods were Akima Spline interpolation [3], and ordinary linear interpolation of which the latter showed the best results across all different lengths of missing windows of observations, hence is chosen to be the method of interpolation used in this project. We apply a method similar to the one presented in the previous section making the interpolator on the known points and evaluating it at the missing time points inferred from the assumption of 15 minutes between time points.

Part II

Research Outcomes

Chapter 6

Summary and Perspective of Research

In this chapter, a general summary and discussion of the research carried out as part of this thesis will be given. Though all of the work concerns itself with extracting useful information from multiple time series, this has been done with one overall goal in mind, namely anomaly detection or localization of faults in the HFC network. Meanwhile, consulting the literature and investigating the state of the art in network monitoring it is clear that many authors list a fully known and updated network topology as a requirement for proper maintenance or anomaly localization [97, 174, 71, 173, 77]. Remote localization is especially important for optimal routing which could help prevent unnecessary driving which is one of the overall goals of our project. For these reasons, another subgoal of this research has been to infer the missing topology of the HFC network using monitoring data from the customer nodes. Meanwhile, for the work carried out in the other papers we have assumed that the topology of the network is fully known and up to date, making us able to utilize it in the model to localize the fault.

In section 6.1 papers A and B will be introduced and discussed. Each of these deal with a different way of detecting faults in broadband networks. In section 6.2 paper C will be discussed which deals with the extraction of useful events for topology reconstruction in rooted trees.

6.1 Anomaly detection in time series

As described in section 4.1, many of the methods developed to detect and classify errors in the HFC network are based on FBC data making them great tools for technicians to use in the field but which are not practical for remote detection due to the immense amount of equipment needed to gather the data and the space needed to store it. Therefore methods for detecting anomalies using the aggregated time series data explained in chapter 5 will be of great value and thereby the main aim of this thesis. We propose two different approaches to anomaly detection in HFC networks based on, respectively, a supervised method trained on a dataset labeled by domain experts and an unsupervised method based on NFs which is a recent framework for deep generative modeling used for accurate and tractable modeling of densities of complex distributions.

6.1.1 The supervised approach

One of the most common errors occurring in HFC networks is the so-called CPD which is the product of an impaired connection where rust or a loose connector prevents the signal from traversing freely causing noise in the signal. This type of error is generally well-studied and well-known enabling the creation of a manually labeled dataset [164].

Paper A – SeePD: Detecting Common Path Distortion Faults in Broadband Networks

In cooperation with domain experts, we propose the Broadband Common Path Distortion dataset (BoCPaD) consisting of 655 observations where each observation consists of the time series from a set of modems located topologically beneath a given amplifier and a label denoting the presence or absence of CPD at the given amplifier (topology is assumed known). We investigate different approaches to performing feature engineering on the observations, converting the (different amount of) time series into a fixed number of features for subsequent modeling. We investigate different ML methods and report their performance in distinguishing between the presence and absence of CPD. For the optimal model, we report the feature importance making us able to propose a simple 'business rule' based on thresholding on the two most important features. This proposed business rule enables ISPs to choose between the better-performing, but unintuitive to interpret model or the model that is easier to implement and interpret while reaching similar performance.

The proposed dataset is limited in size but it is believed to be representative of the general behavior concerning CPD. The dataset ends up showing great potential for modeling the distinction between the presence and absence of CPD meaning that gathering a bigger dataset in the same way or gathering a dataset using an HFC laboratory in which all parameters and traffic could be fully controlled would be of tremendous value for ISPs. This could also allow for modeling the effect of CPD of various degrees while investigating the stationary network parameters such as the number of modems connected to the local node or the length of the cables.

6.1.2 The Unsupervised Approach using Normalizing Flows for Multivariate Time Series

Even though some of the most frequent errors occurring in HFC networks like CPD examined in the above-mentioned paper are well studied, not much is known about how other types of errors manifest themselves in the remotely monitored data to be used in this project. This type of data is abundant with data points including a long range of metrics being gathered at every single online modem in the network every time data is polled. In the case of TDC NET that amounts to around 58 mio. observations of each recorded metric every single day. However, no accurate indication of the presence, respectively, absence of errors exists. TDC NET does have records of the faults reported by customers that lead to a subsequent technician dispatch but as explained in section 4.1 this is not believed to be a precise measure for the presence of errors. One idea could be to simply use the trouble tickets that TDC NET does have in a supervised manner, acknowledging that a good model would lead to a large number of false positives that were actually true positives. However, even though such a model would work well, we would have no way to verify that it truly works.

Instead of considering only the distinction between errors and non-errors, it would be more valuable to quantize abnormal behavior and try to understand the underlying reason for this behavior. This could e.g. be achieved by clustering different instances of abnormal behavior that show similar trends. These clusters could subsequently be related to actual root causes in the network in cooperation with domain experts. Estimating the distribution of time series behavior is a very recent problem, hence not much research exists. However, two recent papers both make use of NFs to estimate the density of time series behavior. Dai & Chen made use of graph convolutions and an RNN to summarize the historical and inter-sensor contexts and use that to estimate the conditional density of a single observation using an NF [39]. Guan et al. also estimated the conditional density of a single observation using an NF but this time using an aggregation

of historic data and a positional encoding as the conditional [68]. Even though both of these methods achieve promising results, the conditional nature makes post-hoc analysis of the underlying errors hard.

Paper B – Anomaly Detection in Broadband Networks: Using Normalizing Flows for Multivariate Time Series

In paper B we estimate the density of a window of a multivariate time series by first producing a latent vector representation of it using an AE. We assume that it is sufficient to learn an AE once and use its resulting latent variables as accurate representations of the data. We base the AE on one-dimensional convolutions in order to preserve the context in the embeddings. Using an AE instead of a Variational AE allows the latent representations to constitute a more accurate distribution that is not forced to be continuous like it is in the variational case. The density of this resulting complex and non-continuous distribution is then estimated using an NF in an appropriate way. This two-phase approach means that we have three different representations of the input time series data:

1. The original time series window of size $\omega \times \Delta$ where ω is the length of the window and Δ is the number of input parameters
2. The latent representation vector of size δ encoded using the AE where δ is the predetermined size of the latent representation. The density of these vectors is estimated using the NF.
3. The normalized representation vector of size δ which is the corresponding value after the above latent vector has been transformed (backward) through the sequence of functions of the NF. These vectors will follow a simple δ-dimensional Gaussian distribution and, hence have densities accordingly.

The paper presents very promising results, demonstrating that the learned density of the latent representations (the second representation in the list) is a good indicator of outlyingness. The paper furthermore proposes an algorithm based on bootstrapping for linking densities learned by the NF, which are often hard to interpret due to the discontinuity in the space, to a p-value by estimating the Cumulative Distribution Function (CDF) of the one-dimensional distribution of densities. Given that the NF learned an accurate representation of the distribution, this p-value estimates the probability of observing the given observation or an observation that is less likely thereby enabling interpretation of the density in a statistical context.

6.1.2.1 Learning Only the Systematic Behavior

Since the NF is modeled on the data including outliers and it is not known which observations are outliers, it is assumed that outliers are few in number and hence fall into low-density regions of the learned distribution. If on the other hand, one wishes to model the distribution of only the systematic behavior, it means that the outliers will be modeled as Out-Of-Distribution (OOD) points. In paper B we propose to do this by taking advantage of the two-phase nature of the algorithm which means that we can learn the NF, constituting the second phase, multiple times but weighing the observations differently each time. We propose an algorithm that iteratively down-weighs observations that fall in low-density zones thereby slowly excluding the unlikely observations. The paper presents promising results by showing that the normalized representation (third representation in the above list) of the outlying observations is pushed away from the high-density bell curve (that follows a Gaussian distribution) in the center, making the Mahalanobis distance in the normalized space an increasingly better predictor for outlyingness. Furthermore, the observations that are pushed away from the bell curve in the center seem to arrange themselves in clusters along with other observations showing similar behavior. These clusters can subsequently be related to an underlying problem that, in cooperation with domain experts, can be connected to an actual root cause.

All in all this work shows great potential but is still in the preliminary phase. Future work entails proving the method on a high-quality multivariate time series dataset with underlying root causes that are well-understood. Additionally, more complex models can be employed in both phases of the framework. For example, an enhancement to the model could involve using an autoencoder (AE) that learns representations across different scales. This would allow behaviors of varying lengths or characteristics to be encoded into the latent representation.

6.2 The missing topology

Although many authors working with HFC networks list a fully known and updated topology as a requirement for proper maintenance of the network, not much literature exists that deals with this problem. According to domain experts at TDC NET, it has previously been possible to modify the signal transmitted to a group of modems for a short period of time by e.g. dampening the amount of amplification for a specific amplifier and then observing which modems were affected by this modification. This is, however, not believed to be feasible using the newest technology (DOCSIS 3.1), as modems are vulnerable

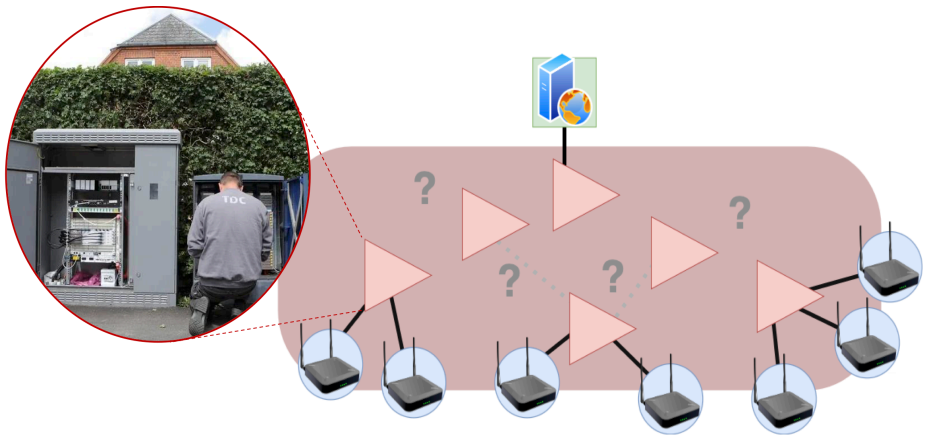


Figure 6.1: Illustration of the topology reconstruction problem. Blue circles are customer modems, the green square is the CMC, and the red triangles are the amplifiers (and usually splitters) sitting in street cabinets. It is usual for network owners to know all the components of their HFC network, but often not the complete mapping of the connections.

to even small fluctuations in the power of the signal and might cause connection breakdown, and because the PNM filters have become very good at dealing with impaired signals (see section 2.2.2.3). This means that the inference of network topology continues to be problematic and that a solution considering only time series data gathered by each of the customer modems (leaf nodes) would be valuable. The problem is visualized in Figure 6.1.

6.2.1 Borrowing from Biology

Later collaborators Sørensen & Pisinger developed a method for inferring the topology of an HFC network using both geographical coordinates and discrete data points gathered at the customer level [138]. For each time point, they considered discrete data points in an alphabet, Ω, and gave each character (or state) in the alphabet an interpretation by assuming different events could occur along the network edges or amplifiers. By considering these series of discrete variables coded sequences like that of DNA in different species, it allowed them to use the so-called *parsimony* score as an optimality criterion while searching for the optimal tree or the topology that best explains the data sequences. The parsimony score is a method originally intended for inferring phylogenetic trees in biology by counting the number of mutations of an original (shared ancestor) sequence that would be needed to explain the states at the leaf level. This

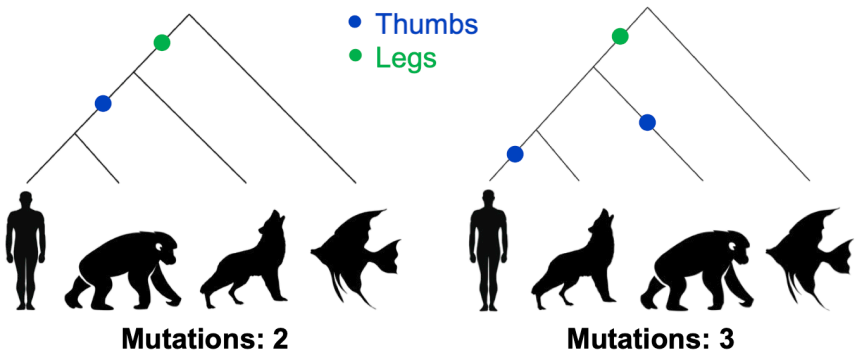


Figure 6.2: Two candidates for phylogenetic trees explaining the ancestral evolution of four species. The genes that code for traits, including having legs and having thumbs, arose as mutations in earlier species. By counting the number of mutations happening on each of the trees, one would assume that the left tree is the most likely, as it would be unlikely that two different species developed thumbs independently.

concept is illustrated in Figure 6.2. To demonstrate their method for inference of the missing topology in HFC networks, Sørensen & Pisinger simulated binary data series. In this simulated data, a value of 0 would imply that no event was affecting a given customer at a given time, while a value of 1 would imply the contrary. For each time point, events were simulated randomly on the network edges and assumed to affect all customers topologically beneath it equally. Thus, they did not prove their method on continuous time series data - a step that would be critical for the deployment in real-world networks.

6.2.2 Considering Multiple Different Topologies

In the attempt to overcome the challenge of encoding continuous data into sequences of discrete events (a seq2seq - sequence-to-sequence model), a tempting idea is to simply train a model using the parsimony score as a loss function. Since the goal of the algorithm developed by Sørensen & Pisinger is to find the tree topology that requires the least mutations to explain the leaf data, it seems logical to minimize the parsimony score during training. However, this approach poses a problem since data sequences in which every data point has the same value would not require any mutations to explain. This means that the trivial (zero-encoder) would be learned; i.e. the model that encodes everything to the same discrete value. Reconsidering the aim of the parsimony algorithm it becomes clear that the objective is to calculate a value for a tree *in relation to*

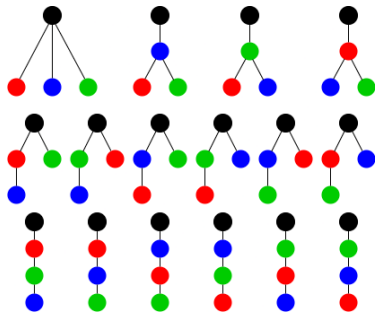


Figure 6.3: The 16 different permutations of a rooted tree with four internal nodes taken from [147].

other trees in order to arrive at the optimal one. This means that to extract significant events with respect to the parsimony score, one would need to consider a set of potential trees and how the parsimony score treats these differently.

Paper C – Topology Reconstruction in Telecommunication Networks: Embedding Operations Research within Deep Learning

In paper C, we investigate the potential of using a contrastive approach to learn the discrete event encoder, acknowledging that multiple different topologies must be compared in the loss function.

In this pilot study, we investigate networks of relatively small sizes. This enabled us to easily consider the entire set of true topologies, as there are for instance only 16 possible ones when considering a tree with four internal nodes (of which one is the CMC), and, for the smallest trees, to incorporate all of them into the loss function, while still maintaining fast training. See Figure 6.3 for the different permutations of a tree with four nodes.

We proposed to train the encoder using a Siamese network [91], meaning that the time series from each of the individual modems will be encoded using *the same encoder model* when turning the continuous time series into sequences of embeddings. We then use the parsimony algorithm to calculate the number of mutations needed to explain this encoded data for a set of potential topologies individually. This enabled us to propose a contrastive loss function with two terms: one being the minimization of the parsimony score when calculated using the *true topology* (the one used to sample the events in the input time series), and the other being the maximization of the differences between parsimony scores calculated using any two different topologies.

While this initially showed great potential and good results, we also identified further problems to be solved and generalizations to be investigated before being able to fully ascertain the feasibility of the approach in real networks. For instance, it was found that the original parsimony algorithm proposed by Hartigan in 1973 [69], was not guaranteed to yield *uniquely* optimal solutions. In some cases, a topology different from the true topology would, in all cases, give the exact same score as the true one, making them indistinguishable from a parsimony perspective.

6.2.3 Training a Discrete Event Encoder

To adequately confirm the feasibility of training an encoder able to transform continuous modem data into (in this case) binary sequences based on the parsimony principle, several things need to be investigated:

- How can we guarantee unique solutions using the parsimony principle?
- How can Hartigan's algorithm [69] be changed to guarantee uniquely best solutions?
- How do we ensure proper calculations of gradients through the parsimony algorithm?
- How do different data characteristics affect the performance of our approach?
- How well does the approach generalize to be used on trees of various sizes?

Paper C provides answers to all of the above questions by demonstrating the feasibility of training a sequence-to-sequence encoder able to extract relevant encodings to be used in the algorithm developed by Sørensen & Pisinger in [138]. It achieves high accuracy in both end goals: minimizing the parsimony score among all possible topologies for the true topology (i.e. the one from which the data was sampled) and reconstructing the underlying sampled events. In the process, however, the paper also has implications that reach far beyond these concrete goals and beyond what was initially believed.

6.2.4 Embedding Operations Research in Deep Learning

Although every ML problem is in reality an optimization problem in which an objective (the loss function) is optimized with respect to some decision vari-

ables (weights), we believe to be the first to embed an optimization algorithm directly within a deep learning loss function. This means that every time the loss function is called the optimal solution to an optimization problem needs to be found. The parsimony score of a tree is, in fact, an optimization problem because it is exactly the optimal way in which a base signal from the root of the tree is evolved through the branches using the least amount of mutations to the data sequence.

Having made a slight modification to the tree reconstruction problem in which an interpretation of one of the states of the model is given, we also provide a novel algorithm that is proved to provide the optimal solution with regard to this updated formulation. This algorithm is believed to ensure solutions that are both optimal and unique. This means that no topology different from the true topology will give parsimony scores equal to or better than that of the true topology. This claim cannot be proved for trees of all sizes but is formulated as a conjecture that is left for others to potentially prove in future work. Even though the conjecture can only be proved for trees with up to and including six internal nodes by trying every single combination, a series of experiments are conducted in which a sophisticated topology sampling algorithm is used to cover the space of possible topologies to look for cases disproving the conjecture.

A second modification to the algorithm considers not discrete data states, but probabilities for each of the data states. While this solves the problem of discontinuity when calculating the loss, it also has further implications since it could be used in cases where the data states cannot be determined precisely but are affiliated with some uncertainty.

6.3 Conclusion

The research carried out in this project demonstrates different ways in which time series data can be used to extract relevant information. We show that value can be created regardless of whether one already has a strong understanding of the time series and their inherent faults, whether the goal of the analysis is to gain insights into the problem, or even when one is dealing with a higher-level issue without a specific focus within the time series.

In paper A, we show that a good understanding of the physical properties of certain faults makes a good foundation for training a model to detect that very problem. We show promising results for detecting CPD using both feature extraction and simple thresholding learned by a supervised model. In paper B, we show that abnormal behavior can be quantified and detected in low-density

zones of the distribution of time series behavior while showing that it is possible to exclude abnormal behavior in such a way that they fall into clusters from which underlying root causes can be identified. Finally in paper C, we showed that relevant time series events can be extracted with a higher-level goal in mind. We show that it is possible to use an optimization problem directly in the loss function toward the goal of reconstructing the topology of an HFC network in regions where it is not known.

Though all of the work shows great potential, it suffers from the data quality made available throughout this project making it somewhat preliminary. In the CPD case, a dataset of higher quality would be valuable in which various degrees of CPD could be controlled and tested to see how they manifest themselves in the aggregated time series data. For the unsupervised problem, a multivariate time series dataset with a number of well-known underlying root causes would enable an even more sound analysis of our proposed approach. For the same problem, an accurate ground truth of abnormal time points of the broadband dataset would enable a validation of the method to be used on that data. This is also evident for the topology reconstruction problem, where the missing ground truth forced us to simulate continuous time series data with known events to properly validate our approach instead of using actual data from the network.

All in all, a good understanding of the domain, the data, and the faults that are inherent therein is crucial when performing time series analysis. Unsupervised approaches can be used when a ground truth is unavailable, but they should not be used without careful consideration of intuition and the potential implications of the results. This means that high-quality datasets containing remotely gathered broadband data, accurate ground truths of errors, and labels for different types of errors would be immensely valuable to the ISPs that own and maintain HFC networks worldwide. These can be achieved both by letting domain experts and technicians examine the network closely and register errors that are found or by simulating a dataset in a laboratory where different faults and varying degrees thereof can be fully controlled.

While our studies are somewhat preliminary for the direct application to HFC networks, the results of this thesis show potential for future studies and implications that reach further than broadband networks. Being able to accurately estimate the density of a multivariate time series could be of great value in many fields where multivariate time series plays a role. Additionally, being able to cluster abnormal behavior and relate it to actual root causes in the modeled system would be of even greater value. Finally, embedding an optimization problem directly into a deep learning loss function could make a difference in fields where one does not know what specifically to look for when optimizing a higher-level problem. We show that these things are possible – even when the data quality is not optimal.

Chapter 7

Paper A
SeePD: Detecting Common Path Distortion Faults in Broadband Networks

Meadhbh Healy[a,*] · Tobias Engelhardt Rasmussen[a,*]
Tarun Kumar Maddimsetty[b] · Vamsi Krishna Vedantam[b] · Andreas Baum[a]
Thomas Martini Jørgensen[a]

[a] Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark
[b] TDC NET A/S, København C, Denmark
[*] Meadhbh Healy and Tobias Engelhardt Rasmussen, placed alphabetically, are the corresponding authors and contributed equally to this work

Publication Status: Paper is submitted and under peer-review for publication in *The Journal of Signal Processing*

Abstract

Common Path Distortion (CPD) is an extensive issue affecting Hybrid-Fiber Coaxial (HFC) Networks. Although the problem has a direct effect on the customer experience, there is no decisive method of remotely detecting CPD in an HFC network. Devices such as spectrum analyzers can be used to identify CPD at source, however, these are costly and inefficient to deploy. Proactive Network Maintenance (PNM) metrics provide an insight into the status of the network but there is limited knowledge on the manifestation of CPD.

We introduce BoCPaD (Broadband Common Path Distortion dataset); a novel CPD-detection dataset consisting of 655 manually labeled observations from the HFC network of TDC NET spanning most of Denmark. To precisely identify CPD faults, we present two feature-engineering schemes based on firstly, the binning, and, secondly, the utilization of distributional moments of specified, relevant variables. These are subsequently modeled using a variety of machine learning (ML) models. An explainability technique known as Shapley values is implemented on the highest-performing model to ascertain the most influential parameters. We use this analysis to propose an intuitive business rule based on two-layer thresholding of significant features.

We find that the two proposed feature-engineering approaches each allow for accurate modeling of CPD, but also that the simpler, intuitive business rule achieves comparable results using far fewer estimated parameters. Our intuitive business rule allows internet service providers to choose a model that is easy to interpret and implement while still achieving results comparable to state-of-the-art ML models using our feature-engineering approaches.

7.1 Introduction

A Hybrid Fiber-Coaxial network provides broadband connectivity directly to the subscribers on its network. The architecture comprises both fiber optic cabling and coaxial cabling to transfer signals from the common root node to its customers and vice versa. The network consists of independent channels (carrier

frequencies) for both the upstream (US) and downstream (DS) signals[1].

Degradation can occur in the signal which causes an individual customer or multiple customers to lose or have significantly reduced connectivity [199]. One of the most prolific problems that cause network breakdown is called Common Path Distortion (CPD). CPD is an impairment that occurs in two-way cable systems where signal leakage seeps into the US signal from its corresponding DS signal. As the problem materializes in the US signal, it generally causes multiple connectivity failures on the network at the same time [164].

Though CPD is a common problem, there is no shared consensus on how best to detect it remotely in an operating network. The goal of this paper is to 1) introduce a labeled dataset based on remotely acquired parameters that can be used to detect cases of CPD, and 2) examine the performance of different models in detecting CPD using the obtained dataset.

The data used for this research has been provided by TDC NET, Denmark 's largest internet service provider [180]. TDC NET supplies approximately 600,000 customers throughout their HFC network spanning the whole of Denmark. Domain experts at TDC NET estimate this issue affects roughly 50,000 customers on their network per year. CPD can be difficult to distinguish from other noise problems in the US signal as it generally manifests itself as an increase of noise in the signal.

In section 7.2 we explain the background of the HFC networks and the data generated therein. We present current state-of-the-art and our novel contributions in section 7.3, explain the proposed data generation scheme in section 7.4, and finally the model in section 7.5. We evaluate our proposed method and data set in section 7.6 followed by a discussion in section 7.7.

7.2 Background and Data

7.2.1 Cable Network Architecture

Coaxial cables are a type of electrical cable consisting of an inner copper conductor with an insulator shield surrounding it. Additionally, a braided metal mesh prevents external interference, since coaxial cables can be prone to radio frequency (RF) interference. See Figure 7.2 for an example of both ends of a

[1]The US direction of the HFC network is when data is transmitted from the individual customer modems up to the headend and the DS signal carries data in the opposite direction.

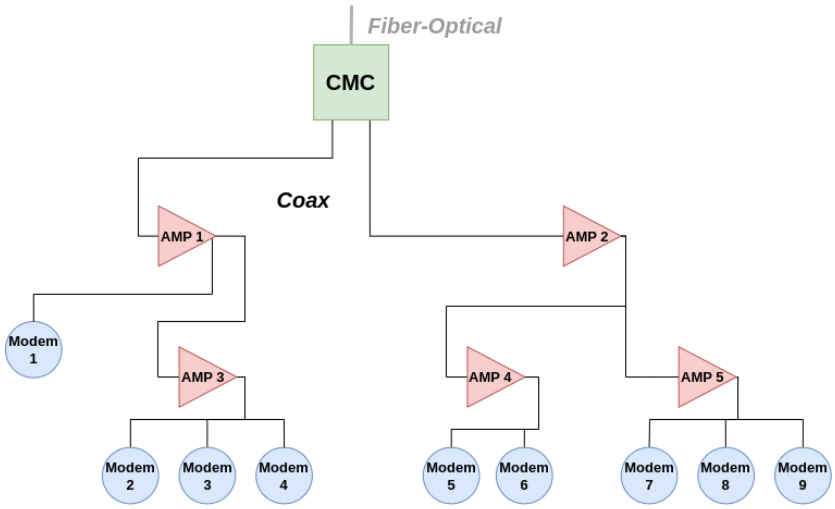


Figure 7.1: Overview of the Hybrid Fiber Coaxial (HFC) architecture.

coaxial cable. The cable broadband system is an access network, meaning that it transmits signals to all the subscribers on the network. A given coax network architecture resembles a tree (like a hierarchy) with the central root node as the root and the individual customers as leaves on the tree. The central root node may be hereafter referred to as the Coaxial Media Converter or CMC and the individual customers on the networks will be described as customer modems or CMs. The CMC is connected to the modems through a sequence of amplifiers which are linked in various ways and make up the topology of the coaxial network. Due to the CMC being connected to the backbone internet, the CMs provide internet access to the individual users on the network. All cabling and signals beyond the CMC are optical and are transformed to coaxial at the CMC. Therefore, each hierarchy of the coaxial part of the network can be treated as separate segment. An overview of a segment of a typical architecture of a coaxial network from CMC to individual CMs is visible in Figure 7.1. The equipment of a HFC network, which generally consists of amplifiers and splitters (splitting the signal) are placed inside street cabinets in the local area hosting the coaxial network.

In order to mitigate against possible faults that can arise in the HFC network, the cable industry has established standard data metrics which are collected to analyse the health of the network [184]. These metrics are a part of Proactive Network Maintenance (PNM). PNM metrics help improve the overall reliability of the networks by proactively detecting and fixing faults before they become an issue for customers [77, 203]. PNM metrics are collected at both the CMC and at

each CM. The amplifier devices which are components in the HFC structure are passive devices that implement non-linear RF amplification and PNM metrics are not collected at these points.

PNM data are routinely collected in HFC networks and has been recommended for network maintenance by the Data Over Cable Service Interface Specification (DOCSIS) since 2005 [203]. The most optimal way of using the data to improve network reliability, however, continues to be unresolved [77]. This is due in part to the the quantity of parameters available to technicians meaning that it is not easy to monitor on a constant basis, and partly due to the vast amount of different faults that can happen in the network. For instance weather conditions have shown to have an effect on network quality [195, 129]. Prior research has mainly focused on detecting arbitrary faults, rather than classifying specific sources of network degradation. The metrics collected can provide indicators of faults that may or have occurred in the system, they do not directly pinpoint any specific cause of network breakdown. Additionally, PNM data does not identify which device in the network is the root cause for a failure [71]. Technicians in the field are still obligated to manually detect faults which is both time consuming and prone to human error.

7.2.2 Dataset

In collaboration with TDC NET, three months of anonymized PNM data was obtained. The type of HFC network data being used in this research is Quadrature Amplitude Modulation or QAM. QAM is a type of signal in which the two carriers on the same frequency are shifted 90 degrees. As a result of this phase difference they are in quadrature which gives rise to the name [45]. As mentioned previously, the volume of PNM parameters is extensive. Both the CMC and each channel[2] of the CM collect similar metrics, separately for both US and DS, and new data is polled from each device in a raw format every fifteen minutes. Amongst the variables collected are: cable modem transmission power (Tx power), ratio of codewords that could not be corrected (uncorrectable ratio), ratio codewords that have been corrected (corrected ratio), signal to noise ratio (SNR) and modulation error ratio (MER). We have selected a subset of the listed variables based on their prevalence in existing research and on the knowledge of domain experts [203, 77], Therefore, we focus our research study on minimum CM MER (minimum MER during the fifteen minutes between polls) and US CMC SNR data. Both terms can be seen as analogous and refer to a measure that compares the level of desired signal to the level of unwanted

[2]The frequency band is divided into a set of non-overlapping channels that facilitate transmission of large amounts of data for both US and DS signals.

Figure 7.2: Image of corroded connector which results typically results in a CPD fault. Taken from [71].

noise, however, signal to noise ratio quantifies the baseband signal whereas MER refers to the information carrying part.

7.2.3 Common Path Distortion

Common Path Distortion (CPD) can be the result of a variety of root causes including stresses or corrosion on the connector elements which results in the creation of a non-linear diode junction[193]. The consequence is a non-linear distortion, where forward moving signals in the spectrum are pitched at new frequencies as they pass the source of the CPD. Historically, CPD manifested itself by recurring 6MHz spikes across the return spectrum, however, in QAM channels it results in an increased noise floor and is thus indistinguishable from other noise in a digital network[164]. An example of a corroded connector can be seen in Figure 7.2. As the US SNR metric at the CMC is the result of a hierarchical aggregation of signals from all the CMs and up, it follows that a fault which will initially affect only a single signal can rapidly propagate to multiple customer signals and in extreme cases destroy all connectivity under a single shared amplifier. In comparison, the DS MER metric measured at each CM is a result of the specific path of the signal from CMC through amplifiers to the customer, hence, DS faults can be traced to individual customer signals where they can be repaired by technicians. Due to technicians having the problem of not knowing where in the network the problem is located, the process of investigating and repairing the faults becomes extensive. Whilst the CMs and CMCs collect PNM parameters at both ends of the network, the amplifiers are passive devices and therefore do not collect data. therefore it is difficult to ascertain if the fault occurred near a customer's home or at another location on the network.

7.3 Overview

7.3.1 Existing Literature

To the best knowledge of the authors, previous research that specifically pertains to detecting CPD in HFC networks is very limited [194]. Currently, fault repair is dependent on customer calls, which can be an unreliable means of detection, since costumers are unlikely to call unless the connection is severe and/or network issues are prolonged. When the customer calls and a trouble ticket is logged, the repair technician will first go to the customer's home. However, the fault may not be located in or near the home and therefore the technician will have to conduct a binary method of moving through the network in order to pinpoint the fault location. This is problematic, both for the repair technician who is persisting with a time-consuming means of fault detection and the customers on the network who experience connectivity issues until the problem is fixed. Heiler *et. al* released a paper in 2022 which focused on detecting CPD by using the customer trouble tickets to label the PNM data [71]. TDC NET also collect customer trouble tickets which were made available to the authors. However, we found the TDC NET trouble tickets lacked the necessary information to make accurate labels. We also found that quite often faults are reported by customers which are unrelated to a network error and are instead due to an installation issue in the customer premises equipment (CPE). The converse is also true, with customers not logging when they have a problem. As described by Chen *et. al* in [33], the relationship between network failures and customer calls can be jumbled, as some customers may not call in the event of reduced connectivity whilst other calls will be dictated by the impact on the customer, the customers availability, and the time of day. Hu *et. al* also discuss the uncertainty of customer trouble tickets to determine network breakdown when developing their Cablemon tool[77]. In this paper, they use customer trouble tickets as hints to filter the data rather than as representing an absolute truth. However, the research focuses on sources of general network breakdown rather than detecting CPD specifically. In other research, such as the work by Simakovic *et. al*, a binary threshold is set on a parameter called 'cdxCmtsCmRegistered', which records the number of online and active modems at a given timepoint, and is used to detect a network fault. When the number of active modem users is 15% less than the previous twenty iterations of the same collection, a failure is recorded [173]. Hard thresholds of the PNM metrics have also been used in additional studies to detect network issues[94, 159], however again not CPD specifically.

As described previously, much of the relevant research available in PNM in the HFC networks cites MER as a highly significant variable. This is certainly the

case with the PNM paper by Zhu *et. al* where the authors outline a number of signal faults that can be detected by specifically using Receiver MER (RxMER - MER from the DS signal) [203]. They then use this parameter to create an automated detection platform of faults using machine learning methods. Overall, however, using PNM metrics to detect CPD in a coaxial cable network has not been extensively investigated, despite it being such a pervasive issue.

In many of the HFC networks referred to in the above work, namely [71] and [173], it specifically mentions the fact that amplifiers do not collect data, hence it is difficult to gain an accurate insight into the status of the health of the network at every location. In [71], the authors overcome this by aggregating the values of PNM parameters of all CMs that share the lowest common amplifier in the network hierarchy or *last line amplifier*. If the behaviour at multiple last line amplifiers is anomalous, then the conclusion is that the problem is located at an amplifier higher up in the hierarchy.

7.3.2 Problem Complexity

Common Path Distortion is described as the leading cause of network failure in coaxial cable networks annually by TDC NET. It is indistinguishable from other high noise impairments in the upstream signal. Consequently, analysing PNM metrics becomes necessary in order to identify any prevailing patterns in the data. Nonetheless, using PNM metrics to detect CPD is not a straightforward solution and, as stated previously, some of the issues include:

- There is no shared consensus on how best to use the PNM parameters to detect faults
- The amount of PNM data being collected is extensive and therefore difficult to monitor
- There is a lot of noise in the PNM parameters and this makes it difficult distinguishing high noise errors from point anomalies which do not generally cause network errors

Therefore, the way in which CPD presents in the data must be specifically examined and predetermined before using it as a means of detecting faults. Unfortunately, even this is not a clear-cut process. There is no existing dataset containing labelled instances of CPD, and any known methods of detecting network breakdown are considered unreliable indicators i.e. customer trouble tickets. For the reasons described in Section 3.1, it was decided to omit applying data from trouble tickets entirely from the labeling process.

7.3.3 Contributions

It is the aim of this research to:

- Establish a reliable, labelled dataset for classifying CPD faults in HFC networks. The dataset is to be made public and used in this and future work.
- Propose a precise way of identifying CPD in a HFC network, directly from PNM data, comprehensively testing this hypothesis using statistical and machine learning methods on the developed datasets.

It is the aim of this research to, in collaboration with industry domain experts, establish a precise way of identifying CPD in a HFC network, comprehensively testing this hypothesis using statistical and machine learning methods and making the methods and datasets used in this paper public for future use and analysis.

7.4 Dataset Generation

In collaboration with domain experts at TDC NET, we have generated a dataset to be used for further analysis. The dataset consists of instances in the PNM technology parameters (MER and SNR), which have been labelled positively or negatively for the presence of CPD. We name the dataset BoCPaD (Broadband Common Path Distortion dataset). In the following, we describe details of the generation of the dataset.

7.4.1 BoCPaD: Real-World Labelled Dataset

Spectrum analyzers are used by field work technicians to successfully determine anomalies at a given location. Although a spectrum analyzer, with a high degree of accuracy, can identify cases of CPD the use of the spectrum analyzer is not considered optimal for the detection of CPD cases. This is due to the fact that spectrum analyzers must be at the point of the network at which the fault is present in order to verify an instance of CPD. This results in extremely time-consuming and resource-heavy fault detection for field work technicians.

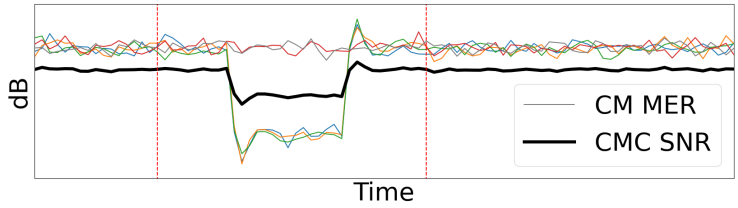


Figure 7.3: Representation of the rule of thumb for the presence of CPD in PNM data. Some CMs experience a DS MER downwards deviation correlating with the CMC SNR.

A CPD instance is associated with *a last-line amplifier on a specific day* and the corresponding data is the time series measurements from all the CMs that are children of that amplifier and the CMC-measurements of the ancestoring CMC (the root of the tree) during the given 24 hour period. The behaviour of the CMC SNR and the minimum CM MER (herafter just denoted CM MER) parameters at the time and location of the fault are examined. Based on the technicians' domain knowledge about HFC networks in general combined with their experience obtained from measurements with a spectrum analyzer, they are able to distinguish a fluctuation pattern in the parameters caused by CPD from other typical fluctuations in the MER and SNR behaviour. This allows them to manually label CPD instances by visually inspecting the time series graphs obtained on the PNM metrics.

When the technicians/domain experts are asked how they achieve the CPD labelling on the data series, a single quantitative description cannot be pinpointed. This is due to it being partially based on tacit knowledge (the aim of building our ML models is to bring this tacit knowledge into an applicable quantitative model). However, they can provide us with a rule of thumb, which can be summarized as follows: when the CM MER deviates significantly downwards between timestamps simultaneously for a group of modems under a particular amplifier, and the US CMC SNR moves downwards during the same time period, then CPD is present in the network and is affecting the given amplifier. A visual representation of this rule of thumb is illustrated in Figure 7.3.

Furthermore, we learn from the domain experts that instances with too few modems are disregarded as these are deemed too challenging to label. It is determined that in order to effectively conclude whether CPD is present for a given set of lowest line amplifier and its underlying modems, at least 40% of the modems must show signs of CPD fluctuation. Hence only these will be labelled in this study. For example, in Figure 7.4, it can be seen while the symptoms of CPD are present in Modem 3 and 4 for a given aggregation of a given day,

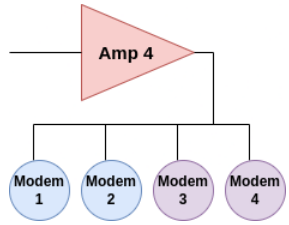


Figure 7.4: Lowest line amplifier with some modems showing signs of CPD (purple).

Modem 1 and 2 are behaving normally. If the fluctuation was only visible in Modem 3, then it could not be precisely deduced that the fluctuation present is caused by CPD. Also the labelling procedure pays attention to the CMC behaviour and therefore identifying an instance of CPD is conditional on the upstream SNR of the CMC also deviating. If, the conditions are not fulfilled in any way[3] then the suspected case is not indicative of CPD but rather another type of error. Therefore, the correlation between CMC SNR and CM MER also has to be taken into account. After manually labelling the instances using the algorithm described in section 7.A, negative samples are randomly extracted to ensure a balanced dataset. The sampling is carried out to ensure that the dataset contains an accurate representation of the distribution of cases of CPD in the actual network. It should be noted that the instances of CPD which are negative in the dataset may contain other network errors, but are acknowledged to be free from CPD.

7.4.1.1 Preprocessing and time point alignment

Both CM MER and CMC SNR are measured over six channels, recording a value every 15 minutes on each. The values of the channels are averaged for each time point. In order to make the time series in the same instance directly comparable, the time points of the polls need to be aligned and the missing values imputed. CMs are not always polled at the same time and with exactly 15 minutes between. We align the time points by assuming measurements are polled simultaneously for both CMs and the CMC every 15 minutes, which generates a sequence of time points with exactly 15 minutes between each point. The new sequence would then be shifted so that it would be polled according to the original CMC time point sequence (the CMC time series is to be compared to all of the CM time series individually). Furthermore, we use linear interpolation to impute missing values. The alignment and imputing scheme is illustrated in

[3]For example, if the modem deviates and the CMC does not

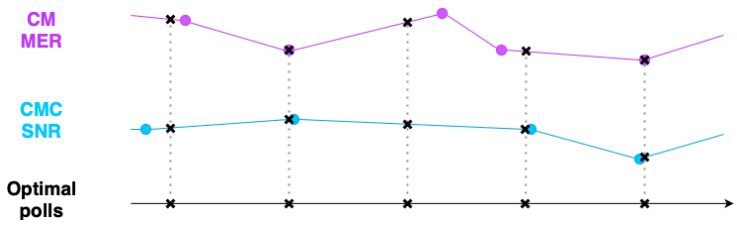


Figure 7.5: Illustration of the time point alignment and missing time point imputation scheme. The optimal placement of polls with respect to time is derived from the CMC SNR polls assuming polls every 15 minutes. These optimal polls are then used to impute missing data and align time points across multiple time series using linear interpolation.

Figure 7.5.

7.5 Proposed Method

In this section we present our proposed algorithm for detection of CPD faults in broadband networks using individual 24-hour time series. Our algorithm consists of two steps. Firstly, a feature engineering scheme is applied to the time series data. Secondly, the application a given ML method to the feature-engineered variables in order to learn the labelled classes of the data. We investigate different ML methods, including XGBoost[31], Random Forest[105], Logistic Regression[36], and a Multi-Layer Perceptron (MLP)[118]. We compare these models to an intuitive business rule based on the raw data. The business rule (explained in section 7.5.3) is rooted in both; the exhibited signal behaviour when a CPD fault is present, as believed by domain experts in the broadband network field; and a SHAP (SHapley Additive exPlanations) analysis of important variables with respect to the ML models used on the feature-engineered variables.

7.5.1 Feature Engineering

The classification of a given observation in the dataset is based on whether or not a CPD fault is present at a given last-line amplifier on a given day. Since an amplifier can be connected to an arbitrary number of modems, it means that the basis of the features of an observation consists of an arbitrary number of time series across a number of observations. In order to facilitate the learning of ML

models that require a fixed-size input, two different feature engineering schemes are proposed. These schemes have two objectives: 1) ensuring the features for a given observation have standardized size, and 2) encoding information identified to be relevant for learning whether or not a CPD fault is present. The first scheme is based on binning the relevant variables at distinct intervals whereas the second scheme makes use of distributional moments of the same variables.

7.5.1.1 Binning scheme

We consider N observations/examples, and a given observation $i \in \{1, \ldots, N\}$ has associated with it; an amplifier a_i, a specific date d_i, and a label $y_i = 1$ if a CPD fault is associated with a_i during d_i, and $y_i = 0$ otherwise. Let the matrix $\boldsymbol{M}_i$ be the time series made up of RxMER measurements during d_i from each child modem of amplifier a_i, which means that $\boldsymbol{M}_i$ will have shape $M_i \times T_i$ where M_i is the number of modems under the amplifier a_i and T_i will be the number of sampled measurements during the day[4]. Finally, let $\boldsymbol{c}_i$ be the vector of CMC SNR measurements in time during d_i from the CMC that is the ancestor of a_i. According to observation i, we calculate the following features ($|\bullet|$ denotes the cardinality):

- **CMC SNR Standard deviation**:

$$S_i = \sigma(\boldsymbol{c}_i) \tag{7.1}$$

  where $\sigma(\bullet)$ is the standard deviation.

- **Binned CMC SNR z-scores** Normalized fraction of polls in bins with breakpoints $\boldsymbol{b}_z$, resulting in a vector $\boldsymbol{z}_i$ with elements:

$$\boldsymbol{z}_i(l) = \frac{1}{T_i} \sum_k \mathbb{1}^{(l)}_{\boldsymbol{b}_z} \left(\frac{\boldsymbol{c}_i(k) - \bar{\boldsymbol{c}}_i}{S_i} \right), \quad l = 1, \ldots, |\boldsymbol{b}_z| - 1 \tag{7.2}$$

  where $\boldsymbol{c}_i(k)$ is the k-th element of $\boldsymbol{c}_i$ and:

$$\mathbb{1}^{(\alpha)}_{\boldsymbol{\beta}}(x) = \begin{cases} 1 & \text{if } \boldsymbol{\beta}(\alpha) \leq x < \boldsymbol{\beta}(\alpha + 1) \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

- **Binned CM MER standard deviations** Normalized fraction of CMs with MER standard deviation in bins with breakpoints $\boldsymbol{b}_s$ resulting in a vector $\boldsymbol{s}_i$ with elements:

$$\boldsymbol{s}_i(l) = \frac{1}{M_i} \sum_k \mathbb{1}^{(l)}_{\boldsymbol{b}_s}(\sigma(\boldsymbol{M}_i(k))), \quad l = 1, \ldots, |\boldsymbol{b}_s| - 1 \tag{7.4}$$

[4] T_i is always approximately 96, but can vary slightly due to positions of the polls.

where $\boldsymbol{M}_i(k)$ denotes the k-th row of $\boldsymbol{M}_i$ corresponding to the time series from the k-th modem and where the indicator is given in (7.3).

- **Binned correlations** Normalized fraction of CMs for which the correlation of the MER with the CMC SNR falls in bins with breakpoints $\boldsymbol{b}_r$, resulting in a vector $\boldsymbol{r}_i$ with elements:

$$\boldsymbol{r}_i(l) = \frac{1}{M_i} \sum_k \mathbb{1}^{(l)}_{\boldsymbol{b}_r} \left(\rho(\boldsymbol{M}_i(k),\ \boldsymbol{c}_i) \right), \quad l = 1, \ldots, |\boldsymbol{b}_r| - 1 \tag{7.5}$$

where $\rho(\bullet)$ is the Pearson correlation and the indicator is given in (7.3).

- **Combinations of bins** Fraction of CMs in every combination of the bins given in (7.4) and (7.5), resulting in a matrix $\boldsymbol{Q}_i$ with elements:

$$\begin{aligned} \boldsymbol{Q}_i(l,h) = \frac{1}{M_i} \sum_k \mathbb{1}^{(l)}_{\boldsymbol{b}_s}(\sigma(\boldsymbol{M}_i(k))) \cdot \mathbb{1}^{(h)}_{\boldsymbol{b}_r}(\rho(\boldsymbol{M}_i(k),\ \boldsymbol{c}_i)) \\ l = 1, \ldots, |\boldsymbol{b}_s| - 1 \qquad h = 1, \ldots, |\boldsymbol{b}_r| - 1 \end{aligned} \tag{7.6}$$

The resulting observation will be a concatenation of all these features after flattening $\boldsymbol{Q}_i$:

$$\boldsymbol{X}(i) = \begin{bmatrix} S_i & \boldsymbol{z}_i^\top & \boldsymbol{s}_i^\top & \boldsymbol{r}_i^\top & \boldsymbol{Q}_i(1) & \boldsymbol{Q}_i(2) & \cdots \end{bmatrix} \tag{7.7}$$

which means that each observation will have

$$\begin{aligned} |\boldsymbol{X}(i)| &= 1 + (|\boldsymbol{b}_z| - 1) + (|\boldsymbol{b}_s| - 1) + (|\boldsymbol{b}_r| - 1) \\ &\quad + (|\boldsymbol{b}_s| - 1) \cdot (|\boldsymbol{b}_r| - 1) \\ &= |\boldsymbol{b}_z| + |\boldsymbol{b}_s| \cdot |\boldsymbol{b}_r| - 1 \end{aligned} \tag{7.8}$$

features associated with it. The bins for the standard deviations and correlations respectively are included in the data because these are beneficial in detecting fluctuations in the parameters. For the CMC SNR, the standard deviation and binned z-scores are included to make the model more robust to short and insignificant fluctuations in the US signal.

7.5.1.2 Distributional moments scheme

A probability distribution is characterized by its moments which analyze the dispersion and location of the data. The third moment, for example, carries information about the skewness of the data. In this instance, we wish to know if by calculating the moments of the distribution of modem standard deviations of

RxMER and correlations to the CMC SNR, it will provide a more optimal feature description than that of the binning approach or whether it could enhance the binning approach in another way. As before, an observation $i \in \{1, \ldots, N\}$ consists of an amplifier a_i, a specific date d_i, and a label $y_i = 1$ if a CPD fault is associated with a_i during d_i, and $y = 0$ otherwise. The same notation exists for the matrix of modem RxMER features and CMC SNR features. This means that for each observation i we have three distributions that we want to characterize:

- **Standard deviation of modem RxMER**

  Let $\boldsymbol{sm}_i$ be the vector of standard deviations of the RxMER measurements for the different modems in the i-th observation, i.e. with elements:

  $$\boldsymbol{sm}_i(k) = \sigma(\boldsymbol{M}_i(k)) \tag{7.9}$$

  Where $\sigma(\bullet)$ is the standard deviation.

- **Correlations between modem RxMER and CMC SNR**

  Let $\boldsymbol{sc}_i$ be the vector of correlations between modem RxMER measurements and CMC SNR measurements for the ith observation. Elements are given by:

  $$\boldsymbol{sc}_i(k) = \rho(\boldsymbol{M}_i(k), \boldsymbol{c}_i) \tag{7.10}$$

  Where ρ is the Pearson correlation.

- **CMC SNR time series measurements**

  Time series SNR measurements from the CMC of the i observation, denoted by $\boldsymbol{c}_i$.

For each of these distributions we calculate characteristics according to the first four distributional moments [204]. That is we calculate the **mean**, the **variance**, the **skewness** given by:

$$\mu_3(\boldsymbol{x}) = \frac{\sum_k (\boldsymbol{x}(k) - \overline{\boldsymbol{x}})^3}{(|\boldsymbol{x}| - 1) \cdot \sigma(\boldsymbol{x})^3} \tag{7.11}$$

and the **kurtosis** given by:

$$\mu_4(\boldsymbol{x}) = \frac{\sum_k (\boldsymbol{x}(k) - \overline{\boldsymbol{x}})^4}{(|\boldsymbol{x}| - 1) \cdot \sigma(\boldsymbol{x})^4} \tag{7.12}$$

Therefore, a single observation will comprise of a concatenation of all of these features.:

$$\begin{aligned} \boldsymbol{X}(i) = \big[\; & \overline{\boldsymbol{sm}_i} \quad \sigma(\boldsymbol{sm}_i)^2 \quad \mu_3(\boldsymbol{sm}_i) \quad \mu_4(\boldsymbol{sm}_i) \quad \ldots \\ & \ldots \quad \overline{\boldsymbol{sc}_i} \quad \sigma(\boldsymbol{sc}_i)^2 \quad \mu_3(\boldsymbol{sc}_i) \quad \mu_4(\boldsymbol{sc}_i) \quad \ldots \\ & \ldots \quad \overline{\boldsymbol{c}_i} \quad \sigma(\boldsymbol{c}_i) \quad \mu_3(\boldsymbol{c}_i) \quad \mu_4(\boldsymbol{c}_i) \; \big] \end{aligned} \tag{7.13}$$

We calculate the standard deviation instead of the variance for the CMC SNR measurements due to comparability to the other methods.

7.5.2 Modelling

We have examined and compared the classification performances of four standard machine learning models on our data set. The models are:

- **Random Forest** - which is an ensemble method combining a set of different classification binary decision trees to make a final prediction[105].
- **Logistic Regression** - which is a linear model of the log-odds of the outcome[36].
- **XGBoost** - which is a boosting algorithm based on classification trees[31].
- **Multi-Layer Perceptron** - which is neural network consisting of at least two fully-connected, linear layers[118].

We evaluate the models by quantifying each classifier on a number of performance metrics including accuracy, sensitivity, specificity and precision. We use five-fold cross-validation to assess the performance on a test set[152].

For models trained on the distributional moments features we perform a SHAP analysis [108] to identify important features with respect to the classification problem. We use these features to propose an intuitive business rule based on two-step thresholding and compare it to the models trained on the engineered features.

7.5.2.1 Model selection

In the case of using the binning scheme described above for feature-engineering, we search for the optimal number of histogram bins for the feature-engineered parameters and for the optimal hyper-parameters for the different ML models to be investigated. With respect to the size of the bins or number of breakpoints potentially impacting the performance of the model, a number of trial and error experiments were carried out to assess whether an optimal number of bins and breakpoints could be found. Initially, the bins were generated using 8 equal-sized quantiles. This means that an observation (given a set of time series during a day) is comprised of:

- Standard deviation of the CMC SNR
- Fraction of CMs with a MER standard deviation within each corresponding bin
- Fraction of Z-scores of the CMC SNR (statistical measurement of a value's normalised distance to the mean) within each corresponding bin
- Fraction of CMs for which the correlation between the MER and the CMC SNR is within each corresponding bin.

It was eventually found that whilst bin sizes did not have a significant impact, the placement of the breakpoints had some dependency with the performance of the model. Therefore, the data was eventually processed so that it was divided into just two bins and with the position of the bin breakpoints varying with each iteration. In training each of the ML models (using feature-engineered variables) and in the case of binary bins, we perform a grid search of optimal breakpoints for the CM MER standard deviations and the correlations to the CMC SNR values. For each pair of breakpoints, we train and validate each of the models using five-fold cross validation and report the mean AUC over the five folds.

The machine learning models were selected due to their suitability in performing binary classification. Logistic regression was used as it predicts probabilities of an observation belonging to a particular class. It was also important to assess the performance of ensemble learning methods, therefore Random Forest was selected. On the basis of their dominance in machine learning in terms of speed and performance XGBoost and a Multi-Layer Perceptron (Neural Network) was also selected. Tuning the hyperparameters of these models was not a priority of this research as the focus was placed both on the feature engineering scheme and the explainability of the features.

The `scikit-learn (1.0.2)` library [136] in `python (3.9.1)` was used for implementations of all the different models.

7.5.3 Explainability and Important Variables

In order to avoid the ML models being complete black boxes with lack of transparency several methods exist to give some degrees of explainability. Where the coefficients obtained for a linear model like logistic regression can give some immediate insight, more sophisticated methods are needed for non-linear models. Shapley values, or SHAP values, are a commonly used approach initially

developed in cooperative game theory which can help define which parameters of a machine learning model are the most useful [108] or have the biggest impact. It is a model agnostic framework and thus can be used to interpret the output of a variety of ML models. In essence, Shapley values describe the mean marginal contribution of each feature value across all values in a feature space. The formula is described in equation (7.14) ($|\bullet|$ again denotes the cardinality):

$$\varphi_i(v) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(F - |S| - 1)!}{|F|!}(v(S \cup \{i\}) - v(S)) \tag{7.14}$$

where F is the set of features, S is the coalition subset of players and $v(S)$ is the total value of S players.

Shapley values can provide an insight into both the magnitude and direction of each feature's effect on an observation. They can describe both contributions of individual observations, or local behaviour, and the conduct of the data collectively, or global behaviour [6]. While Shapley values can be demanding to calculate for general ML models the scheme is especially suited for ensembles of decision trees.

7.5.3.1 Intuitive business rule

For comparison to the four ML models described in section 5.2.1, we develop an intuitive business rule[5] based on the outcome of an analysis of the most important features for modelling CPD as described above and according to the proposed signal footprint during the presence of a CPD fault. Domain experts believe that simultaneous drops in CM MER exhibited by modems that share the same last-line amplifier and a concurrent drop in CMC SNR will be evidence of CPD. An analysis of the SHAP-values of various ML methods further supports this idea (see section 7.5.3 for the analysis). With this in mind we develop a business rule based on two-step thresholding for the CM MER standard deviations, and the correlation between the CM MER and the CMC SNR. It should be noted that this business rule does not require fixed-size inputs, hence will be evaluated on the raw time series from an observation and not on the feature-engineered parameters.

Given an observation, i, we define two parameters α_i and β_i whereof the first is the fraction of modems exhibiting a CM MER standard deviation above a predetermined threshold, ω_σ:

$$\alpha_i = \frac{\sum_{k=1}^{M_i} \mathbb{1}_{\omega_\sigma}(\sigma(\boldsymbol{M}_i(k)))}{M_i} \tag{7.15}$$

[5] A model based on (multiple levels of) thresholds of specific parameters.

where σ is the standard deviation and where:

$$\mathbb{1}_a(x) = \begin{cases} 1 & \text{if } x > a \\ 0 & \text{otherwise} \end{cases} \tag{7.16}$$

The other is based on the fraction of modems for which the correlation between the CM MER and CMC SNR is above a predetermined threshold, ω_ρ:

$$\beta_i = \frac{\sum_{k=1}^{M_i} \mathbb{1}_{\omega_\rho}(\rho(\boldsymbol{M}_i(k), \boldsymbol{c}_i))}{M_i} \tag{7.17}$$

where ρ is the Pearson correlation and where the indicator is given in (7.16). We then model the outcome of the ith observation using two additional thresholds for each of the two parameters respectively, namely ϕ_σ and ϕ_ρ:

$$\hat{y_i} = \begin{cases} 1 & \text{if } \alpha_i > \phi_\sigma \quad \wedge \quad \beta_i > \phi_\rho \\ 0 & \text{otherwise} \end{cases} \tag{7.18}$$

During the search for the optimal thresholds, we treat β_i as the prediction probability and carry out a grid search for optimal values of ω_σ, ω_ρ, and ϕ_σ by maximizing the ROC AUC (an aggregated measure of performance) over all possible classification thresholds. For comparability, we train using five-fold cross-validation and report the optimal parameters and AUCs averaging over the five folds. For each proposed value of ϕ_σ we set $\beta_i = 0$ if $\alpha_i \leq \phi_\sigma$ and otherwise we keep its value.

7.6 Evaluation

In this section we first evaluate the characteristics of our obtained dataset before we present the results of applying the proposed models to it. For all modelling results, we choose the optimal threshold to be the one maximizing the $F_{1/2}$-score, which means that we set the precision to have twice the importance of recall. This choice was made due to the fact that sending a technician to a false positive would be costly and undesirable, hence we would prefer less severe cases of CPD remain undetected rather than the opposite[6].

7.6.1 Resulting Dataset

Table 7.1 shows the characteristics of the dataset resulting after manually labelling relevant observations. The dataset observations consists of; an array of

[6]We make the assumption that less severe cases of CPD will lead to lower prediction score.

time series measurements of the CM MER from a set of CMs, a vector of time series measurements of the CMC SNR, a label, and timestamp vectors that allow for correct interpolation of missing values or proper alignment of time series polls.

Table 7.1: The metadata of the acquired dataset that consists of observations manually labelled by domain experts at TDC NET.

	BoCPaD
Observations	655
CPD cases	212 (32.4%)

7.6.2 Modelling Results

Results for the individual models using the feature-engineering scheme based on binning are shown in Table 7.2. To be able to compare the results of the intuitive business rule to the rest of the models, results of these are reported based on (individual) optimal break-points in the binary bin case of feature engineered variables. Table 7.3 shows the results of the individual models using the moments feature-engineering scheme and does not include the intuitive business rule because it is not dependent on feature-engineering. Results for both tables include AUCs (Area Under the Curve) of both the ROC (Receiver-Operator Characteristic) and the PR (Precision-Recall) curve, and the optimal $F_{1/2}$ score along with corresponding precision and recall values. From both tables it is clear that the XGBoost model is the superior model achieving a precision of 92.0% and 93.7% respectively, while detecting respectively 77.8% and 68.7 % of the cases. It seems that the intuitive business rule achieves comparable results though estimating less parameters (especially compared to the ensemble method). For the binning-scheme, it should be mentioned that all models, including the intuitive business rule, have been allowed to search the continuous space for optimal break-point values.

7.6.2.1 Optimal thresholds and breakpoints

For both the intuitive business rule and the ML models that are run on the binned features, a search for optimal thresholds and breakpoints needs to be carried out. In training the intuitive business rule, we searched the space of threshold pairs, ω_σ and ω_ρ. For every pair we we calculated the AUC for every

Table 7.2: The cross-validated results of applying different models to the BoCPaD dataset using the binning feature-engineering sceme. Binary bins have been used with individually optimized breakpoints. The results for the recall and precision are chosen at the threshold maximizing the $F_{1/2}$ score.

	ROC AUC	PR AUC	Recall	Prec.	$F_{1/2}$
Business rule	.916	.831	**.848**	.797	.807
Logistic reg.	.930	.827	.737	.862	.833
Random forest	.956	.911	.808	.883	.867
XGBoost	**.960**	**.927**	.778	**.920**	**.888**
MLP	.942	.854	.758	.868	.844

Table 7.3: The results of applying different models to the BoCPaD dataset using the distributional moments feature-engineering scheme. The results for the recall and precision are chosen at the threshold maximizing the $F_{1/2}$ score.

	ROC AUC	PR AUC	Recall	Prec.	$F_{1/2}$
Logistic reg.	.918	.838	**.737**	.826	.806
Random forest	.954	.910	**.737**	.889	.854
XGBoost	**.959**	**.928**	.687	**.937**	**.873**
MLP	.905	.822	**.737**	.796	.783

Table 7.4: Optimal threshold values resulting after cross-validating our intuitive business rule approach. Notice that the optimal value for ϕ_ρ is the only one chosen post-hoc (maximizing the $F_{1/2}$ score), as the fraction that governs it has been treated as the prediction probabilities during learning.

Parameter	ω_σ	ω_ρ	ϕ_σ	ϕ_ρ
Optimal value	0.3701	0.5048	0.2778	0.2953*

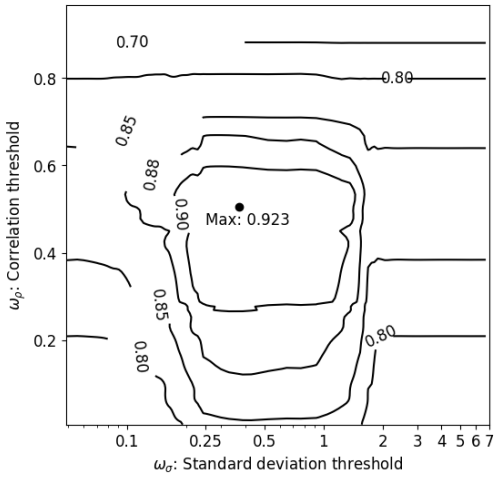


Figure 7.6: Contour plot of the cross-validated maximum ROC AUCs given different combinations of CM standard deviation (ω_σ) and correlation thresholds (ω_ρ) for the business rule. The x-axis is plotted on a logarithmic scale and the black point is the combination for which the model performs the best. Optimal values can be seen in Table 7.2.

value of ϕ_σ and report the maximum AUC and corresponding value of ϕ_σ. A contour plot of the cross-validated maximum AUC given thresholds for the standard deviation (ω_σ) and correlation (ω_ρ) respectively is shown in Figure 7.6. From the plot it seems that the maximum AUC is a rather smooth surface on which optimal values of ω_σ and ω_ρ can be found as the global maximum. The optimal value for ϕ_ρ is chosen as the threshold maximizing the $F_{1/2}$ score, i.e. choosing precision to have double the importance of recall. The performance of the intuitive business rule using these values are given in Table 7.2 along with results from the other models. Optimal threshold parameters are reported in Table 7.4. As a results of our search for optimal break-points, a contour plot of the cross-validated maximum AUC for the XGBoost model using different choices of breakpoints is shown in Figure 7.7. The AUC landscape seems smooth,

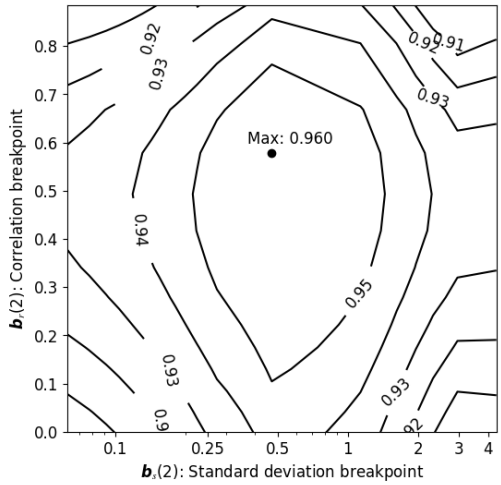


Figure 7.7: Contour plot of the cross-validated maximum ROC AUCs for the XGBoost model given different combinations of breakpoints for the CM MER standard deviation ($\boldsymbol{b}_s(2)$) and correlation to the CMC SNR ($\boldsymbol{b}_r(2)$) bins respectively in the case of having only two bins for either of the two parameters. The standard deviation axis is plotted using a logarithmic scale, and the black point is the combination that lead to the model performing the best.

meaning that the XGBoost model is robust to choices of breakpoints. Results for all models are given in Table 7.2 for the case of binary bins and optimal breakpoints have been found for each model individually using five-fold cross validation. The corresponding optimal breakpoints are reported in section 7.B.

7.6.2.2 Number of bins for feature-engineered variables

As a results of our grid search for the optimal number of bins for either parameter, a contour plot of the maximum AUC of the XGBoost model given different combinations of number of bins for each variable is shown in Figure 7.8. From the figure is seems that the relation is somewhat random with many local maxima and minima. Combined with Figure 7.7 it suggests that it is not the number of bins, but rather the position of break-points that is important.

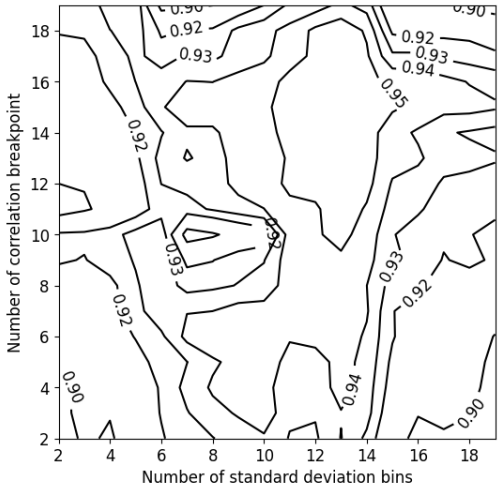


Figure 7.8: Cross-validated AUCs for the XGBoost model given different combinations of number of bins for respectively SDs and correlations.

7.6.2.3 Shapley values for moments features

We calculated the SHAP values for the best performing ML model, XGBoost, on the moments feature-engineered dataset. The features that had the highest impact are plotted in a summary plot to interpret the model's global behaviour. It can be seen in Figure 7.9.

The figure shows that the Mean Correlation ($\overline{\boldsymbol{sc}}$) and Modem Standard Deviations ($\overline{\boldsymbol{sm}}$) are the most important features for detection of instances of CPD, whilst the Skewness of the CMC SNR ($\mu_3(\boldsymbol{c})$) is the least important observed. When Mean Correlation has high values it has positive SHAP values, suggesting it drives the models decision towards detecting CPD. Similar behaviour can be seen for Mean Standard Deviations of modem MER. However, the converse is true for the Kurtosis of Correlation ($\mu_4(\boldsymbol{sc})$), for example, where high coloured values result in negative SHAP values, suggesting that it drives the decision of the model towards a case of CPD.

The SHAP framework also allows for local explanations of the outcome of single observations. An example of this is provided in section 7.C.

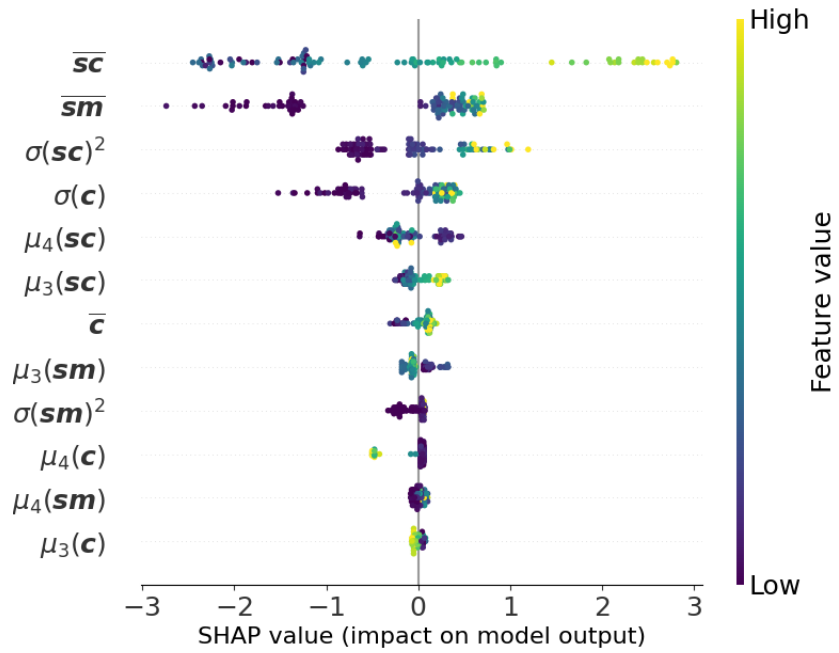


Figure 7.9: Summary plot of XGBoost model to detect instances of CPD

7.7 Discussion

In tables 7.2 and 7.3 it is clear that from the models proposed, the XGBoost model achieves the best accuracy in distinguishing between CPD and non-CPD cases. We believe that the precision is sufficiently high to be effective for proactive maintenance of the network. In the feature-engineering scheme based on binning, two bins seem to be sufficient for modelling the problem, if care is taken to search for optimal break-points. This is shown in Figure 7.8 where modelling performance does not seem to depend on the number of bins, but rather on the location of the break-points. This is supported by the fact that the optimal break-points in the case of binary bins leads to an AUC of 0.960 for the XGBoost model while the case of numerous bins hardly ever reaches that performance using the same model. Comparing tables 7.2 and 7.3 we see that performances for the two feature-engineering schemes are very similar, suggesting that a feature-engineering scheme that properly encodes the distributional properties of respectively the CM MER standard variation and the correlation to the CMC SNR is appropriate for modelling CPD.

Looking at the intuitive business rule, we show that the CM MER SDs and correlations to the CMC SNR have enough information to allow for rather simple thresholds to obtain comparable results using far less estimated parameters. ISPs can choose to use the intuitive business rule, as opposed to the other models, if interpretability, ease of implementation, and ease of training rank higher than precision of predictions.

In general for all the models, it seems that the recall decreases and the precision increases as the model complexity increases. This suggests that the challenge for the models lies in identifying non-CPD observations among instances that exhibit behavior typically associated with CPD, (i.e. high standard deviations for CM MER and correlations with CMC SNR) rather than detecting CPD cases in general.

The distributional moments feature-engineering scheme allowed us to perform an analysis of which features were most important in detecting CPD cases. Just as expected by domain experts, the features that contribute the most are the mean correlation between the CM MER and the CMC SNR, and the mean standard deviation of CM MER. The analysis also showed that it is not only the mean correlation, but also the variance of correlations that contributes to the detection. This suggests that the impact on modems varies further suggesting that CPD has many faces and is not a straightforward problem.

7.7.1 Assumptions

The dataset used to train and test the methods described here is assumed to be representative of real network behaviour and has been balanced in order to allow for accurate modelling of the problem. This is important as the proportion of anomalous data to normal data can effect the behaviour of the model.

We make the assumption that if CMs under the same amplifier show anomalous behaviour at the same time, this is most likely due to a collective fault. This hypothesis is supported through experimentation and through discussions with domain experts.

It is assumed that 24 hours of real time measurements is sufficient when detecting instances of CPD. It was important to find a period that is sufficiently long for the CPD footprint to be measurable, but sufficiently short so that the probability of the error appearing or disappearing within the period is low. The choice of 24 hours was informed by domain experts and by examination of known instances of CPD that have been identified. This means that the timepoints of CM and CMC values are aggregated to a single observation for 24 hours. Windows never overlap, but due to the fact that the window size is a 24-hour period, two observations could stem from the same amplifier and modems on consecutive days when the fault is unresolved on the first day. This introduces unwanted correlation. However, due to the length of the sampling period and the size of the whole network, this is unlikely. Although the fact that the parameters are aggregated by 24 hours could be considered a limitation, it can be considered a significant improvement over the work in [71], which required 72 hours of data to detect a case of CPD. We also note that that the algorithms / models presented here do not depend on information from customer trouble tickets and are therefore less prone to human error.

7.7.2 Limitations

The methods described here depend on the entirety of the HFC topology being known i.e. the path from the CMC to each CM must be detectable in order for the CMs to be aggregated under the last line amplifier. However, for many service providers, including TDC NET it is not always feasible to know how each amplifier in the topology is connected. This is due to a variety of reasons, for example customers changing address and taking their CM to a new address, the service operator obtaining new networks from other providers, or a part of the network being operated by private unions. Therefore, in cases where the network topology is not totally known, the methods are not feasible.

The dataset was randomly sampled with negative instances of CPD to represent a realistic balance between the classes. However, samples taken on subsequent days may not necessarily be independent. As it is unknown when the error was repaired, it could be possible that an observation that has been labelled negative is in fact positive.

The data is aggregated to a single observation for every individual date. This means that if the CPD occurs over the course of two days, but the standard deviation does not significantly differ on each individual day, the case of CPD is not detected. This could potentially be fixed by having overlapping instances, but that is not how the data was acquired for this paper.

The data was labelled after extensive discussion and analysis with domain experts in the field of HFC networks and verified using specialized equipment in the field. However, here is a risk that not all cases of CPD in the network were detected, i.e. false negatives. CPD is not a straightforward problem and there may be variability in how it manifests itself in the PNM data. As only two PNM parameters (SNR and RxMER) were utilized in this research, there is a possibility that symptoms of CPD is present in other PNM parameters that have not been tested here. Nonetheless, although there may be cases of CPD that do not present themselves in these parameters, the methods here have successfully detected a verified number of CPD cases with a low error rate. Failing to detect less severe cases of CPD might not be problematic, since less severe cases would also be less of a priority for the customer experience.

7.7.3 Future Work

There are recommendations that could be made to improve the quality of CPD detection however, including a labelling process where all known cases of CPD which are detected in historical data are labelled. This would enable a more thorough examination into the variability of how CPD manifests itself.

Because the detection of CPD is a complex issue, a simplification of the setup would be valuable. This could be achieved e.g. by simulating the same or varying degrees of CPD faults in a laboratory or testbed setup with full control of the network architecture and the network traffic. In this way, a controlled experiment could be carried out, ensuring the exact location of the fault. This would result in a high-quality dataset with a known ground-truth that could be used to investigate the effect of varying degrees of CPD on the PNM metrics.

7.8 Conclusion

We have created BoCPaD; a novel dataset consisting of 655 manually labelled observations from the HFC network of TDC NET spanning most of Denmark. Observations include time series measurements from the CMs and CMCs in the network and are labelled either prone to CPD (32.4 % of the observations) or not.

To create a model that can detect CPD from the dataset we propose two different approaches: (1) a feature-engineering approach based on either binning or calculation of distributional moments of modems with specific behaviour in terms of relevant variables and subsequent classification using state-of-the-art ML models, and (2) an intuitive business rule based on thresholds of fractions of modems with a specific behaviour in terms of relevant variables that we verify using Shapley-values on the ML models.

We conclude that both approaches accurately model the problem, but that the feature-engineering approach based on binning achieves higher precision detecting 77.8 % of the cases with a precision of 92.0 % using an XGBoost model. We also observe that the accuracy is not very sensitive to the amount of bins, but rather the location of a specific break-point justifying the sufficiency of binary bins. We find that both proposed feature-engineering approaches achieve similar performance.

Our intuitive business rule approach detects 84.8 % of the cases with a precision of 79.7 % using far less estimated parameters. These results allow for ISPs to choose a model that is easier to interpret and implement while still achieving comparable performance.

Acknowledgment

The authors would like to thank Senior Technical Architect Claus Dreisig from TDC NET for valuable discussions and feedback. This work is supported by the Innovation Fund Denmark, project number 0224-00055B, Greenforce.

Availability

The BoCPaD (Broadband Common Path Distortion) dataset will be made publicly available online.

Compliance with Ethical Standards

This study does not contain experiments with human subjects.

Appendices

7.A Instance Labelling Algorithm

Pseudocode for the instance labelling algorithm is given in Algorithm 7.1.

7.B Optimal Break-Point for Binary Bins

The optimal break-points for the ML models using binary bins in the binned feature-engineering scheme is provided in Table 7.B.1.

Table 7.B.1: Optimal break-points of the CM MER SD and the CM MER and CMC SNR correlation respectively for each of the models. Optimal break-points are found maximizing the cross-validated ROC AUC. Notice that some values are repeated across models. This is due to the models searching the same finite grid.

	CM MER SD	CM CMC Corr.
Logistic reg.	.274	.493
Random forrest	.471	.493
XGBoost	.471	.578
MLP	.471	.578

Algorithm 7.1 Identifying instances for manual inspection / classification

$\boldsymbol{X} \leftarrow$ Extract minimum downstream SNR for each modem in time from dataset and arrange as a $modems \times time$ array
for `day` in dataset **do**
  for `amp` in last-line amplifiers under `CMC` **do**
    $Y \leftarrow 0$
    $N \leftarrow 0$
    for `modem` in modems under `amp` **do**
      if $sd(\boldsymbol{X}[\texttt{modem}, (\texttt{day} - 24H) : \texttt{day}]) > 1.5$ **then**
        $Y \leftarrow Y + 1$ ▷ Detecting fluctuations in SNR
      else
        $N \leftarrow N + 1$
      end if
      $p \leftarrow \frac{Y}{Y+N}$
      if $p > 40\%$ **then** ▷ Enough evidence is needed
        Manually inspect and label instance
      else
        Continue
      end if
    end for
  end for
end for

7.C Example of Local Explainability of an Observation

In Figures 7.C.1 and 7.C.2, examples of the impact of the features on a single observation in the data can be seen. The y-axis details the features in decreasing order of significance. If a point has a positive value that means they it has a positive impact on the prediction of the class being explained. If it has a negative value such as the Mean Correlation and the Variance of Correlation then it has a negative impact on the prediction of class. It can be seen that, in both cases, three features of the model have a negligible impact.

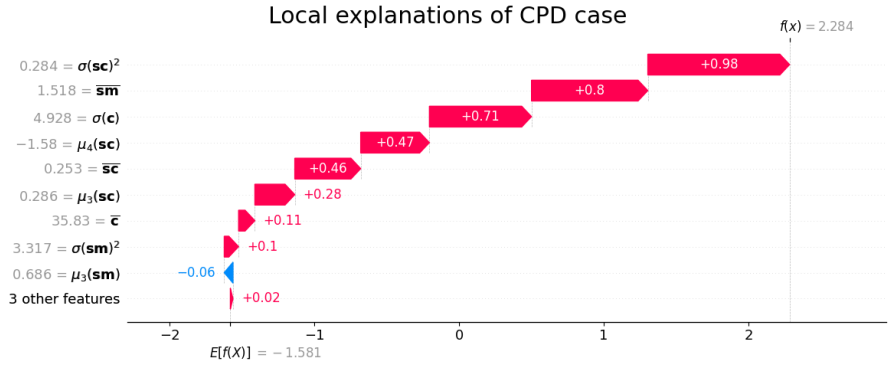


Figure 7.C.1: Waterfall plot of XGBoost model on how the features impact a single CPD-positive observation

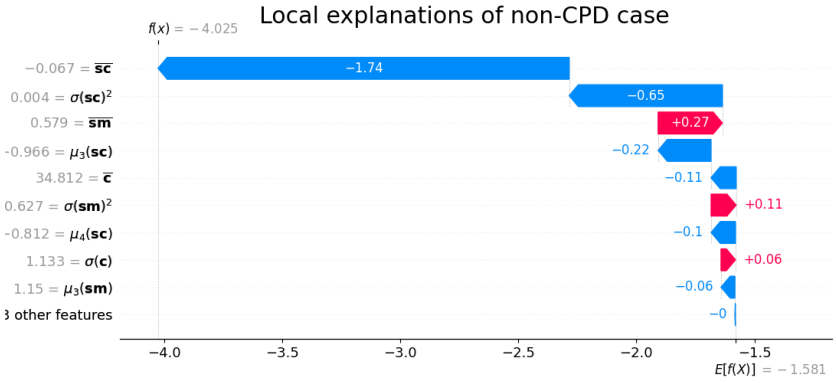


Figure 7.C.2: Waterfall plot of XGBoost model on how the features impact a single CPD-negative observation

Chapter 8

Paper B
Anomaly Detection in Broadband Networks: Using Normalizing Flows for Multivariate Time Series

Tobias Engelhardt Rasmussen[a] · Facundo Esteban Castellá Algán[a]
Andreas Baum[a]

[a] Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark

Publication Status: Paper is submitted for publication in the *Signal Processing*.

Abstract

Hybrid Fiber-Coaxial (HFC) networks are a popular infrastructure for delivering internet to consumers, however, they are complex and susceptible to various errors. Internet service providers currently rely on manual operations for network monitoring, underscoring the need for automated fault detection.

We propose a novel framework for estimating the density of multivariate time series, tailored for anomaly detection in broadband networks. Our framework comprises two phases. In the first phase, we employ an autoencoder based on one-dimensional convolutions to learn a latent representation of time series windows, thereby preserving context. In the second phase, we utilize a Normalizing Flow (NF) to model the distribution within this latent space, enabling subsequent anomaly detection. For efficient separation, we propose an iterative weighing algorithm allowing the NF to model only the systematic behavior, thereby separating outlying behavior.

We validated our methodology using a publically available synthetic dataset and real-world data from TDC NET, Denmark's leading provider of digital infrastructure. Initial experiments with the synthetic dataset demonstrated that our density-based estimator effectively distinguishes anomalies from normal behavior. When applied to the unlabeled TDC NET dataset, our framework exhibits promising performance, identifying outliers clustering themselves away from the high-density region, thus enabling subsequent root cause analysis.

8.1 Introduction

Hybrid Fiber-Coaxial (HFC) networks are one of the most popular technologies used to provide a cabled internet connection to customer premises [18]. Though the technology has been around for many years, continuing development and the relatively cheap cost of deployment suggest that it will still be an important part of digital infrastructure for many years to come. Each customer is connected to a terminal in its local area through a sequence of coaxial cables that have connections in street cabinets where the signal can potentially also be split, branching out to additional customers, or amplified to adjust for the degradation of signal power caused by resistance in the cables. The so-called Coaxial Media Converter, hereinafter CMC, connects all the customers in the

local area to the backbone internet structure using fiber technology and also acts as a converter between the fiber-optic signals and the coaxial signal used to transmit data to (Downstream - DS) and from (Upstream - US) the customers. The coaxial signal is based on an electric Radio Frequency (RF) signal that is sent through insulated copper wires designed to shield the signal from inevitable outside RF interference that would otherwise impair the signal. Due to their high complexity, HFC networks are generally vulnerable and prone to many different errors constituting e.g. degradation due to stress or corrosion [164], weather factors such as temperature and humidity [195, 129], improper customer equipment, outside electromagnetic inference [139, 140] or more abrupt errors such as when the cables in the ground are physically harmed during excavation projects [199, 173]. Reliability in broadband networks is described as being of high importance and many works have investigated how to assess and improve it [15, 67, 11, 99].

TDC NET which is the biggest provider of digital infrastructure in Denmark has a fleet of around 800 technicians, as of 2022, driving upwards of 80,000 kilometers every single day to service the network [17]. Presently both remote and on-site network monitoring are heavily based on manual operations, highlighting the need for an automatic fault detection algorithm [97]. A good understanding of the different faults that can occur in the network along with a good anomaly detection algorithm could potentially improve the daily maintenance in various ways. For instance, smart information like identification of potentially troublesome modems could be used to enhance the technician routing algorithm and thereby include the condition of the network during planning. This would allow for technicians to be dispatched to perform maintenance on nearby areas while they are already out on a task, reducing unnecessary driving in the future. Knowing the location of the error helps in planning, as it allows for the determination of whether or not a technician would need access to the customer premises. Additionally, understanding the type of error from a network monitoring perspective ensures that the technician with the appropriate skills will be sent to address the issue.

Every single modem connected to the HFC network gathers different metrics and technicians can use the so-called spectrum analyzer to obtain the real-time Full-Band Capture (FBC) of the modem, i.e. the power of the signal for every single frequency used to transmit data. This continuous real-time monitoring is, however, not practical as it would generate a significant volume of data making it harder to store and analyze. For this reason, the time series data gathered from a remote monitoring perspective is often aggregated by performing statistics or summation over some polling period. Performance metrics in the data include Modulation Error Ratio (MER), power levels of the signal, and the total number of bits sent including how many of these had to be corrected and how many could not.

One of the main challenges in detecting faults in the HFC networks is the missing ground truth, i.e. no unambiguous labels exist for when an error is present. The best indication that Internet Service Providers (ISP) have for the presence of a fault is customer tickets that are created whenever a customer calls to report an impairment in their connectivity. This is, however, not believed to be an accurate measure because whether or not a customer reports an error and the time it takes for them to make the report is believed to be more a matter of personality than of the quality of the signal [33, 77, 130]. This means that an unsupervised approach to anomaly detection in HFC networks is highly relevant.

Normalizing flows (NFs) are a type of generative model that recently has received considerable attention for density estimation/outlier detection building on a sequence of bijective functions. The use of non-linear bijective functions for density estimation was first introduced by Dinh et al. in 2014 [47] but has since been formulated as a general framework for generative modeling first mentioned by Rezende & Mohamed in 2015 [154] and popularized by Dinh et al. in 2017 [48].

In numerous cases, NFs have been used for anomaly detection in unsupervised cases - mainly due to their ability to calculate tractable probabilities and the assumption that outliers will fall into low-density regions of the input space. Research since then, however, has mostly revolved around images. Many authors have, however, concluded that out-of-distribution (OOD) anomalies are generally not well learned by deep generative models because the models are trying to force the training data into some complex distribution by performing pixel-wise transformations, which means that they often fail to capture the semantics of the data and that for instance images of something completely random can be given a relatively high probability even though the model has not yet seen such an image [90, 126, 125, 166, 34, 202, 165].

In this work, we propose a framework for unsupervised time series analysis with an emphasis on outlier detection. Our proposed method is able to detect specific anomalies in large amounts of telecommunications data and can therefore provide valuable insights/input to subject matter experts who may subsequently perform root cause analyses. Inspired by Rezende & Mohamed in [154], we apply an AutoEncoder (AE) on a time series window of a pre-determined size to obtain a descriptive embedding while denoising the input signal. We use a 1D Convolutional Neural Network (CNN) in the encoder model and deconvolutions in the decoder to preserve contextual information, hence information that can be deemed critical by a subject matter expert. Subsequently, we apply an NF to obtain the density of the embedding space. This is unlike the work of Rezende & Mohamed who derived a common loss for training both simultaneously. In our

work, however, we will train the two separately due to the preferred simplicity and the ability to train the NF on its own. We propose a novel algorithm for iteratively downweighing observations when learning the NF in order to accurately capture only the systematic behavior in the time series, by allowing the NF to update its weights solely influenced by normal (as expected) behavior while gradually disregarding anomalous behavior during the training phase. We perform an explorative analysis of the embedding space of the AE using the learned densities with the goal of exposing archetypical outlying behavior. We validate our method using the GHL dataset of multivariate time series and perform a purely explorative analysis on the data provided by TDC NET due to missing ground truth.

The main **contributions** of this work are:

- A novel framework for modeling the density of time series behavior preserving contextual information.
- A novel algorithm for learning the density of the systematic variation in a dataset that also includes observations that may contain several types of outlying behavior.
- A novel algorithm for estimating the cumulative density function (CDF) of the one-dimensional density of a multivariate distribution using bootstrapping to be used for visualization purposes.
- An explorative analysis of the outlying behavior of the time series stemming from modems in the broadband network of TDC NET.

8.2 Background and Data

In an HFC network, each customer is connected through a sequence of coaxial cables interspersed with amplifiers and splitters, usually located in street cabinets arranged in a tree-like structure rooted in the CMC. This means that the RF signal sent to a customer is a result of the path it traverses from CMC to the modem or vice versa. The RF signal is sent using a wide band of frequencies divided into different blocks or channels used for either US or DS. Each modem in the network gathers performance metrics over time, including the signal power, the Modulation Error Ratio (MER), which indicates the modem's ability to interpret the signal, the number of bits sent, the number of bits that had to be corrected, and the number of bits that could not be corrected. Due to the size of the network, these parameters are often reported as some statistic or summation during a poll period.

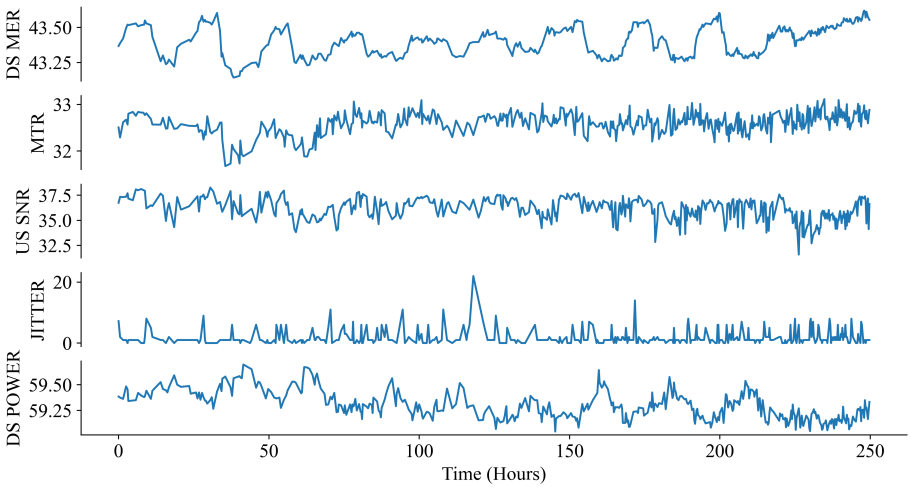


Figure 8.2.1: An example multivariate time series from a single modem from TDC NET. Power, SNR (Signal-to-Noise-Ratio), MTR (Main Tap Ratio), and MER (Modulation Error Ratio) are all measured in decibels while the jitter is measured in milliseconds. All the parameters are different metrics of the quality of the signal.

8.2.1 The TDC NET dataset

The data used in this work is provided by TDC NET which has a broadband network spanning hundreds of thousands of customer modems. Individual time series from a total of 785 modems were gathered during the months of June-September 2022 using a poll period of 15 minutes. No ground truth labeling of which time series include abnormal behavior has been provided. An example time series of a subset of the parameters based on a single modem from TDC NET is shown in Figure 8.2.1. An overview of all the parameters used in this project is provided in the appendix.

8.2.2 The GHL dataset

To validate our proposed approach, we apply it to the GHL dataset [59]. The GHL dataset is a simulated dataset based on a Modelica model of a gasoil heating loop. The authors created a setup based on a set of tanks designed to slowly heat gasoil and include different sensor measurements gathered in time. Anomalies consist of hacking attacks where parameters such as the max level

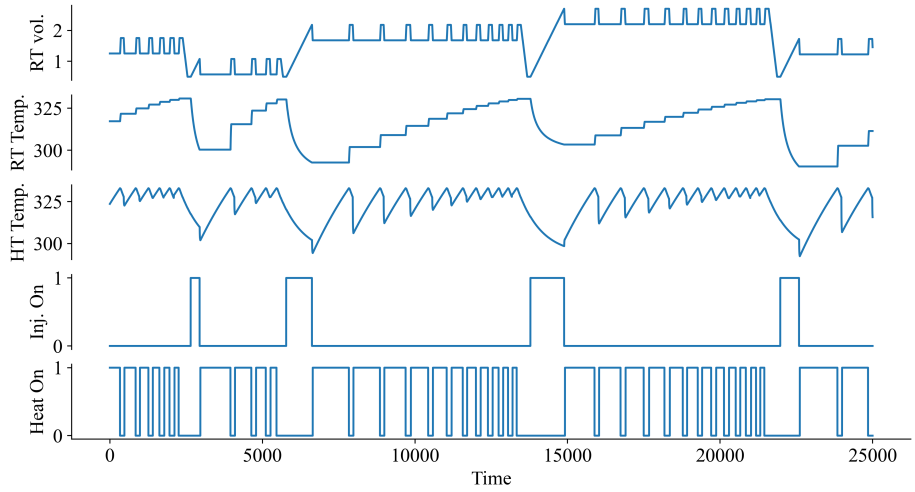


Figure 8.2.2: Example of the five most influential parameters from the GHL dataset [59]. HT is the heating tank where the gasoil is heated and RT is the receiving tank where the gasoil will be pumped to and from while being heated in portions. The temperatures are measured in Kelvin and the volume is measured in L.

in a tank or the pumping rate are modified. The dataset consists of one long multivariate time series where no attack is applied (normal behavior) of length 1,535,118 and 48 independent time series of length approximately 205,000 that each has an anomaly reported. The authors gathered a total of 270 parameters but only used the five most influential parameters for their detection algorithm. These five parameters will also be used in this work. See Figure 8.2.2 for an example of these five time series parameters.

Because our goal is to find anomalies and not attacks per se, a thorough manual relabeling has been carried out based on the behavior of the normal time series. This entails labeling longer-lasting effects of an attack as outlying instead of only the attack itself and additionally labeling instances if they were observed rarely (or never in the normal time series).

8.3 Related Work

Because this paper touches upon multiple research problems, many related works have previously been conducted. In this section, we aim to give a short overview

of all the work that this paper has been built upon toward the overall aim of detecting and understanding faults or anomalies in cabled broadband networks.

8.3.1 Fault detection in HFC networks

Many attempts have been made to properly detect or localize faults in the networks but some of these use FBC data. Zhu et al. used a 1D CNN to detect a set of known anomalies in the FBC data (for a single time point) in a supervised way [203]. Gibellini & Righetti used unsupervised clustering to group modems with similar behavior in order to use topological information to determine the location of a given fault [65]. Williams derived a test procedure for detecting a specific type of fault often occurring in HFC networks based on knowledge about the manifestation of the fault [194]. FBC data, however, is inconvenient as it requires a tremendous amount of space to store. Additionally, network owners rarely have access to a fully known and updated topology.

Other works use a dataset more similar to the one provided to us. Of those, some use trouble tickets registered by the network owner to detect faults. Heiler et al. use the trouble tickets in a supervised way to detect outliers on the aggregated data from all modems under a single amplifier using different ML models such as logistic regression and NNs [71]. Hu et al. use the trouble tickets as hints by calculating thresholds that maximize the density of incoming tickets at times where this threshold is exceeded [77]. As already mentioned, trouble tickets are believed to be prone to uncertainty which means that the performance of a model using those will be limited - especially if understanding the outliers while taking into account the contextual information is the goal.

Some works do not use trouble tickets in developing their models. In both the works by Lartey and by Rupe, technical knowledge of the manifestation of specific types of errors is used to derive hard thresholds for the parameters [94, 159]. Simple thresholds, however, are believed to be unreliable and cause too many alarms to overwhelm technicians. Simakovic & Cica detect hard failures (total loss of connectivity) by looking at sudden drops in the number of users connected to a common amplifier thus constituting a problem [173]. Hard failures, however, are often easier to detect and act on, whereas the remaining errors are more difficult to identify while still resulting in impaired connections for customers.

8.3.2 Anomaly Detection in Multivariate Time Series

Anomaly detection in time series is the process of tagging anomalous behavior within a certain time frame [101]. This field is critical for detecting unexpected changes, flaws, or unusual occurrences that may signify significant events or issues. It is widely employed in several industries, including telecommunication, manufacturing, and healthcare. A variety of methods of detecting anomalies in data have been developed.

Statistical Methods encompass Parametric Models, Non-Parametric Models, and Similarity-Based Methods. Parametric models assume data originates from a known distribution, such as distribution-based models [196], multinomial distributions for categorical data [168], Markovian models for sequential data [168, 109], and hybrid models combining multiple distributions [2]. These models are effective for large datasets due to their consistent complexity and quick evaluations but depend on accurate distribution assumptions and can struggle with multidimensional data. Non-parametric models, unlike parametric models, do not assume a specific data distribution, offering more flexibility [92]. They adapt their complexity to the data size and intricacy, with examples including histogram analysis [9] and kernel density estimators (KDE)[168]. Parzen window methods use Gaussian kernels at each data point to estimate overall density with a single variance hyper-parameter[25]. Depth-based methods identify anomalies by considering data points' positions in multidimensional space, with lower depth points seen as anomalies [160]. Similarity-Based Methods include Distance-Based, focusing on proximity between data points like k-nearest neighbors [144, 70], Dissim distance [123], and Dynamic Time Warping distance [190, 106], and Density-Based, examining local data density, such as the Local Outlier Factor method [196], Local Sparsity Coefficient [110], and Multi-Granularity Deviation Factor [131].

Clustering-Based Methods detect anomalies by identifying data points that do not fit well into clusters, typically in two phases [1]. First, a clustering method such as K-Means [200], Fuzzy C-Means (FCM)[179], or other techniques[137, 107, 156] is applied to generate subsequences of multivariate time series. In the second phase, an anomaly score is calculated based on how well subsequences fit the clusters, often using the distance between subsequences and cluster centers [86]. Other approaches involve computing weighted Euclidean distances, using an improved ant colony method for clustering [153, 46], and a novel method that reduces time complexity by using K-Means to convert a correlation matrix into clusters, estimating anomaly scores from multivariate normal distributions [141]. These models do not require prior data knowledge but rely on identified clustering centers and often face high time and space complexity.

Classification-Based Methods use labeled data to distinguish between normal and anomalous instances, making them a supervised approach. A classifier is trained on a dataset of normal instances and assigns an anomaly score to each instance in the testing set [28]. These methods identify anomalies based on patterns learned from the training data. Examples include using a linear regression model followed by a Bayesian maximum likelihood classifier to detect anomalies [4, 78]. The k-means clustering algorithm and its variations, like fuzzy c-means and probabilistic c-means, are also used in this context [10, 169, 83]. Other models include Self-Organizing Maps (SOM), Expectation-Maximization, Find Out, CLAD, and CBLOF [73]. These methods generally offer higher detection accuracy than unsupervised detectors but require time-consuming data collection and labeling. And we do not have accurate labels, so these methods are not the best option for this case.

Transformation-Based Methods simplify data complexity to facilitate anomaly detection. Common techniques include time series projection and independent component analysis (ICA) to reduce dimensionality and speed up computation [72, 5]. Another method involves using principal component analysis (PCA) to measure novelty dissimilarity in multivariate time series, combining distance, rotation, and variance components [133]. These methods can simplify analysis and enhance computational efficiency but risk losing critical information, potentially reducing anomaly detection accuracy.

Modeling-Based Methods, or **Machine Learning-Based Methods**, include supervised, unsupervised, and semi-supervised approaches that use advanced techniques to identify anomalies by modeling intricate data relationships and temporal dependencies. These methods are highly flexible and adaptable, making them suitable for complex datasets. Examples include using a Hidden Markov Model (HMM) for anomaly detection in multivariate time series [102], and enhancing HMM with PCA, FCM, Sugeno Integral, and Choquet Integral [103]. Another model learns graphical models of Granger causality and uses Kullback–Leibler (KL) divergence to compute anomaly scores with a multi-scale convolutional recurrent encoder–decoder [142, 201]. Cheng et al. proposed a weighted graph representation for multivariate time series, using the RBF function for similarity [5, 56]. Neural networks, known for their robustness and accuracy, are used for forecasting medical time-series data and stock market prices [20, 21]. Multi-Layer Perceptron (MLP) models learn complex time dependencies, while Recurrent Neural Networks (RNNs), including Long Short Term Memory (LSTM) networks, excel in time-series forecasting [22, 26, 177, 79]. Temporal Convolution Networks (TCN) address some limitations of LSTM in modeling time series [7, 183]. Despite their advantages, these methods require significant computational resources and time to train and implement effectively [38].

Most of the described and existing techniques primarily focus on univariate time series (UTS), analyzing only a single variable and often overlooking the complexities of multivariate time series (MTS). This narrow focus simplifies anomaly detection but fails to address the intricacies involved in handling multiple variables simultaneously, which significantly increases the difficulty of the detection process. Furthermore, these techniques predominantly target point anomalies and frequently disregard the contextual information that is crucial for accurate anomaly detection. The lack of attention to context and the relationships among variables in MTS limits the effectiveness of these methodologies thus highlighting the need for more advanced approaches that can manage the complexity and provide a more comprehensive analysis.

8.3.3 Normalizing Flows for time series

Normalizing flows are generative models that transform complex real-world data distributions into simpler, known distributions, like the Gaussian (normal) distribution, through a sequence of easily invertible and differentiable transformation functions. Initially, Rezende et al. (2015) introduced a novel method for specifying approximate posterior distributions for variational inference by learning transformations of simple densities into complex ones by using a Normalizing Flow [154]. Dinh et al. (2016) introduced RealNVP, a foundational normalizing flow architecture for density estimation, which efficiently transforms data by stacking a sequence of invertible bijective transformation functions, called coupling layers, that apply affine transformations of scale and shift to subsets of a first data dimensions based on the remaining data dimensions [48]. The NICE model, a predecessor of RealNVP, employed a series of invertible transformations known as additive coupling layers, which focus on preserving volume through simpler affine transformations without the scale term [47].

Various extensions and improvements were proposed such as the Glow model that builds on NICE and RealNVP by introducing invertible 1×1 convolutions to replace the reverse permutation operation, enhancing the expressiveness and efficiency of the generative process while simplifying the architecture for synthesizing high-resolution natural images [88]. The Masked Autoregressive Flow (MAF) by Papamakarios et al. (2017), a generalization of the RealNVP approach, views an autoregressive model as a normalizing flow, using the Masked Autoencoder for Distribution Estimation (MADE) to build the transformation layer as an autoregressive neural network. This approach enables fast density evaluations and training on parallel computing architectures, though it is computationally expensive to invert for sampling high-dimensional data [132]. Hoogeboom et al. (2019) extend the Glow model by generalizing 1×1 convolutions to more flexible invertible $d \times d$, where d represents the kernel size, oper-

ating on both channel and spatial axes. This approach demonstrated through chaining specific autoregressive convolutions and decoupling in the frequency domain, significantly enhances the performance of generative flow models, especially on galaxy images [74]. These foundational works on normalizing flows laid the groundwork for the field, yet none specifically addressed their application to multivariate time series or time series in general. For that purpose, other research has emerged to fill this gap.

Rasul et al. (2021) introduced one of the first time series forecasting models using conditional normalizing flows. The work models the multivariate temporal dynamics of time series via an autoregressive deep learning model, where the data distribution is represented by a conditional normalizing flow. This approach combined the extrapolation strength of autoregressive models with the flexibility and high-dimensional distribution modeling capabilities of normalizing flows [149]. Graph-Augmented Normalizing Flows (GANFs) enhanced normalizing flows for anomaly detection in multiple time series by incorporating graph structure learning with Bayesian networks. This approach uses a directed acyclic graph (DAG) to model causal relationships, factorizing the joint distribution into conditional densities, which addresses high dimensionality and interdependency challenges, making GANFs effective for detecting anomalies in low-density regions [39]. Guan et al. (2023) introduced the Attention Factorization Normalizing Flow (AFNF) algorithm for unsupervised multivariate time series anomaly detection, utilizing a factorization strategy in the time and attribute dimensions to convert joint probabilities into manageable conditional probabilities. Then, the conditional normalizing flow (MAP) was used to evaluate the conditional density produced by the factorization and thus perform anomaly detection [68].

8.4 Methodology

In this section, we present our proposed framework for modeling the distribution of systematic behavior in a time series dataset. We first present our proposed approach for turning a set of time series windows into a latent representation and subsequently into their densities. Additionally, we present an algorithm for iteratively down-weighing outliers allowing for the learned distribution to model only the systematic behavior. Lastly, we present an algorithm for estimating p-values for a single observation (a time multivariate time series window from a specific modem) based on the estimated cumulative density function of the learned densities.

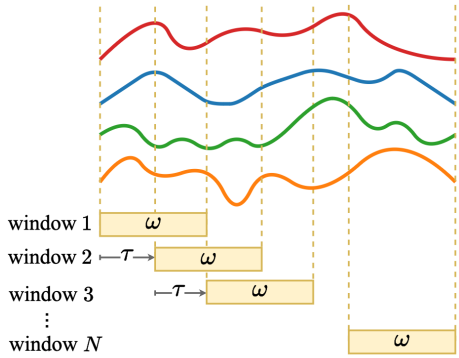


Figure 8.4.1: Depiction of the Window Creation Process for Time-Series Data. The colored lines represent different features data streams over time, with each dashed line indicating the start of a new time interval. Overlapping windows of size ω are created at intervals of τ for subsequent analysis, allowing for continuous monitoring and processing of the data streams.

8.4.1 Modeling Time Series Behavior

We first apply an AE based on one-dimensional convolutions that encodes the multivariate time series into a vector of embeddings while preserving contextual information [93]. Subsequently, we train an NF to accurately model the complex density of the resulting latent space. The AE requires a fixed-size input, thus preprocessing of the time series into windows is required. This does, however, make the method ideal for post-hoc analysis of the representation of the windows. Preprocessing the input time series into a set of windows introduces two hyperparameters: window size, ω, and the stride, τ (see Figure 8.4.1). Together, these parameters determine the amount of overlap or distance between consecutive windows, which in turn affects how independent observations from the same time series instance become.

Let $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N$ be the individual observations consisting of multivariate time-series windows of size $\omega \times \Delta$, where ω is the window size and Δ is the number of dimensions in the input. Notice that windows from different modems are created individually and concatenated to form a single set of time series windows. The AE is trained by minimizing the reconstruction error:

$$\underset{\phi,\theta}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \Big(\boldsymbol{X}_i - \underbrace{D_\theta \left(E_\phi \left(\boldsymbol{X}_i \right) \right)}_{\hat{\boldsymbol{X}}_i} \Big)^2 \tag{8.1}$$

$$\text{where} \quad E_\phi : \mathbb{R}^{\omega \times \Delta} \to \mathbb{R}^{\delta}, \quad D_\theta : \mathbb{R}^{\delta} \to \mathbb{R}^{\omega \times \Delta}$$

E_ϕ is the encoder designed to transform the input time series into a latent vector of length δ and D_θ is the decoder designed to transform a δ-dimensional vector into a multivariate time series of the same size as the input. Ideally, δ should be chosen small in order for the latent space to be a lower-dimensional representation of the input time series:

$$E_\phi(\boldsymbol{X}_i) = \boldsymbol{e}_i \tag{8.2}$$

After having learned the AE, the encoder is used to calculate the δ-dimensional embedding for each observation. We then train an NF to learn the complex distribution of this embedding space based on the Masked Autoregressive Flow (MAF) model introduced by Papamakarios et al. in 2017 [132]. In this type of model, a set of invertible transformations, $f_1, f_2, \ldots, f_K$ are sequentially applied to the data from a multivariate standard Gaussian distribution. The transformations will successively turn the standard Gaussian distributed data into the complex distribution of the original data space. That means that in our case:

$$\boldsymbol{e} = f_K \circ f_{K-1} \circ \cdots \circ f_1(\boldsymbol{z}) \qquad \text{where} \quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{8.3}$$

Ensuring the invertibility of these functions means that we can calculate the density of an observation in the original space by applying the change-of-variable formula. Let $\boldsymbol{u}_k$ be the output of f_k which means that $\boldsymbol{u}_K = \boldsymbol{e}$ and let $\boldsymbol{u}_0 = \boldsymbol{z}$. Then we have:

$$\begin{aligned} p(\boldsymbol{e}) &= p(\boldsymbol{z}) \prod_{k=1}^{K} \left| \det \frac{\partial f_k(\boldsymbol{u}_{k-1})}{\partial \boldsymbol{u}_{k-1}} \right|^{-1} \\ &\Updownarrow \\ \log p(\boldsymbol{e}) &= \log p(\boldsymbol{z}) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k(\boldsymbol{u}_{k-1})}{\partial \boldsymbol{u}_{k-1}} \right| \end{aligned} \tag{8.4}$$

This means that we calculate exact probabilities using only the simple distribution of $\boldsymbol{z}$ and by calculating the derivatives of the transformations applied to it. Additionally, we can sample from the original (input) space by generating random numbers from the simple distribution, Z, and applying the sequence of transformations.

In the MAF model presented in [132], each of the invertible transformations f_k is based on an autoregressive model on the input vector $\boldsymbol{u}_k$ in which the jth conditional is parametrized as follows:

$$p(u_{k,j} | \boldsymbol{u}_{k,1:j-1}) = \mathcal{N}\left(u_{k,j} \,|\, t_{k,j}(\boldsymbol{u}_{k,1:j-1}), \left(e^{s_{k,j}(\boldsymbol{u}_{k,1:j-1})} \right)^2 \right) \tag{8.5}$$

Where $t_{k,j}$ and $s_{k,j}$ are scalar functions computing, respectively, the translation and scale of the given conditional given all previous values of the vector. This

means that we can generate data from the modeled distribution using random data from the previous distribution in the sequence using the following recursion:

$$u_{k,j} = u_{k-1,j} \cdot e^{s_{k,j}(\boldsymbol{u}_{k,1:j-1})} + t_{k,j}(\boldsymbol{u}_{k,1:j-1}) \tag{8.6}$$

Which also gives us the transformation $\boldsymbol{u}_k = f_k(\boldsymbol{u}_{k-1})$ that is easily inverted using the following recursion:

$$u_{k-1,j} = (u_{k,j} - t_{k,j}(\boldsymbol{u}_{k,1:j-1})) \cdot e^{-s_{k,j}(\boldsymbol{u}_{k,1:j-1})} \tag{8.7}$$

Because the conditional functions are built from simple scaling and translations, the determinant of the derivative of the transformations can be easily calculated by multiplying the scaling factors:

$$\left| \det \frac{\partial f_k(\boldsymbol{u}_{k-1})}{\partial \boldsymbol{u}_{k-1}} \right| = \exp \sum_j s_{k,j}(\boldsymbol{u}_{k,1:j-1}) \tag{8.8}$$

This means that we do not need the derivates of $s_{k,j}$ and $t_{k,j}$ meaning that these can be arbitrary complex neural network functions with parameters $\xi_{k,j}$ and $\psi_{k,j}$, respectively. By substituting (8.8) into (8.4), the density of an observation is easily computed and used to train the NF by minimizing the log-likelihood across all observations in the training set:

$$\arg\min_{\mathcal{S},\mathcal{T}} -\sum_{i=1}^{N} \log p(\boldsymbol{e}_i) \tag{8.9}$$

where $\mathcal{S} = \{\xi_{k,j} \,|\, k = 1 \dots K, j = 1 \dots \delta\}$ and $\mathcal{T} = \{\psi_{k,j} \,|\, k = 1 \dots K, j = 1 \dots \delta\}$ are the weights of the NF. Our framework for modeling the density of time series behavior is illustrated in Figure 8.4.2.

8.4.2 Learning only the systematic behavior

Though the NF has proven good at estimating densities of complex distributions, it does this using a relatively naïve objective. Namely, trying to force the whole dataset into a simple and well-known distribution like the Gaussian. This means that outliers or OOD observations will also be forced into this distribution, possibly limiting the quality of the modeling of the systematic behavior which is often more desirable.

We propose iteratively downweighing low-density observations in order to allow the NF to learn only the systematic behavior of the time series. In this way, relatively unlikely, but unproblematic behavior will be more appropriately encoded

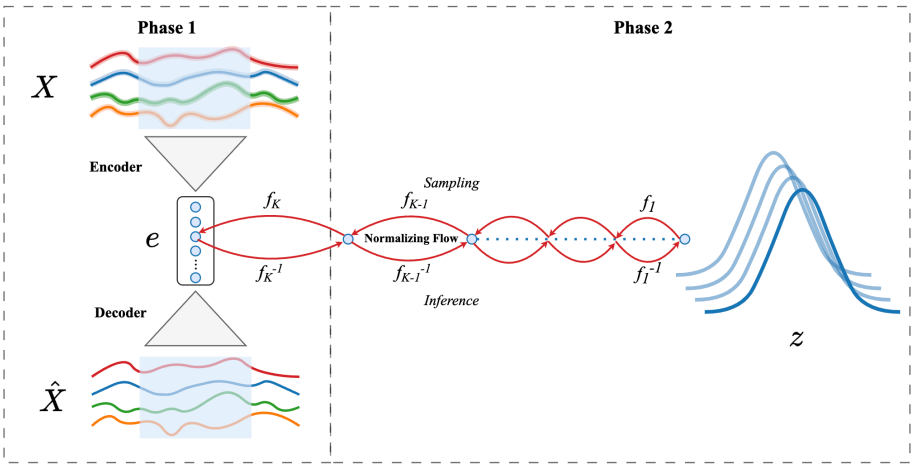


Figure 8.4.2: Illustration of our proposed framework consisting of two phases. In Phase 1, an AE is trained to accurately model the input time series windows (appropriately preprocessed from the full data see illustration in Figure 8.4.1) using a latent vector. In Phase 2, the density of the distribution of these latent vectors is estimated using an NF.

into the low-density zones of the distribution, while the OOD observations will be truly outlying (having near-zero probability).

Let $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_N$ be the latent vectors of the encoded time series achieved by (8.2) and let $\boldsymbol{w}_0 = (1, 1, \ldots, 1)$ be the vector of length N initiated to be ones. We propose learning the NF by minimizing the negative log-likelihood as given in (8.9) but also using the weight vector:

$$\underset{\mathcal{S},\mathcal{T}}{\arg\min} -\sum_{i=1}^{N} \log p(\boldsymbol{e}_i) \cdot w_i \tag{8.10}$$

At each iteration of the algorithm, after training the NF until convergence, we propose identifying the fraction, ϵ, of the observations with the lowest densities. We propose decreasing the weights of these observations by a constant value, γ. In other words, we let $\boldsymbol{d} = (p(\boldsymbol{e}_1), p(\boldsymbol{e}_2), \ldots, p(\boldsymbol{e}_N))$ be the vector of densities for all observations and use this to find the ϵth quantile, $d_{(\lfloor \epsilon N \rfloor)}$. Then we update the weights accordingly:

$$w_j^* = \begin{cases} \max(w_j - \gamma, 0) & \text{if } p(\boldsymbol{e}_j) \leq d_{(\lfloor \epsilon N \rfloor)} \\ w_j & \text{otherwise} \end{cases} \tag{8.11}$$

These updated weights are then used to train a new and randomly initiated NF by optimizing (8.10). This iterative process is continued until the fraction of

observations that have a zero weight passes some predetermined threshold, α. It should be mentioned that observations with zero weight should not be counted when calculating the quantile of the densities. Pseudo-code for the algorithm is provided in Algorithm 8.1.

Algorithm 8.1 Iterative downweighing algorithm

procedure SYSTEMATICBEHAVIOR($\boldsymbol{E} = [\boldsymbol{e}_1^\top, \boldsymbol{e}_2^\top, \ldots, \boldsymbol{e}_N^\top], \epsilon, \gamma, \alpha$)
  $\boldsymbol{w} = [w_1, w_2, \ldots, w_N] \leftarrow [1, 1, \cdots, 1]$ ▷ Initialize the weight vector
  $N_0 \leftarrow 0$ ▷ Number of zero-weight observations
  while $N_0 < \alpha N$ **do**
    $\mathcal{S} \leftarrow$ Initialize randomly ▷ Initialize weights of the scaling models of the NF
    $\mathcal{T} \leftarrow$ Initialize randomly ▷ Initialize weights of the translation models of the NF
    $\mathcal{S}, \mathcal{T} \leftarrow \arg\min_{\{\mathcal{S},\mathcal{T}\}} -\sum_{i=1}^{N} \log p(\boldsymbol{e}_i) \cdot w_i$ ▷ Train NF model until convergence
    $\boldsymbol{d} \leftarrow [p(\boldsymbol{e}_1), p(\boldsymbol{e}_2), \ldots, p(\boldsymbol{e}_N)]$ ▷ Calculate densities using the NF model
    $q_\epsilon \leftarrow d_{(\lfloor \epsilon N + N_0 \rfloor)}$ ▷ Calculate the ϵth quantile excluding zero weight observations
    for $i = 1 \ldots N$ **do**
      if $d_i \leq q_\epsilon$ **then**
        $w_i \leftarrow \max(w_i - \gamma, 0)$ ▷ Update the weight if the density is under the threshold
        if $w_i = 0$ **then**
          $N_0 = N_0 + 1$
        end if
      end if
    end for
  end while
  return $\mathcal{S}, \mathcal{T}$
end procedure

8.4.3 Estimating p-values using Bootstrapping

Due to the complexity and discontinuity of the distributions that the NF learns, densities are not readily interpretable. We propose an algorithm based on boot-

strapping for calculating p-values from observation densities (or log densities) for later interpretation. Assuming that the density of the complex distribution is well modeled by the NF, we can use the NF to generate random samples from it including densities in the original space. Generating a very large number of samples in the original space enables us to estimate the one-dimensional CDF, F_D of the density distribution, D. This function can later be used as a p-value estimator when given a density. For an accurate estimate of the CDF of the densities, the number of random samples, U, should be chosen big and significantly bigger than the number of observations, $U >> N$. The algorithm is outlined in Algorithm 8.2.

Algorithm 8.2 P-value estimation algorithm

procedure pValueEstimator($\mathcal{S}, \mathcal{T}, K, \delta, U$)
  $\tilde{\boldsymbol{U}}_0 \leftarrow$ sample $U \times \delta$ values from $\mathcal{N}(0, 1^2)$
  for $i = 1 \dots K$ **do**
    $\tilde{\boldsymbol{U}}_i \leftarrow$ perform recursion, (8.6), using $\tilde{\boldsymbol{U}}_{i-1}$ and weights in $\mathcal{S}$ and $\mathcal{T}$
  end for
  $\boldsymbol{d} \leftarrow \log\left(\mathcal{N}\left(\tilde{\boldsymbol{U}}_0 \,|\, \boldsymbol{0}^\delta, \boldsymbol{I}^{\delta\times\delta}\right)\right) - \sum_{i=1}^{K}\sum_{j=i}^{\delta} s_{k,j}(\boldsymbol{u}_{k,1:j-1})$
    ▷ Combining (8.4) and (8.8)
  $\boldsymbol{d} \leftarrow \texttt{sort}(\boldsymbol{d})$
    ▷ Sorting the samples from the density distribution, D
  $\tilde{F}_D(d) = P(D \leq d) \leftarrow \frac{1}{U}\sum_U I(\boldsymbol{d} \leq d)$
    ▷ Estimate the CDF using the indicator function, I
  return $\tilde{F}_D$
end procedure

8.5 Experiments and Results

In this section, we discuss the experiments used to validate the proposed method and report the results. In sections 8.5.1 to 8.5.3 we validate our proposed approach on the GHL dataset due to the ground truth being known. In section 8.5.4 we perform an explorative analysis on the broadband dataset from TDC NET.

Table 8.5.1: Performance of the learned density of our proposed framework as an estimator for outlyingness reported as the mean and standard deviation over six repetitions.

Curve	AUC
ROC	0.963 (±0.016)
PR	0.731 (±0.077)

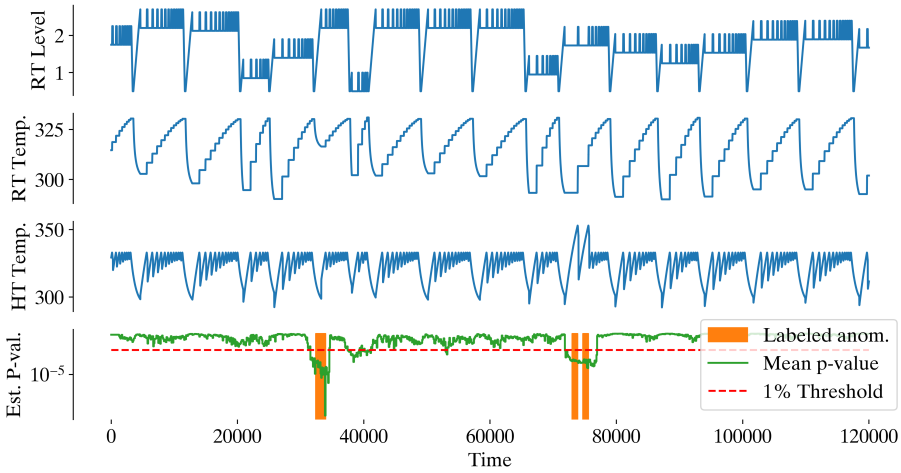


Figure 8.5.1: Example of three parameters of the GHL dataset plotted in time along with the estimated p-values plotted on a log scale including ground truth labels.

8.5.1 Density as an Estimator

We validate the learned densities of our framework on the GHL dataset with known labels (without weighing as described in (8.9)). We report the classification performance using density as an estimator by reporting the Receiver-Operator Characteristic (ROC) Area Under the Curve (AUC) and Precision-Recall (PR) AUC, respectively, as the mean and standard deviation over six repetitions with random initialization of the NF. Table 8.5.1 shows the results that show that our learned densities can be used to detect anomalies quite accurately. Figure 8.5.1 shows an example of a time series plotted with the estimated p-values resulting after applying Algorithm 8.2. This plot also shows that the density is a good estimator for outlyingness including an interpretation in the shape of p-values that are very low when an outlier is present.

8.5.2 Systematic Behavior Algorithm

We validate our proposed algorithm for learning only the systematic behavior on the (relabeled) GHL dataset. We find the optimal configuration of the AE to learn representative embeddings. Subsequently, we apply our proposed approach (Algorithm 8.1) using hyperparameters $\epsilon = 0.01$ (fraction of down-weighted observations) and $\gamma = 0.1$ (constant weight decrease). We execute six repetitions of our algorithm with new and random initializations for each one. After every iteration in the algorithm, we report the performance of the intermediate model by calculating the Area Under the Curve (AUC) of the Precision-Recall (PR) curve when using, respectively, the learned density and the Mahalanobis distance ([117]) in the normalized ($\boldsymbol{z}$) space as predictors of outlyingness. We consider an observation to be anomalous if any of the time points in the corresponding time series window is labeled as being outlying. We use the PR AUC due to the imbalance of the classes in the dataset (4.3% of the observations are labeled as anomalous). Figure 8.5.2 shows the performance of the intermediate models as a function of iterations along with the fraction of observations excluded after each one. From the figure, it is evident that the Mahalanobis distance becomes an increasingly better estimator for outlyingness effectively meaning that outliers are pushed away from the (high-dimensional) bell curve in the normalized space. This happens seemingly without a loss of performance of the density as an estimator. The figure also shows that the performance is maximized not long after the actual fraction of anomalies in the dataset have been excluded by the algorithm whereafter the PR AUC of the density seems to fall slightly. The performance of the algorithm does, however, seem to be rather robust to the number of iterations.

For one of the repetitions of the above results, we further plot the evolution of Mahalanobis distances for different degrees of anomalous behavior. We do this by splitting the time series windows into one of four groups based on the fraction of time points labeled as anomalous. Figure 8.5.3 shows the evolution of Mahalanobis distances for the four groups as a function of the iteration. From the figure it is clear that non-anomalous observations keep having a low Mahalanobis distance, meaning that they remain in the center of the bell curve. Interestingly, partially anomalous observations are pushed far away from the center while entirely anomalous observations seem to remain in the tails of the bell curve in the normalized $\boldsymbol{z}$-space. This could suggest that changes from normal to abnormal behavior or vice versa are more unlikely, thus being pushed further away while entirely abnormal observations are kept as more systematic abnormal behavior. It further seems that the non-anomalous observations are pushed closer to the center of the distribution. This is, however, more a result of the distribution used to calculate the Mahalanobis distance changing characteristics.

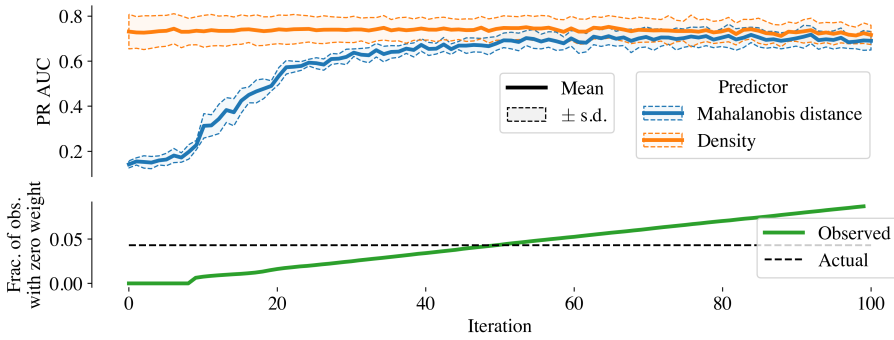


Figure 8.5.2: Performance of learned density and Mahalanobis distance in the normalized (Gaussian) space as predictors for abnormality on the GHL dataset. PR AUCs are reported as the mean and standard deviation over six repetitions. The bottom plot shows the fraction of corresponding observations found to have a weight of zero as a function of the iterations. Reported as the mean over six runs. The uncertainty has been omitted due to insignificant fluctuation.

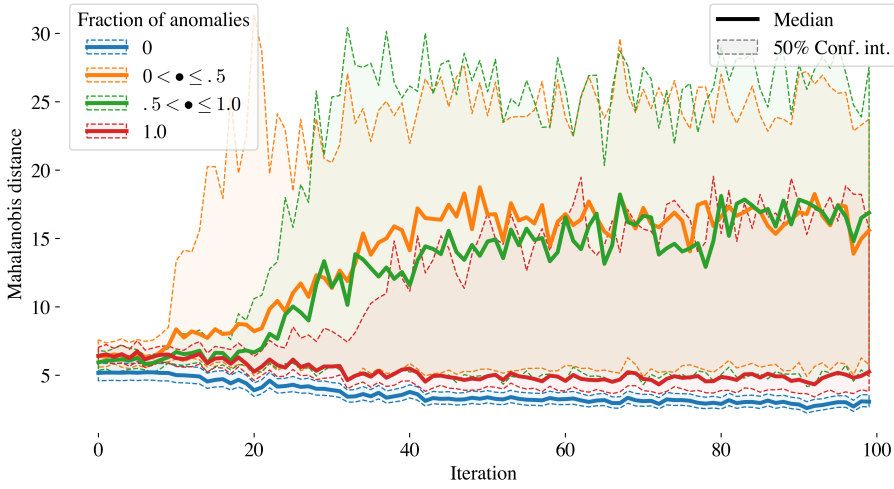


Figure 8.5.3: Grouped distributions of Mahalanobis distances in the normalized space as a function of iterations of the proposed method on the GHL dataset. Reported as the median in the group based on the degree of abnormality. The intervals (shaded areas) are bounded by the 25th and 75th quantiles, respectively.

8.5.3 Clustering Abnormal Behavior

It is evident that our systematic behavior algorithm causes outliers to be pushed out of the distribution. If we assume that different types of anomalies exist that have each their unique characteristic, it would mean that the outliers would be close in the embedding space ($\boldsymbol{e}$) and consequently also in the normalized ($\boldsymbol{z}$) space. This further means that the outliers being pushed out by the algorithm will arrange themselves in clusters in the space around the bell curve. We use a Principal Component Analysis [116] (PCA) because the observations normalized by the learned flow no longer follow a multivariate standard normal distribution. Hence, we expect to localize outliers and respective clusters easily in a lower dimensional representation using a few principal components.

Using the GHL dataset, we perform a PCA analysis on the normalized embeddings using the NF learned by applying Algorithm 8.1 with 80 iterations. Figure 8.5.4 (left) shows a scatter plot of the two most influential principal components, out of approximately six significant components including six highlighted example observations visually identified to belong to one of three clusters. The three first parameters of the GHL dataset are plotted for each of these three groups including some time points before and after for visualization and context purposes. We visually identify the anomalies prone to each of the three groups. In the blue group, the RT level is raised causing the temperature of the RT to drop more than usual. In the orange group, the temperature of the HT is unusually raised. In the green group, the temperature of the RT is not decreased as much as usual after the target temperature has been reached. These findings seem promising for clustering anomalous behavior which can subsequently be used for root cause analysis.

8.5.4 Explorative Analysis of the Broadband Dataset

We train a model using our proposed framework, however, making sure that phase one (the AE) reaches appropriate performance keeping in mind that real data is often noisy and thus harder to capture in a latent representation. An example of a subset of the variables from a time series from the TDC NET dataset is shown in Figure 8.5.5. The figure shows that our model accurately captures that the modem has some connectivity issues at the later time points. Since neither the MTC or the DS MER variables seem outside of the ordinary, it seems that the modem is struggling with the US connection. This is also evident from the fluctuations in the US SNR parameter during the same period.

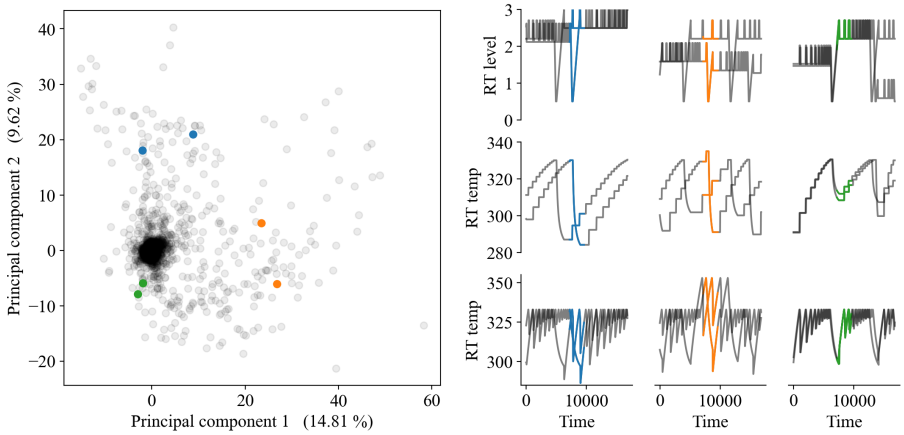


Figure 8.5.4: Six examples of time series windows from the GHL dataset (right) from three different clusters identified visually in the scatter plot (left) showing the two most influential principal components of the PCA, explaining 14.81%, respectively, 9.62% of the total variation in the data. Notice that the observations correspond only to the colored part of the time series and have been plotted including time points both before and after for visualization purposes. The anomaly of the blue group is a raised RT level, the orange group, a heightened temperature in the HT, and the green group, an RT that is not fully cooled down.

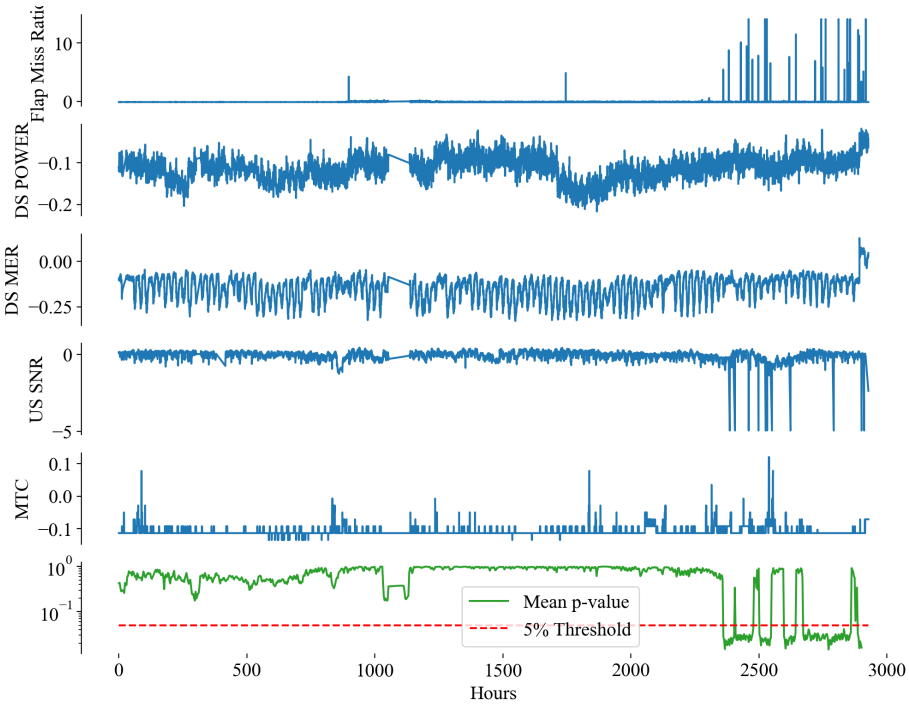


Figure 8.5.5: Example of the time series from a modem from the broadband dataset of TDC NET including estimated p-values calculated using a sliding window. It seems that the given modems has a slightly impaired connection in the later time points which is accurately captured by our model. P-values are estimated using Algorithm 8.2.

We additionally deploy our systematic behavior algorithm to the TDC NET dataset. Since we do not have a good indication of the amount of errors present in the data, we choose to stop after 14 iterations at which just under 1 % of the observations have been removed. As for the GHL dataset, we assume that the excluded observations fall outside the bell curve and arrange themselves in clusters. To analyze these clusters we perform a PCA analysis on the normalized embeddings of the observations. Figure 8.5.6 shows the two most influential components plotted against each other of approximately six significant components (left). From this plot, a number of anomalous groups have been visually identified and example time series plotted (Figure 8.5.6 right) on a number of informative variables. It seems that each group corresponds to a certain type of fault or anomalous behavior. The blue group seems to correspond to the sudden improvement of the signal which is visible in both the FLAP parameters and the jitter. This means that the modem goes from being unresponsive and taking a long time to transmit messages to work well. The opposite is the case for the yellow group which seems to correspond to the sudden occurrence of an impairment of the signal. Lastly, the green group corresponds to a peak in the number of FLAP misses (modem becomes unresponsive) along with a sudden fall in the US SNR. This could mean that the US path (cabling) is impaired or noisy, hence does not allow for the modem to transmit messages effectively.

8.6 Discussion

Our results are promising meaning that our two-phased approach seems to be able to accurately capture the distribution of the behavior of a multivariate time series. This is both evident from the results provided in Table 8.5.1 that show promising classification performance when distinguishing between normal and abnormal behavior and from Figure 8.5.1 that illustrates the estimated p-value as a function of time. Our p-value estimation algorithm allows for interpretation of the densities learned by our model using a statistical context but making no assumptions apart from the NF model having accurately learned the density of the distribution. Our systematic behavior algorithm seems to be accurately penalizing outliers which is evident from both Figures 8.5.2 and 8.5.3 where it is shown that the performance of the Mahalanobis distance of the observations in the normalized space as a predictor for outlyingness increases as a function of the number of iterations used in the algorithm. This is additionally accomplished without a decrease in the performance of the density as a predictor of the same outcome. From Figures 8.5.4 and 8.5.6 we observe that the outliers that are pushed away from the bell curve arrange themselves in clusters corresponding potentially to the underlying root cause responsible for their outlyingness. Actual root cause analysis would have to be performed subsequently in cooper-

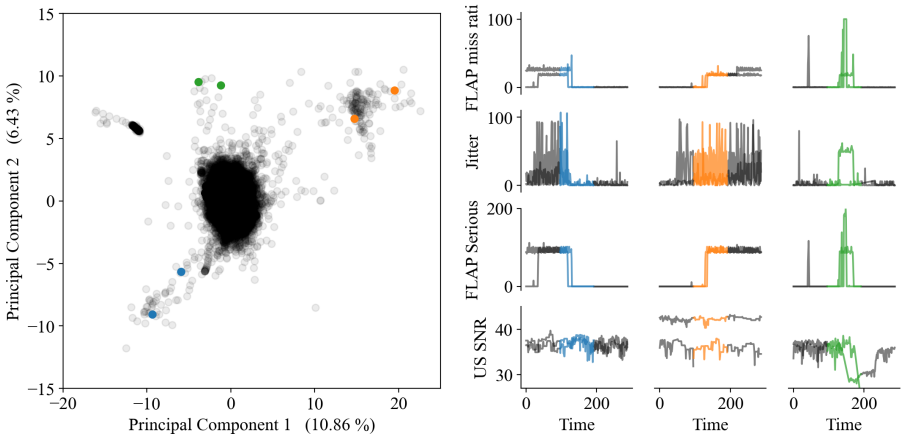


Figure 8.5.6: Six examples of broadband time series windows (right) from three different groups of outliers visually identified from the scatter (left) showing the loadings of the two most influential principal components of the applied PCA, explaining 10.86%, respectively, 6.34% of the total variation. Notice that the observations correspond only to the colored part of the time series that have been plotted including some observations both before and after for visualization purposes. It seems that the blue group corresponds to the disappearance of an impaired signal (maybe due to reparation), the yellow to a sudden occurrence of an impaired signal, and the green to peaks in the number of FLAP misses (modem becomes unresponsive) along with a decrease in the US SNR. The concentrated group in the upper left corner corresponds to constant time series due to long sequences of missing values for the upstream parameters.

ation with domain experts to validate the findings. Especially in Figure 8.5.6 we also see how our proposed approach preserves contextual information since it can distinguish between the appearance and disappearance of a fault in the given window.

Though this paper shows promising results, training the models and identifying the underlying root causes are not trivial tasks. Multiple assumptions need to be made and various hyper-parameters need to be set that are not easy to optimize - especially not when there is no ground truth and the domain knowledge of the data is limited. Some of the more crucial parameters have to do with the pre-processing of the data since the length of the time series windows, ω, becomes the scale at which the behavior is modeled. The window size needs to be chosen such that the occurrence of an error does not become irrelevant in terms of the full window while still containing enough information to be able to make it out. Therefore a good understanding of the domain of the time series is valuable. The window stride parameter, τ, determines the correlation between consecutive windows. If data are abundant, τ can be chosen such that there is a gap between windows for minimum correlation between observations. On the other hand, if the amount of data is limited, correlation in the form of overlap can be introduced to increase the number of resulting windows. However, making τ too small resulting in many repeated data points could increase the risk of overfitting. When pre-processing the input time series from multiple sensors one also needs to consider that these time series can be correlated in time or can be prone to different systematic behavior. If data from multiple sources is included, the overall behavior of all the sources is modeled simultaneously which could help unfold errors common to all sensors while sacrificing accurate modeling of the behavior of the individual sources.

Our proposed systematic behavior algorithm seems robust to hyper-parameter values. Lower values of ϵ and γ give observations more potential to stay in the distribution while taking longer (more iterations) to obtain a fraction of α observations excluded. α itself is another hyper-parameter that is very hard to choose without knowing how many of the time points are actually anomalous. From Figure 8.5.2 it seems that α should be chosen to be close to the actual fraction of time points or perhaps a bit higher to optimize the performance of the algorithm. However, again the algorithm seems robust to higher values of α.

Our two-phase approach to learning the distribution of behavior of multivariate time series assumes that it is sufficient to learn a single AE that provides an accurate latent representation of the time series. This AE should be chosen complex enough to accurately capture the important fluctuations while preventing overfitting. The architecture could benefit from considering multiple scales as proposed by Shi et al. in 2022 [170] which we leave for future work.

8.7 Conclusion

We propose a novel framework for estimating the density of the behavior of multivariate time series to be used for remote anomaly detection in broadband networks. The proposed framework consists of two phases. In phase one, a latent representation of the distribution of time series windows is learned using an AE based on one-dimensional convolutions for preserving contextual information. In phase two, we apply an NF to learn the complex distribution of the latent space.

Our approach shows promising results with the learned density achieving a mean PR-AUC of 0.731 on the publically available GHL dataset when used as an estimator to distinguish anomalies from normal behavior in multivariate time series whilst preserving contextual information. We propose an algorithm for estimating the p-value of an observed window enabling statistical interpretation and meaningful visualizations. We use this to perform an explorative analysis of time series data gathered remotely from an actual broadband network.

Furthermore, we utilize the two-phased structure of our approach to develop an algorithm for learning only the systematic behavior in a dataset. This is done by iteratively down-weighting observations falling into low-density regions of the distribution. We show using both datasets that this algorithm not only pushes the outliers out of the high-dimensional bell curve in the normalized space (the NF base representation) but also that the outliers arrange themselves in clusters. These clusters can be used to group similar abnormal behavior and subsequently be connected to root causes, for instance, in cooperation with domain experts—creating immense value for the network owners.

Acknowledgement

This project was supported by Innovation Fund Denmark, project 0224-00055B, GREENFORCE.

Appendices

8.A Overview of parameters in the TDC NET dataset

In the following, we provide an overview of the different parameters in the TDC NET dataset that were used for the analysis. More information on the parameters and the technology can be found in [158].

`FLAP_HIT_MISS_RATIO` Ratio of flapping messages that are read. If the CMC response is completely missing or takes more than 25 ms, flap misses increase by one, whereas a successful response increases the hits by 1. Provided as a ratio (percentage).

`PACKET_LOSS_RATE` Ratio (percentage) of packets that have been lost during transmission.

`FLAP_INS` Total accumulated attempts to re-register with the CMC outside the defined configuration, which is generally unwanted and could point to communication problems. Interpretation: **Normal:** < 2, **Major: 2-5, Critical:** > 5 (accumulated during 24 hours).

`FLAP_POWER_ADJUSTMENT` Total accumulated automatic power adjustments of the modem due to unstable signal levels indicating problems in the network/return path. Interpretation: **Normal:** < 20, **Major: 20-50, Critical:** > 50 (accumulated during a period of 24 hours).

`JITTER` Variation in packet delay given in milliseconds [ms]. Interpretation: **Normal:** < 30, **Major:** $30 - 150$, **Critical:** > 150.

`LATENCY` How long it takes for a package to arrive after having been requested [ms]. Interpretation: **Normal:** < 100, **Major:** $100 - 150$, **Critical:** > 150.

`FLAP_CRC_FAILURE` Total accumulated Cyclic Redundancy Checks (CRC). Modems with high CRC errors have bad upstream paths. Interpretation: **Normal:** < 50, **Major:** $50 - 150$, **Critical:** > 150 (accumulated ins 24 hours).

`FLAP_SERIOUS_MISS_NUM_DIFF` The number of serious FLAP misses since the last poll.

`UNSTABLE_CONNECTION_NUM_DIFF` Accumulated Sum of metrics that constitutes an unstable connection. Interpretation: **Normal:** < 5, **Major:** $5 - 15$, **Critical:** > 15.

`MISSING_CM` Whether data from one of the three combined tables was missing (1) or not (0).

`DS_AVG_RX_POWER` Average downstream received power during the last poll period [dB].

`DS_AVG_RX_MER` Average downstream received Modulation Error Ratio (MER) during the last poll period [dB].

`DS_SUM_CODEWORDS` Total sum of received codewords during the last poll cycle.

`DS_SUM_CORRECTED_CODEWORDS` Total sum of codewords that were successfully corrected during the last poll cycle. Interpretation: **Normal** $< 20\%$**, Major:** $20 - 50\%$**, Critical:** $> 50\%$.

`DS_SUM_UNCORRECTED_CODEWORDS` Total sum of codewords that could not be successfully corrected during the last poll cycle. Interpretation: **Normal:** $< 1\%$**, Major:** $1 - 3\%$**, Critical:** $> 3\%$.

`DS_UNCORRECTED_CODEWORD_PCT` Total ratio of codewords that could not be corrected given as a percentage of the total codewords. Interpretation: see above.

`DS_MIN_RX_POWER` Minimum downstream received power during the last poll cycle [dB].

`DS_MAX_RX_POWER` Maximum downstream received power during the last poll cycle [dB].

`DS_MIN_RX_MER` Minimum downstream Modulation Error Ratio (MER) during the last poll cycle [dB].

`DS_MAX_RX_MER` Maximum downstream Modulation Error Ratio (MER) during the last poll cycle [dB].

`MISSING_DS` Whether data from one of the three combined tables was missing (1) or not (0).

`MTR` Main Tap Ratio - the ratio of energy in the main tap to the energy in all other taps combined. Measured in decibels [dB].

`SNR_US` Signal-to-Noise Ratio (SNR) in the upstream signal. Measured in decibels [dB]. Aggregated at the CMC level, hence quite similar for all modems.

`MTC` Main Tap Compression - This metric is given by the ratio of the energy in all taps to the main tap energy and is measured in decibels [dB]. An MTC ratio greater than 2 dB may suggest that equalization compensation can no longer be successfully achieved. Interpretation: **Normal:** < 1 **dB, Major:** $1 - 2$ **dB, Critical:** > 2 **dB**.

`MISSING_US` Whether data from one of the three combined tables was missing (1) or not (0).

Chapter 9

Paper C

Topology Reconstruction in Telecommunication Networks: Embedding Operations Research within Deep Learning

Tobias Engelhardt Rasmussen[a,*] · Siv Sørensen[b]
David Pisinger[b] · Thomas Martini Jørgensen[a] · Andreas Baum[a]

[a] Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark
[b] Technical University of Denmark, DTU Management, Kgs. Lyngby, Denmark
[*] Corresponding author

Publication Status: Paper is submitted and under peer review for publication in *Computers and Operations Research*.

Abstract

We consider the task of reconstructing the cabling arrangements of *last-mile* telecommunication networks using customer modem data. In such networks, downstream data traverses from a source node down through the branches of the tree network to a set of customer leaf nodes. Each modem monitors the quality of received data using a series of continuous data metrics. The state of the data, when it reaches a modem, is contingent upon the path it traverses through the network and can be affected by, e.g., corroded cable connectors.

We train an encoder to identify irregular inherited *events* in modem quality data, such as network faults, and encode them as discrete data sequences for each modem. Specifically, the encoding scheme is obtained by using unsupervised contrastive learning, where a Siamese neural network is trained on a positive (true) topology, its modem data, and a set of negative (false) topologies. The weights of the Siamese network are continuously updated based on a new modified version of the Maximum Parsimony optimality criterion. This approach essentially integrates an optimization problem directly into a deep learning loss function.

We evaluate the encoder's performance on simulated data instances with randomly added events. The performance of the encoder is tested both on its ability to extract and encode events, as well as whether the encoded data sequences lead to accurate topology reconstructions under the modified version of the Maximum Parsimony optimality criterion.

Promising computational results are reported for trees of a varying number of internal nodes up to 20, where the encoder identifies a high percentage of simulated events, leading to nearly perfect topology reconstruction. Overall, these results affirm the potential of embedding an optimization problem into a deep learning loss function, unveiling many interesting topics for further research.

9.1 Introduction

Broadband technology is one of the most widespread technologies for providing internet access to paying customers. According to CableLabs, that is one of the leading providers of broadband technology, this technology was the most

accessible in both Europe and the US in 2016 [18]. The relatively cheap cost of deployment and continuous innovation suggest that it is going to be an important part of digital infrastructure for many years to come.

Hybrid-Fiber Coaxial (HFC) networks are used to connect users to the internet using coaxial cables in which data is transmitted using radio frequency (RF) signals. Each HFC network is connected to the optical backbone grid through a local singular conversion node, called a CMC (Coaxial Media Converter), where the signal is converted from optical to electronic form as it transitions from the fiber optic backbone to the coaxial network, and vice versa. Within a given HFC network each customer is connected to the CMC through a sequence of cable amplifiers and cable splitters arranged in a tree-like structure allowing the HFC network to cover a local area by gradual branching. In this network architecture, the end connections to the homes of the customers are represented by the leaf nodes of the tree. The internal nodes are the cable splitters, while the root node is represented by the CMC. Lastly, the edges in the architecture correspond to cable connections [135].

The RF signal is transmitted using metal-insulated copper wires due to its vulnerability to outside signal interference that would otherwise impair the connection of the customer. In general, HFC networks are prone to a range of errors, making maintenance an important part of daily operation. For instance, general degradation can cause one or more customers to have reduced connectivity [199], and weather conditions have been shown to affect the quality of the network [195, 129].

Due to digitization, the complexity of the HFC setup, and time constraints, the infrastructure owner only has partial knowledge of the cabling in most HFC networks, hereinafter referred to as the *topology*. This is problematic since knowing the topology is crucial for network maintenance staff when localizing and resolving network faults. Incomplete topology records lead to a significant increase in both resolution and driving time for the maintenance staff. In their review paper on network monitoring, Lee et. al emphasizes that a known and fully updated topology is important for accurate problem detection [97]. Simaković et. al mentioned that a known topology is needed for monitoring non-intelligent devices, such as amplifiers [174]. Additionally, both Heiler et. al and Simakovic et. al listed a full topology mapping as one of the data requirements for their approach to root-cause identification and localization of network failures [71, 173], respectively, while Hu et al. planned to utilize the topology for localizing network errors in future work for their HFC anomaly detection algorithm [77]. Due to the scale and the constant changes to HFC networks and digital privacy legislation, it is not viable to manually reconstruct the missing topology data. This makes an automated approach to solving the problem valuable to network owners.

A recent study shows that it is conceptually possible to infer a missing topology of e.g. an HFC network with high accuracy using discrete time series data collected at modem level only [138]. The authors achieve this by employing a multi-objective approach, incorporating the *Maximum Parsimony* optimality criterion [61] and accounting for the geographic distances between components of the network. The most *parsimonious* tree topology is the topology that best explains the observed leaf (customer) data in terms of the fewest data mutations. This means that since the RF signal is initially transmitted through a singular root node, one can reconstruct the most probable evolution of the signal throughout the network branches at each time step and count the number of signal mutations [61, 69]. This value is often referred to as the *parsimony score* of a proposed tree topology. The term originates from computational biology, particularly within the field of reconstructing family trees, scientifically known as phylogenetic trees.

The limitation of the above-mentioned study [138] is that its proposed method only works for discrete time series data. Moreover, the observed data must contain distinguishable data mutations such as faults, hereinafter *events*, happening on the branches of the network over time. Nonetheless, modems predominantly capture continuous time series data, and the process of identifying inherited events within the data and subsequently encoding them into a discrete format is not a straightforward task. As a result, the study [138] focused solely on simulated data.

Inspired by Pisinger & Sørensen [138], consequently, our work focuses on extracting significant discrete events from continuous time series data, a task which, in this setting, remains unsolved. While, theoretically, the parsimony algorithm only necessitates the encoded sequences to be discrete, we consider strictly binary events in this work for the sake of simplicity.

Because the parsimony score counts the number of mutations needed to explain the observed leaf data, learning an encoding scheme for the time series using the parsimony score as a loss function is not feasible. No mutations are needed to explain identical data, thus, in this setting, the model will simply encode the same data point for all leaf nodes and arrive at a trivial model, hereinafter *zero-encoder*. Instead, our work proposes using the parsimony score to train a binary event encoder using a contrastive approach [95]. We consider the case where the encoder is constructed using a 1D convolutional neural network architecture, which is then treated as a Siamese network [91]. Hereby, the encoding scheme is similarly applied to all customer modems, meaning that we learn a general encoder.

As a preliminary investigation, we aim to analyze the feasibility of the proposed approach using simplified simulated data, where all possible topologies

are known and unique events are simulated on all network edges. We experiment with different data simulation parameters and investigate how the effect of the contrastive approach depends on the time series properties (e.g. length and fault duration) along with network characteristics (e.g. the number of nodes in the network). Moreover, we propose a modified version of the parsimony algorithm to improve the uniqueness of the optimal solution. This modification is based on a new assumption about the state of the data signal at the root-node level. We show evidence that strongly suggests that the modified version leads to better and more unambiguous topology predictions.

The main **contributions** of this paper are:

- An alternative formulation of the tree topology reconstruction problem using leaf node data, also known as the problem of *inferring phylogenies* in Computational Biology. The new formulation adds the assumption that one of the discrete data states is a base-level state representing the root node behavior and indicating an unaffected data point.
- A new algorithm that returns the best possible parsimony score with respect to the maximum parsimony optimality criterion, given the above-mentioned alternative formulation.
- A new conjecture based on computational experiments, which states that the true topology, given the new alternative formulation, will always yield a uniquely best parsimony score, provided that at least one isolated distinguishable event occurs on every edge in the tree.
- A novel approach leveraging the synergy between the fields of operations research and deep learning, achieved by integrating an optimization algorithm into a deep learning loss function.
- A contrastive approach for training an encoder that is able to infer events in continuous time series data that are informative in the new alternative parsimony setting. The end-to-end approach considers the new parsimony algorithm, the true topology of the network, and a set of wrong topologies.
- A stochastic version of the parsimony algorithm to be used with automatic differentiation during the learning stage. This relaxation allows each data point to be represented as a set of probabilities for the different possible states, rather than a certain discrete state.
- An approach to simulating customer modem time series data given the new alternative parsimony setting, to be used in testing of the proposed contrastive method.

The paper is **organized** as follows: In the following two sections, we equip the reader with the relevant theory behind the HFC network setting and the maximum parsimony criterion. Section 9.4 provides an overview of state-of-the-art research related to our problem. In section 9.5 we formally define our problem along with the difficulties associated with it, and in section 9.6 we present our methodology. Section section 9.7 defines the computational experiments we perform, the evaluation metrics used to assess the computational results, and the data simulation scheme for the experiments. The results of the experiments are presented in section 9.8. Lastly, we discuss the results, the proposed methodology, assumptions, applicability, and future work in section 9.9. Section 9.10 concludes the paper.

9.2 The HFC network

The backbone (country-spanning) network consists of a set of Optical Line Terminals (OLTs) between which information is transmitted using optical fiber technology in which optical wires are used to transmit data using flickering light [155]. Locally, each OLT is connected to a set of Coaxial Media Converters (CMCs) in which the signal is converted from a fiber signal to an electronic RF signal as part of the transition from the backbone network to the coaxial network, hereinafter *last-mile network*, and vice versa. These CMCs are used as local singular nodes to which all customer modems in a local area are connected. This connection enables data transfer from the backbone internet to the customer modem, i.e. downstream (DS), and the other way, i.e. upstream (US). Each customer modem is connected to a CMC through a sequence of cable amplifiers (amplifying the RF signal) and cable splitters linked by coaxial cables. Usually, the splitters and amplifiers of a last-mile network are sitting close together in the same street cabinet, making it reasonable to illustrate both the amplification and splitting of the signal by a singular process (node). An illustration of a last-mile HFC network is provided in Figure 9.2.1.

9.2.1 Time series data

Due to the many potential challenges faced in the daily operation of the HFC network, multiple metrics are sampled regularly from each of the modems in the network. These metrics are specified in the (Data-Over-Cable Service Interface Specifications (DOCSIS) 3.0 [184] and include Modulation Error Ratio (MER), power levels, the number of bits received during the last poll period, the number of corrupted bits, and other metrics that are used to perform surveillance of

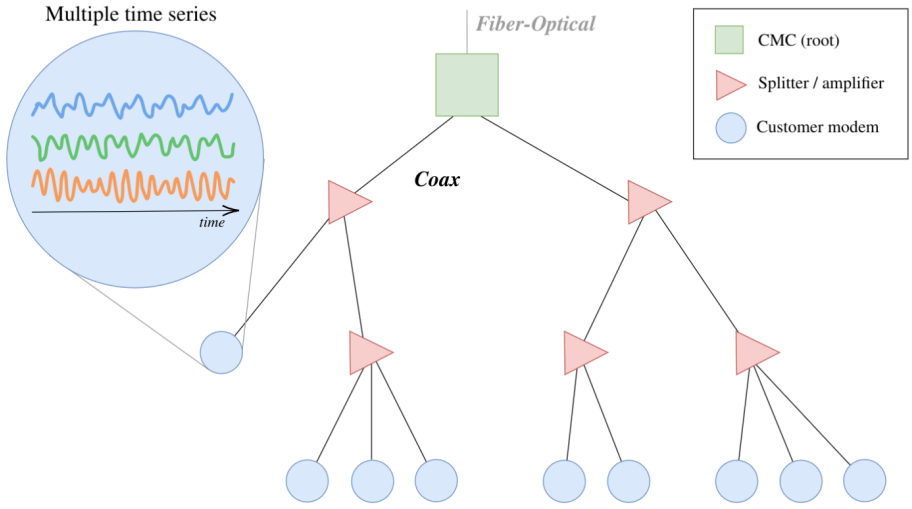


Figure 9.2.1: Illustration of a last-mile HFC network setup. The CMC acts as a local singular node connecting a local network to the backbone internet using fiber-optic technology. Each customer modem is connected to the local CMC through a sequence of amplifiers/splitters (red triangles) linked by coaxial cables. The splitters and amplifiers are usually found close to each other in the same street cabinet. Each customer modem collects a set of performance metrics over time that can be used for monitoring purposes.

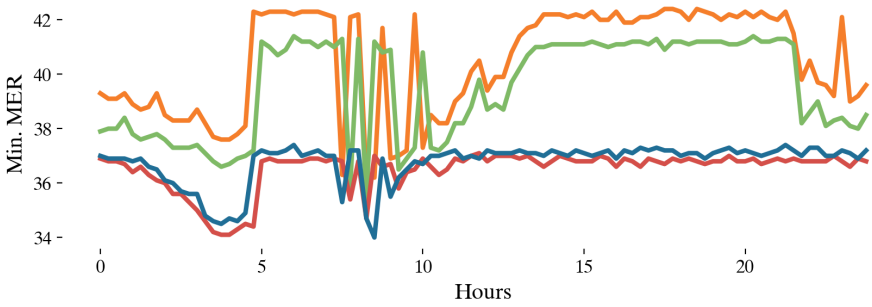


Figure 9.2.2: Example of 24 hours of time series data from four different customer modems consisting of minimum MER values during the last 15 minutes. It appears that there is a baseline signal that all modems adhere to. However, during certain periods, some customers experience signal distortion due to specific events. For example, between approximately hours 7 and 8, the signal for all customers is prone to increased variation with varying degrees. Conversely, during hours approximately 10 to 13 and 22 to 24, only the signal for the green and orange customers is affected.

the quality of the signal over time. Typically, these metrics are aggregated through polling, utilizing statistical methods or summation for each polling period, since continuous real-time monitoring is often impractical due to the significant volume of data that would need to be stored. See Figure 9.2.2 for an example of time series data from four different modems in the same HFC network. These time series consist of minimum MER values during the last polling period (approximately 15 minutes) over a total of 24 hours.

9.2.2 Faults and noise propagation

The complexity and vulnerability of especially the coaxial part of the HFC network make it prone to degradation and various types of errors. This means that many approaches have been proposed to identify and locate errors in the network [71, 77, 94, 159, 173, 194]. A typical error influencing broadband networks is the so-called Common Path Distortion (CPD) that is caused by connectors that have been affected by stress or corrosion [164]. In general, problems in the HFC network are characterized by being sporadic and difficult to detect due to the many potential root causes. Those include physical properties in the wires, weather factors like temperature and humidity, outside RF interference, electromagnetic interference, and improper customer equipment [139, 140]. Some

types of errors are due to problems in or near the network amplifiers, meaning that all customer modems that are located topologically beneath a problematic amplifier will be affected in the DS case [203, 62]. An example of this is visible in Figure 9.2.2, where all customers seem to adhere to some baseline signal, but where some or all customers are affected by specific events during periods of time.

9.3 Maximum parsimony

The study of evolutionary relationships, scientifically known as *phylogenetics*, is a heavily researched discipline within biology, dating back to the 1800s when early evolutionists such as Darwin sketched the first evolutionary trees to represent historical relationships among living species [66]. The reconstruction of phylogenetic trees is typically based on genome (DNA/RNA) sequences from a set of living species, sometimes accompanied by some assumed evolutionary model. Here, the set of living species represents the terminal nodes of the phylogenetic tree, the internal nodes represent extinct ancestors, and the branches depict evolutionary relationships and relatedness.

Numerous methods exist for reconstructing the tree topology [58], where the best choice of approach depends on the available data and evolutionary setting. Maximum parsimony is one of the oldest and most applied methods both praised and criticized for its simplicity and lack of assumption involving underlying evolutionary models [176]. The maximum parsimony principle assumes that the tree which provides the simplest evolutionary explanation is the correct one. Specifically, it aims to minimize the number of necessary data mutations across all branches in the tree, enabling the transformation from a shared ancestral data sequence to each of the observed leaf sequences.

Different variations of evolutionary trees occur in many other problems than biology, one being the reconstruction of coax networks using DS data. In the same way that DNA sequences change for each new species over time, the coaxial signal is distorted based on which physical component it encounters on its way from the CMC root node to each of the customer terminal nodes. However, in biology, binary trees are almost exclusively considered and typically in an unrooted state, as deciding which edge to root the tree in, is a difficult problem of its own. In telecommunication networks, the root node (CMC) is a fixed parameter and the networks are general trees.

9.3.1 The uniqueness of the most parsimonious tree

Although the aim is to reconstruct the one true topology, no reconstruction method can do so for all imaginable types of data sets. In many cases, the best tree topology will not be unique and the true tree might not even be amongst the best solutions provided by some chosen reconstruction model [60]. Jussi & Teemu [124] employed a large-scale computational study with simulated phylogenetic data to estimate the probability that maximum parsimony uncovers the true tree topology and concluded that it performs well under a simple data-generating model if the rate of change, i.e. number of events, is sufficiently small and, crucially, if the length of the data sequences are sufficiently long.

In certain special cases, it has been demonstrated that various reconstruction principles, including maximum parsimony, consistently reveal the true tree topology unequivocally. One such example is based on Buneman's Theorem [16]. Here, in the setting of an unrooted binary tree, if the data in the leaf nodes corresponds to exactly one event occurring on each edge at each time step, such that to the left of the event, one single data state is observed in all nodes, and to the right another single state is observed, then the tree which generated the data will be the unique most parsimonious tree. Later, Fischer [60] proved this to also be true in the case where an event is generated on exactly every two pairwise edges.

9.3.2 The hardness of maximum parsimony reconstruction

Finding the most parsimonious tree has been shown to be NP-hard in many of its variants, including the rooted setting [41, 58]. While calculating the parsimony score of a particular tree is achievable in polynomial time, what makes the problem difficult is that in theory, one must consider all possible topologies and their respective parsimony score to determine the most parsimonious tree(s). Given that the number of possible general tree topologies for n nodes is n^{n-2}, also known as Cayley's formula [24], employing a brute force approach quickly becomes infeasible as the tree size increases.

All approaches to computing the best parsimony score of a specific tree are based on dynamic programming, which makes the maximum parsimony approach computationally efficient compared to other tree reconstruction methods [124]. Fitch [61] famously developed the first algorithm for binary trees, Hartigan [69] developed a more general approach not limited to binary trees, and Sankoff's algorithm [162] addresses the weighted variant of the problem, in which a cost matrix is given as input, stating the cost of switching between all

possible pairwise data states.

9.4 State-of-the-art

Even though a known topology is very useful for Internet Service Providers as previously mentioned, the previous research of trying to reconstruct missing topologies is very limited. To the best of our knowledge, the only paper that has been previously published on the matter is from Pisinger & Sørensen in 2024 [138], using the maximum parsimony criterion, as previously discussed in the introduction. Though the maximum parsimony approach to construct and study phylogenetic trees goes back many years as described in section 9.3, the work of Pisinger and Sørensen is believed to be the first time this approach has been applied to the reconstruction of HFC-network topologies. The work had very promising results, however, was only evaluated on discrete simulated data, highlighting the need for a way to extract events from continuous data.

9.4.1 Representation learning in time series

The extraction of discrete events from multiple time series can be understood as equivalent to learning representations of the time series data. This field has only recently gained momentum, due to the complexity of the problem and the incomprehensible nature of time series in general, which makes it hard to interpret the outcome and assess the usefulness of the learned representations. Representations are usually learned in an unsupervised manner, in which an encoder is learned with some unsupervised goal in mind.

Recurrent Neural Networks (RNN) have traditionally been used as an encoder for a range of time series tasks, such as classification and representation learning. For instance, Malhotra et. al used RNNs in an encoder-decoder setting to represent a normal time series behavior and use that to search for anomalies [113], while Malhotra et al. trained a Variational AutoEncoder (VAE) based on RNNs on a set of different datasets to produce a generic time series feature extractor [114].

Others have used different approaches to encode the time series. For instance, Hyvärinen & Morioka [80] divided a time series into a number of chunks with temporal labels and trained a neural network model that would enable the temporal classification of the time series chunks using logistic regression. Lei et

al. used matrix factorization techniques to produce representations that mimic the distances between time series obtained via dynamic time warping [100].

However, recent work suggests that one-dimensional *Convolutional Neural Networks* (CNN) is the state of the art in various time series tasks, while also being relatively easy to train and interpret [8]. This has resulted in multiple studies using CNNs for time series classification [32, 191], but also for representation learning tasks. Emadeldeen et al. used two types of augmentation based on permutation and scaling, respectively, to learn representations that are contrastive with respect to both time and context [53]. Challu et. al used hierarchical latent factors to represent time series at different scales in a generative model [27]. Lastly, Franceschi et. al used exponentially dilated convolutions and a triplet loss to learn representations that are close to those of sub-sequences of an anchor time series, while far from those of other time series [64].

Nevertheless, all the methods mentioned above, as well as many time series representation methods in general, are based on continuous representations, whereas this work requires discrete representations.

9.4.2 Discrete latent representations

A few studies have been conducted where comprehensible discrete representations were the goal. Van den Oord et. al [188] developed a framework for learning discrete representations based on a continuous embedding space with discrete labels, where the embedding nearest to the encoded input would become the discrete latent representation. This approach, however, was not used on time series. Additionally, the predefined size of the embedding space makes it inflexible to inputs of various sizes that time series are likely to constitute.

A recent work carried out by Fortuin et al. studied discrete representations of time series [63] by applying a Self-Organizing Map (SOM) to the latent space of the encoded values in a VAE setting. This method, however, learns single representations for an input consisting of a sliding window of a time series, but for our work, no assumptions can be made on the length of events. Additionally, this method would not guarantee latent representations that are informative in a parsimony setting.

9.5 Problem Statement

We begin this section by formally introducing the problem. Next, we introduce what the most parsimonious tree is and show how the parsimony score is calculated. Thirdly, we introduce the issue of obtaining a unique solution, and lastly, we show how the frequency of events in the modem data complicates the problem further.

9.5.1 Problem

Let graph $G = (V = \{N, M\}, E)$ be a general tree corresponding to a last-mile HFC network with a set, N, of internal nodes (cable splitters and amplifiers), a set, M, of leaf nodes (customers), and a set edges, E (cable connections). Given the tree property of the graph, we have that $|E| = |V| - 1$, where in this work, $|\bullet|$ represents the cardinality of a set.

Every modem $m \in M$ is associated with input data $\boldsymbol{X}_m \in \mathbb{R}^{|F|\times|T|}$ comprising F features, each recorded over a duration of T time steps. The features are various *continuous* metrics describing the quality of the DS signal received by each modem at specific time points.

For each modem, an encoded version of the data is generated using an encoder $f(\boldsymbol{X})$. The encoded data is represented by a $|F'| \times |T'|$ matrix[1] where each element is a *discrete* event in the alphabet of binary states $\Omega = \{0, 1\}$. Each state in theory corresponds to a physical interpretation, but interpretations can be quite abstract. A simple interpretation could be that state 0 may represent a normal signal, while state 1 may indicate an abnormal signal. All results can readily be generalized to include a larger set of states. We also establish an encoded data sequence $\boldsymbol{z} \in \Omega^{|B|}$ for each modem, arranging the encoded matrix sequentially in a single dimension, with $B = F' \times T'$ (Cartesian product). Here, $\boldsymbol{z}$ straightforwardly represents all the encoded features in vectorized form. We denote the stacked collection of all modem sequences in a graph G as $\boldsymbol{Z} = \Omega^{M\times B}$.

Then, the **aim of this paper** is to develop an encoder, $f(\boldsymbol{X})$, that, based on the maximum parsimony principle, can transform continuous modem data, $\boldsymbol{X}$, into a collection of binary data sequences $\boldsymbol{Z}$. This encoder should ensure that the tree topology, from which the modem data originates, will be the singularly

[1] Where $|F'|$ and $|T'|$ are used to specify the dimensions of the outcome of the encoder, which can be—but does not have to be—of the same dimensions as the input data $\boldsymbol{X}_m$.

most parsimonious tree with respect to all the encoded modem data sequences, $\boldsymbol{Z}$.

In this work we assume, without loss of generality, that all leaf connections are known. Consequently, we only consider the set of tree topologies that emerge from reconstructing the *internal edges* of a last-mile network. We make this assumption based on a first-hand account from the biggest telecommunication infrastructure owner in Denmark, TDC NET. Moreover, we also assume that every internal node has at least one child which is a modem (leaf) node. Although this assumption does not hold for all real-life last-mile networks, it is necessary to prevent the emergence of isomorphic topologies resulting from swapping two interchangeable 'modem-less' internal nodes. Lastly, we assume data events occur uniformly distributed across all internal edges and that events propagate undisturbed down through the network.

9.5.2 The most parsimonious tree

The most parsimonious tree is the tree, G, associated with the lowest *parsimony score* $\mathcal{P}(\boldsymbol{Z}, G)$ based on a set of discrete data sequences, $\boldsymbol{Z}$; one sequence $\boldsymbol{z} \in \Omega^{|B|}$ for each modem. Here, the parsimony score of a tree G is given as the sum of the pairwise Hamming distances between all nodes connected by an edge in G. This equates to inferring the data states, z_i, of the internal nodes and thereby minimizing the number of times the root node signal must change through the sequences of nodes in the tree, at every time point, to explain the observed modem data:

$$\mathcal{P}(\boldsymbol{Z}, G) = \left\{ \begin{array}{lll} \min & \sum_{(u,v)\in E} d_H(\boldsymbol{z}_u, \boldsymbol{z}_v) & \\ \text{s.t.} & \boldsymbol{z}_i \in \Omega^{|B|} & \forall \quad i \in N \end{array} \right\} \tag{9.1}$$

where $(u, v) \in E$ are all edges in G between any two nodes u and v, and d_H, otherwise known as the Hamming distance between two vectors, is defined as:

$$d_H\left(\boldsymbol{z}, \boldsymbol{z}^*\right) = \sum_{b=1}^{B} \mathbb{1}_H(z_b, z_b^*) \quad \text{where} \quad \mathbb{1}_H(z_b, z_b^*) = \begin{cases} 0 & \text{if} \quad z_b = z_b^* \\ 1 & \text{otherwise} \end{cases} \tag{9.2}$$

In short, the parsimony score is the sum of all differences in data between any two nodes connected by an edge in G.

The observant reader might notice that $\mathcal{P}(\boldsymbol{Z}, G)$ is not readily computable, since the data sequences $\boldsymbol{z}$ are known only for the modem nodes $u, v \in M$. Luckily, polynomial time algorithms exist for determining which sequence(s) $\boldsymbol{z}$ for each internal node $u, v \in N$ leads to the minimal parsimony score. These algorithms

were briefly discussed in section 9.3. Since we in this work are considering general trees with binary data states, Hartigan's algorithm [69] is the most appropriate choice.

Hartigan's algorithm is a dynamic programming approach where each internal node is systematically considered in a bottom-to-top order, determined by a postorder traversal of G. Furthermore, all parsimony algorithms compute the score for each time point in $\boldsymbol{Z}$ independently before aggregating the score contributions.

For every internal node $v \in V$, the algorithm exclusively assesses the set of direct descendant nodes D_v. Here, the optimal state of node v is determined by the majority state observed in its descendant set, where multiple states can be optimal in the case of a tie. The cost associated with node v reflects the number of descendants assigned a different state than v, each signifying that a data mutation occurs. In other words, the cost corresponds to the number of descendant nodes conflicting with the majority vote.

The parsimony score for the entire tree with respect to a single data point is then obtained by summing the costs of all internal nodes $v \in V$. Alternatively, as this process simultaneously determines the optimal data state(s) of all internal nodes, equation (9.1) presents a different, but equivalent approach, to computing the optimal parsimony score.

Algorithm 9.1 formally outlines the above procedure for a single data point.

9.5.2.1 Example: calculating the optimal parsimony score

Figure 9.5.1 presents a small illustrative example of running the algorithm on a small tree for the first data point in each $\boldsymbol{z}_m, m \in M$. The example tree G in Figure 9.5.1 contains 15 customer nodes (blue), seven amplifier/splitter nodes (red), and one root node, the CMC (green). The example considers a single binary input data point for each modem, displayed next to each blue node. Following the procedure of Algorithm 9.1, the internal nodes are considered one by one with respect to a postorder traversal; C, D, A, E, F, G, B, R.

The optimal parsimony score for the example tree is five. This score can be obtained in two ways; a) by letting the state in all $\{0, 1\}$-nodes be 0 resulting in a mutation on all red edges, and b) by letting the state in all $\{0, 1\}$-nodes be 1 resulting in a mutation on all green edges.

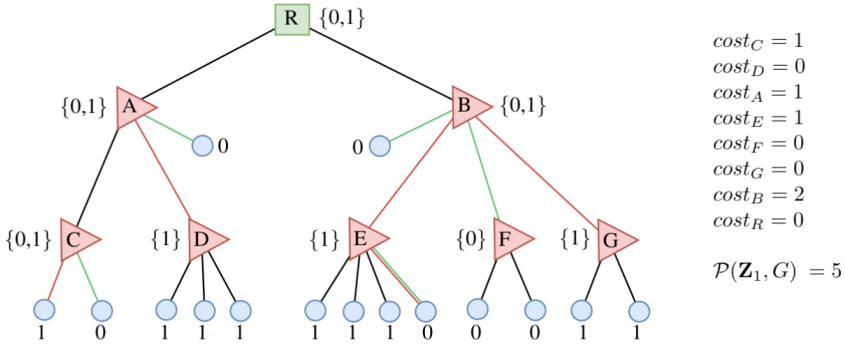


Figure 9.5.1: An example of how to compute the optimal parsimony score for a specific tree G using algorithm 9.1. Here, a single time point in the input data sequences $\boldsymbol{Z}$ is considered. The blue nodes are customers, the red nodes are splitters/amplifiers, and the green node is the CMC. The computed optimal state sets for the internal nodes are listed in curly brackets. Node C has two children with differing states, resulting in a tie for the majority state. Consequently, the optimal state for C can be either state 0 or 1, denoted by $\{0, 1\}$ on the figure. Moreover, the cost of node C is $cost_C = 1$, as one child node will always disagree with C, regardless of whether state 0 or 1 is the true state of C. Node D has three children in which the majority state unanimously is state 1. Thus, the optimal state set for node D is 1 and $cost_D = 0$. Node A has three children, nodes C and D, and one modem child. Out of these three children, state 0 occurs twice (in node C and the modem node), and state 1 also appears twice (in node D and the modem node). Thus, there is a tie for the majority, and the optimal state set of node A becomes $\{0, 1\}$. Regardless of the true state in node A, one child will always disagree, leading to $cost_A = 1$.
The majority state in the children of node E is state 1, but one child disagrees with this state, so the optimal state of E is 1 with $cost_E = 1$. The children of both nodes F and G unanimously agree on the majority state just as in node D. Node B has four children in which both states 0 and 1 appear twice. Consequently, the optimal state set of node B is $\{0, 1\}$ with a cost of 2, as two children will always disagree with the state of B. Lastly, the root node R has two children both with an optimal state set of $\{0, 1\}$, meaning the optimal state set of the root is also $\{0, 1\}$ with $cost_R = 0$.

Continued on the next page

Figure 9.5.1 ***(continued)*:** Summing over all node costs gives an optimal parsimony score for tree G of $\Sigma_{v \in N} cost_v = 5$. However, this does *not* mean that G is the most parsimonious tree for the given modem data. By considering the simplified version of the problem where the topology of only the internal edges is unknown, there are still $|N|^{|N|-2} = 262,144$ possible topology configurations that each potentially could have a better parsimony score.
There exist two reconstructions of the data states in the internal nodes, which both yield a parsimony score of five. Letting the optimal state be 0 in all nodes with an optimal state set of $\{0, 1\}$, leads to the five data mutations illustrated by the red edges. Oppositely, letting the optimal state be 1 in all nodes with an optimal state set of $\{0, 1\}$, leads to the five data mutations illustrated by the green edges.

Algorithm 9.1 Parsimony Score for tree G with modem data, $\boldsymbol{z}$, for a *single time point*

procedure PARSIMONY($G = (V = \{M, N\},\ E)\ ,\ \boldsymbol{z} \in \Omega^{|M|}$)
  $\boldsymbol{z} \leftarrow \left[\boldsymbol{z},\ \text{NULL}^{|N|}\right]$ ▷ Expand $\boldsymbol{z}$ with placeholder for internal nodes
  $\bar{N} \leftarrow$ POSTORDERTRAVERSAL($G[N]$) ▷ "Bottom-up" sorting of internal nodes
  for $v \in \bar{N}$ **do**
    $D_v \leftarrow$ set of child nodes of v
    for $\omega \in \Omega$ **do** ▷ For each character in the set of states, Ω
      $\phi_\omega \leftarrow$ frequency of $\omega \in \boldsymbol{z}(D_v)$
    end for
    $\boldsymbol{z}_v \leftarrow \arg\max_{\omega \in \Omega} (\phi_\omega)$ ▷ Character state(s) of v becomes the most frequent
    $cost_v \leftarrow |D_v| - \max_{\omega \in \Omega} (\phi_\omega)$ ▷ Number of children disagreeing with majority
  end for
  return $\sum_{v \in N} cost_v$
end procedure

9.5.3 Solution uniqueness

In the Maximum parsimony section, we briefly discussed that for unrooted trees, there exist special cases of data generation where the most parsimonious tree is guaranteed to be unique. However, in this section, we demonstrate a counterexample, proving this is *not* the case for rooted trees.

In the unrooted setting, one such special case occurs when exactly one event occurs on every edge at independent points in $\boldsymbol{Z}$, as discovered by Buneman [16]. Figure 9.5.2 (left) illustrates this case for a rooted tree G^p with five internal nodes and modem data sequences of length $|B| = |E| = 4$. At each time point $t \in B$, a new edge e is chosen, and $\boldsymbol{z}[t]$ is assigned state 1 for all modems below edge e. In the figure, the color of the affected data point corresponds to the color of the edge on which the event was simulated. The resulting modem data, $\boldsymbol{Z}$, generated by this approach is depicted in Figure 9.5.2. The optimal parsimony score of tree G^p equals the number of simulated events, namely four. However, as shown on the right in Figure 9.5.2, an alternative tree topology G^n also yields an optimal parsimony score of four for the same input data $\boldsymbol{Z}$, if the root node data is reconstructed as $[0, 1, 0, 0]$.

Although it is unlikely that an encoder can learn to encode a special case of data that generates a unique most parsimonious tree, it remains problematic that it is not even theoretically possible to do so. While it is improbable for an encoder to learn to recognize and encode a special structure in the data and always produce a unique most parsimonious tree, the inability to achieve this even theoretically is a concern. Therefore, we will introduce a new assumption and propose a new modified parsimony approach in the Methodology section to attempt to circumvent this issue.

Lastly, we also find it important to emphasize that it directly follows from the definition of the parsimony score that some form of inheritable event must occur on *every* edge in a tree G, at least once in the modem input data $\boldsymbol{Z}$. If an edge $e = (u, v)$ is never affected by an event, the data signal passing through nodes u and v will be identical. Consequently, replacing edge e with $e' = (v, u)$—that is, switching the order of nodes u and v—will result in a new topology, G', indistinguishable from G. Assuming faults occur at random positions in a network, collecting data over a sufficiently long time period should ensure all edges are affected at least once.

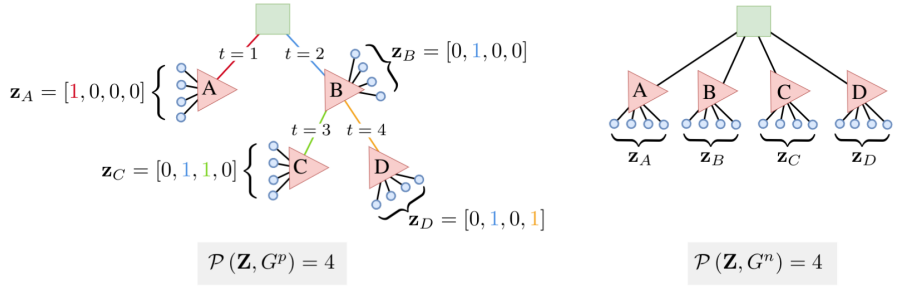


Figure 9.5.2: A counterexample showing that a special data case which results in a unique most parsimonious tree in an unrooted setting [16], does not yield a unique tree in a rooted setting. Here, the left tree is the true tree, G^p, for which an event has been simulated to occur on each internal edge at independent times. Given that there are four internal edges in G^p, the parsimony score of G^p is also four. However, the alternative tree topology to the right, G^n, yields the same optimal parsimony score of four for the same modem data, proving the true tree is not uniquely the most parsimonious.

9.5.4 Events

If $|\Omega|$ different data states are observed at the same time point in $\boldsymbol{Z}$, it will require at least $|\Omega| - 1$ data mutations (events) to explain the data. As an example, in the case of binary data, if both states 0 and 1 are observed in different modems at the same time, then the original state of the signal in the root node must have mutated at least once to explain the two states observed at leaf level. This means that if only singular independent events occur at each data point in $\boldsymbol{Z}$, then there does *not* exist any topology G^n with a better parsimony score than the true topology G^p.

However, if multiple identical events, hereinafter *multi-faults*, occur at the same time, an alternative tree topology G^n can sometimes outperform the true topology G^p, that generated the modem data. Such an example is illustrated in Figure 9.5.3. Here, at time $t = 4$, the last point in $\boldsymbol{Z}$, an event occurs on two different edges, where in both cases the modem data is affected in the same way, i.e. it mutates from state 0 to state 1. As a result, the alternative topology G^n to the right in Figure 9.5.3 is able to group the modems affected by the two simultaneously occurring events under one single edge, (B, D), such that a single event can explain the observed data mutations instead of the two events that occurred in reality.

A set of states more fine-grained than binary, could, in theory, resolve the issue of multi-faults. However, in practice, an encoder would then need to learn not only to identify abnormalities in the modem data but to also categorize or at least tell apart different abnormalities. This task is much more challenging and could easily lead to overfitting.

If in fact, two identical events occur at the same time, i.e. two cables buried next to one another, are both cut through, then the maximum parsimony principle fails. Assuming events are not that frequent, it is much more likely that one single event causes a signal mutation in a set of modems rather than two or more identical events occurring at the exact same time. The optimal tree with respect to the maximum parsimony principle, is the tree that can explain the observed leaf data with the *least* amount of mutations, as formalized in (9.1). Therefore, although the left tree in Figure 9.5.3 is the true tree that generated the data $\boldsymbol{Z}$, according to the maximum parsimony principle, it is more likely that the data originates from the right tree.

Since we in this paper have chosen to work with the maximum parsimony principle, there is not much we can do to combat multi-faults in our methodology. Instead, we have chosen to show the impact on *topology accuracy* (9.9) (defined in a later section) for an increasing amount of multi-faults in the modem data.

9.6 Methodology

In this section, we present our proposed methodology. First, we introduce a new root-node assumption and a modified parsimony algorithm, based on which we also propose a new conjecture. Secondly, we prove that the new modified parsimony algorithm yields the optimal parsimony score given the new root-node assumption. Thirdly, we introduce a contrastive learning approach along with a loss function that is based on the new modified parsimony algorithm; we also introduce a solution to overcoming non-differentiability during the learning phase. Lastly, we introduce a set of neighborhood functions to be used when sampling topologies from the set of all possible topology reconstructions for a given tree size.

9.6.1 Modified Parsimony Algorithm

To circumvent the inability to obtain unique solutions, we propose a new assumption, i.e. that the data states in the root node, $\boldsymbol{z}_{root}$, are known just like

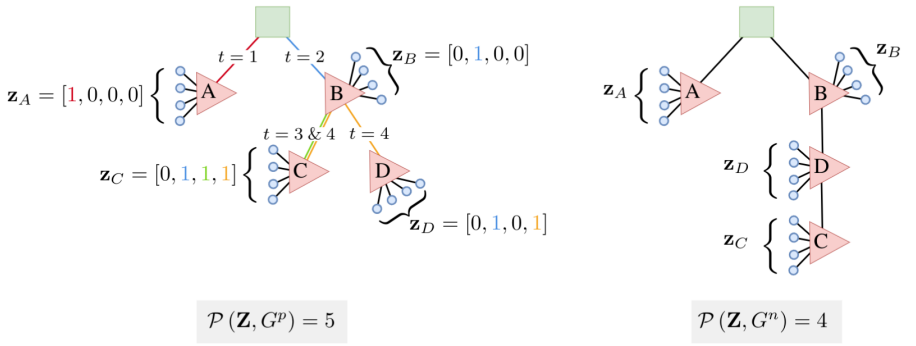


Figure 9.5.3: An illustrative example of the challenge posed by *multi-faults* in a maximum parsimony framework. The network on the left displays the true topology, G^p. On each edge, t indicates when an event occurred on the edge, resulting in a change from the base data state of 0 to 1 in the data sequence of all affected modems. At time $t = 4$, a multi-fault occurs on edge (B, C) and (B, D). The optimal parsimony score of G^p is five: one in each time step with a single fault, and two in the time step with a multi-fault.
In the alternative network topology to the right, G^n, the same modem data yields an optimal parsimony score of four, which is better than the score obtained by the true topology. This is because G^n groups all modems affected by the multi-fault under a single edge, (B, D).

the leaf nodes. At all time points, the root node represents some normal *baseline* signal. Even if the signal received by the root node from the backbone grid at some point t is corrupted, any additional faults in the last-mile network will accumulate on top of this initial corruption acting as the baseline signal at time t.

We propose a new modified parsimony algorithm (algorithm 9.2) identical to the normal parsimony algorithm (algorithm 9.1), except for lines 9–10. Here, when v becomes the root node in the for-loop in line 4, instead of letting the majority state in the children of the root dictate the optimal state of the root, we let the cost of the root be equal to the number of children disagreeing with the presumed input state of the root. This can be formally stated as:

$$\hat{\mathcal{P}}(\boldsymbol{Z}, G) = \left\{ \begin{array}{lll} \min & \mathcal{P}^{\text{sub}}(\boldsymbol{Z}, G) + \sum_{v \in D_{root}} d_H(\boldsymbol{z}_{root}, \boldsymbol{z}_v) & \\ \text{s.t.} & \boldsymbol{z}_i \in \Omega^{|B|} & \forall \quad i \in N/\{root\} \end{array} \right\} \tag{9.3}$$

where $\mathcal{P}^{\text{sub}}(\boldsymbol{Z}, G)$ is the sum of the parsimony scores of all subtrees, $\hat{G}(v)$, rooted in the set of nodes $v \in D_{root}$ which are the direct descendants of the root node. Moreover, $\hat{\boldsymbol{Z}} \subseteq \boldsymbol{Z}$ is the set of discrete data sequences belonging to each modem in subtree $\hat{G}(v)$:

$$\mathcal{P}^{\text{sub}}(\boldsymbol{Z}, G) = \sum_{v \in D_{root}} \mathcal{P}(\hat{\boldsymbol{Z}}, \hat{G}(v)) \tag{9.4}$$

Here, the objective of (9.3) minimizes both the aforementioned sum of subtrees as well as the Hamming distance between the root node to each of its direct descendants. A proof demonstrating that Algorithm 9.2 always yields the optimal parsimony score is enclosed in the supplementary material 9.A.

In Figure 9.5.2, the base state of the root was assumed to be zero at all points, and for every simulated event, we let the data mutate from zero to one. Using the modified algorithm 9.2 on this example, assuming $\boldsymbol{z}_{root} = [0, 0, 0, 0]$, yields a unique most parsimonious score for the true tree to the left in the figure.

9.6.1.1 Uniqueness conjecture

Based on algorithm 9.2, and inspired by the data setting of Buneman [16], we propose the following conjecture:

Conjecture 1 *Given Condition 1 (below), the Modified Parsimony Score algorithm presented in Algorithm 9.2 always yields a unique best parsimony score for the true tree topology.*

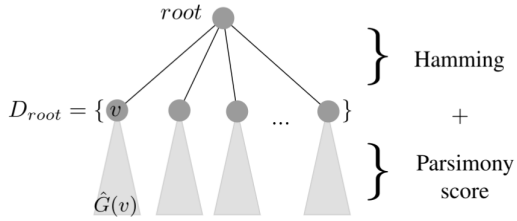


Figure 9.6.1: Illustration to aid understanding of new modified parsimony score and proof of Algorithm 9.2

Algorithm 9.2 Modified Parsimony Score for tree G with modem data $\boldsymbol{z}$ for a single time point

procedure mPARSIMONY($G = (V = \{M, N\},\ E)$, $\boldsymbol{z} \in \Omega^{|M|}$, $\omega_{root} \in \Omega$)
  $\boldsymbol{z} \leftarrow \left[\boldsymbol{z},\ \text{NULL}^{|N|}\right]$ ▷ Expand $\boldsymbol{z}$ with placeholder for internal nodes
  $\bar{N} \leftarrow$ postOrderTraversal($G[N]$) ▷ "Bottom-up" sorting of internal nodes
  for $v \in \bar{N}$ **do**
    $D_v \leftarrow$ set of child nodes of v
    for $\omega \in \Omega$ **do** ▷ For each character in the set of states, Ω
      $\phi_\omega \leftarrow$ frequency of $\omega \in \boldsymbol{z}(D_v)$
    end for
    if v is root **then**
      $cost_v \leftarrow |D_v| - \phi_{\omega=\omega_{root}}$
    else
      $\boldsymbol{z}_v \leftarrow \arg\max_{\omega\in\Omega}(\phi_\omega)$ ▷ Character state(s) of v becomes the most frequent
      $cost_v \leftarrow |D_v| - \max_{\omega\in\Omega}(\phi_\omega)$ ▷ Number of children disagreeing with majority
    end if
  end for
  return $\sum_{v\in N} cost_v$
end procedure

Condition 1 *A tree topology $G = (V = \{M, N\}, E)$ with leaf nodes, M; nodes, $M \cup N$; and edges, E, where the root node and each leaf node contains a discrete data string, $\boldsymbol{z}$ from an alphabet of discrete states, Ω. Every data point in every leaf data string must be inherited from the root through the branch connecting them. If a data point encounters an 'event' through its branch traversal it is modified to a new discrete state $s \in \Omega$, different from the root state. At least one event must occur on each edge of the tree at an independent point in the data string, and no more than one event can occur at the same time. Consequently, the length of each data string, $\boldsymbol{z}$, must be at least $|E|$.*

The validity of the conjecture will be challenged through rigorous simulated tests described in section 9.7.

9.6.2 Contrastive Learning Approach

Since our approach intends to encode the continuous time series data from the customer modems in the network into a sequence of discrete events, there is no definitive ground truth labeling for the output events. This means that the problem will be unsupervised. In the meantime we intend to learn an encoder, f, to be used on any customer modem individually, thus will only depend on the continuous time series data. We will use a contrastive approach, comparing a positive (true) sample to a set of negative (false) samples, based on the parsimony score (see section 9.6.1) to learn an encoder that can extract the most useful events from the time series data in the parsimony setting. This means that we intend the model to encode events with two properties. Firstly, the encoded events should result in the lowest parsimony score when calculated using the true topology (positive sample) as opposed to any other possible topology (negative sample). Secondly, to be able to differentiate between any two topologies, their respective parsimony scores calculated based on the encoded events, should be as different as possible. This could be achieved by minimizing the total parsimony score using the true topology while maximizing a pairwise distance measure between the parsimony scores of any two different topologies. Our contrastive approach is visualized in Figure 9.6.2.

In the following, we describe our approach in the case where the alphabet, Ω, consists of binary events. We choose this for simplicity, not at a loss of generality, as it is straightforward to add an extra dimension for multiple encoded event states. Additionally, the parsimony score is defined for discrete events, not exclusively binary events.

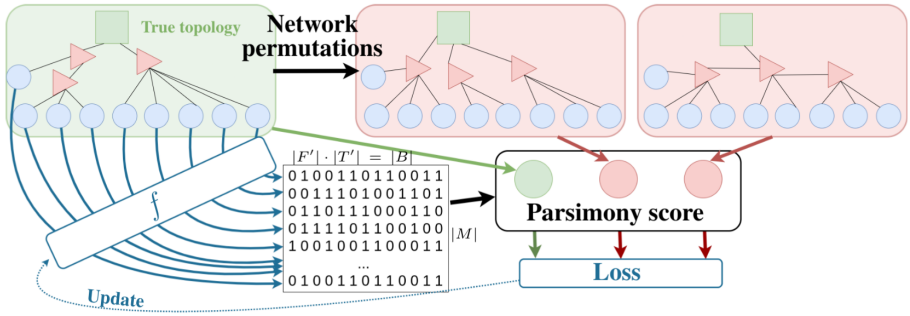


Figure 9.6.2: Illustration of the proposed algorithm. A Siamese network is trained using the parsimony score on a positive (true topology) sample (green) and a set of negative samples (red). The time series from each of the M customers are sent through the same encoder f that encodes continuous time series into an $M \times B$ matrix of binary events. Using this matrix and each of the different topologies, the parsimony scores can be computed. Based on these the loss is calculated and used to update the encoder.

9.6.2.1 Proposed Loss Function

Let $\boldsymbol{X}_m$ be the $|F| \times |T|$ matrix of time series measurements from the mth modem, that is $|F|$ features measured on $|T|$ occasions. Let f be a Siamese neural network that encodes the multiple time series from each customer modem in an HFC network into a matrix (sequence) of events of size $|B| = |F'| \times |T'|$. Because the parsimony algorithm treats each data point individually, each of these matrices, $f(\boldsymbol{X}_m)$, can be flattened and stacked to produce the final encoded event matrix of all modems in the network, $\boldsymbol{Z}'$, of size $|M| \times |B|$. This matrix is given by:

$$\boldsymbol{Z}' = \begin{bmatrix} f(\boldsymbol{X}_1)_1 & f(\boldsymbol{X}_1)_2 & \dots & f(\boldsymbol{X}_1)_{|F'|} \\ f(\boldsymbol{X}_2)_1 & f(\boldsymbol{X}_2)_2 & \dots & f(\boldsymbol{X}_2)_{|F'|} \\ \vdots & & & \\ f(\boldsymbol{X}_{|M|})_1 & f(\boldsymbol{X}_{|M|})_2 & \dots & f(\boldsymbol{X}_{|M|})_{|F'|} \end{bmatrix} \quad ; \quad f(\boldsymbol{X}_m)_j^\top = \begin{bmatrix} f(\boldsymbol{X}_m)_{j,1} \\ f(\boldsymbol{X}_m)_{j,2} \\ \vdots \\ f(\boldsymbol{X}_m)_{j,|T'|} \end{bmatrix} \tag{9.5}$$

where $f(\boldsymbol{X}_m)_j$ is the jth feature in the encoded events matrix measured for modem m. To make the encoded events binary, f should be chosen such that the output values are in the range $[0, 1]$ allowing for subsequent rounding to a binary value:

$$\boldsymbol{Z} = \lfloor \boldsymbol{Z}' \rceil \tag{9.6}$$

Let $\hat{\mathcal{P}}(\boldsymbol{Z}, G)$ be the modified parsimony score (9.3), and $\hat{\boldsymbol{\mathcal{P}}}(\boldsymbol{Z}, G)$ the unsummed *vector* of modified parsimony scores with respect to each time point (column) in the encoded binary events matrix $\boldsymbol{Z}$ using the tree described by $G = (V = \{M, N\}, E)$. We propose the loss function given by:

$$\mathcal{L}(\boldsymbol{Z}, G^p, \mathcal{G}^n) = \frac{\hat{\mathcal{P}}(\boldsymbol{Z}, G^p)}{\frac{|E|}{2} \cdot |B|} + \alpha \sqrt{\frac{1}{K(K+1)} \sum_{g_1 \in \mathcal{G}} \sum_{g_2 \in \mathcal{G} \setminus g_1} \frac{1}{\|\hat{\boldsymbol{\mathcal{P}}}(\boldsymbol{Z}, g_1) - \hat{\boldsymbol{\mathcal{P}}}(\boldsymbol{Z}, g_2)\|_2^2 + 1}} \tag{9.7}$$

where N is the set of internal nodes, G^p is the true topology, $\mathcal{G}^n = \{G_1^n, \ldots, G_K^n\}$ is the set of K negative (sampled) topologies, and $\mathcal{G} = \{G^p\} \cup \mathcal{G}^n$. The first term corresponds to the parsimony score of the true topology and is normalized by the maximum number of data mutations that can occur with respect to the parsimony principle[2]. Meanwhile, it is also natural to normalize the contrastive term by the number of comparisons performed, i.e. the number of contributions in the nested sums; $\frac{1}{2} \cdot K \cdot (K+1)$, and the number of contributions in the sum of squares, $|B|$. Doing this also makes the coefficient α independent of the number of negative samples used in the training loop.

This loss function considers the difference between resulting parsimony scores using all pairwise topology samples, rewarding large score differences and punishing small ones. We calculate these distances in the $|B|$-dimensional space $\left(\mathbb{N}^{|B|}\right)$ allowing for more flexibility in the encoded events.

9.6.2.2 Overcoming the Non-Differentiability

The modified parsimony algorithm described in section 9.6.1 is so far constrained to discrete time series events. Since we use the modified parsimony algorithm directly in our loss function given in (9.7), the rounding operation will be part of the standard forward pass. In order for the back-propagation algorithm to work, gradients need to be computed with respect to all weights in the network individually [157]—also when utilizing automatic differentiation in e.g. pytorch [134]. Due to its nature, the rounding operation is not differentiable because it is not continuous in its domain.

[2]In the case of binary data states, $|\Omega| = 2$, the maximum number of mutations that can occur is $\frac{1}{2}|E|$. For every internal node in a tree, a mutation occurs for each child node disagreeing with the majority state. The majority will always include at least half of the children, thus a mutation can at most occur on half of all edges. It follows from the same argumentation that if $|\Omega| = 3$, the maximum number of mutations is $\frac{2}{3}|E|$, for $|\Omega| = 4$ it is $\frac{3}{4}|E|$, etc.

We overcome this challenge by defining a special forward pass for training in which we let the encoded events take values in $[0, 1]$, thereby representing probabilities. To train the model, we additionally propose a slightly modified version of the parsimony score that allows for continuous events, meaning that the distances in (9.7) will be calculated in $\mathbb{R}_+^{|B|}$. Since all operations will be carried out in the continuous space, this version will allow for the calculation of gradients throughout the full model.

9.6.2.3 Modified Continuous Parsimony Score

The intuition behind the continuous version of the modified parsimony score is to keep track of each event state individually and for each leaf node to encode the probability of the customer modem being in any given event state. We do this by ensuring that the sum of all possible state probabilities sums to one for each customer modem, adhering to the rules of probability theory. The parsimony cost of an internal node will no longer be the number of children disagreeing with the majority, but rather the sum of disbelief (among all the children) in the most probable event state as believed by the children. This means that if every child believes fully in a single (but not necessarily the same) state, the cost will be equivalent to that of the discrete case. Additionally, internal nodes inherit the averaged probabilities for each state from their children, meaning that the degree of belief in an event state will also be inherited and thereby not lost as in the discrete case. The pseudocode for the modified continuous parsimony score is given in Algorithm 9.3. An example of the calculation of the parsimony score for a tree using this probability-based algorithm is shown in Figure 9.6.3.

9.6.3 Neighborhood functions

To generate a set of negative samples $\mathcal{G}^n$, we will need a set of neighborhood functions that can permute some tree G into a negative sample G_1^n through a series of neighborhood moves. Figure 9.6.4 illustrates the three types of neighborhood functions used throughout this paper. These neighborhood functions were originally introduced in [138].

Random swap inserts a new random edge (v, u) and removes the old parent edge of node u, i.e., edge (w, u) . *Subtree shuffle* selects a random sub-tree $\hat{G}(v)$ with k nodes rooted in node v, and shuffles all the edges in $\hat{G}(v)$ at random. *Parent-child swap* swaps the relationship between two node u and v, where v is the parent of u, such that u becomes the parent of v instead.

Algorithm 9.3 Continuous Modified Parsimony Score for tree G with modem data $\boldsymbol{z}$ for a single time point

procedure CMPARSIMONY($G = (V = \{M, N\}, E)$, $\boldsymbol{Z}^* \in \{\boldsymbol{z} \in [0,1]^{|\Omega|} \mid \|\boldsymbol{z}\|_1 = 1\}^{|M|}$, $\omega_{root} \in \Omega$)
  $\boldsymbol{Z}^* \leftarrow \left[\boldsymbol{Z}^*, \text{ NULL}^{|\Omega| \times |N|}\right]$ ▷ Expand $\boldsymbol{z}$ with placeholder for internal nodes
  $\bar{N} \leftarrow$ POSTORDERTRAVERSAL($G[N]$) ▷ "Bottom-up" sorting of internal nodes
  for $v \in \bar{N}$ **do**
    $D_v \leftarrow$ set of child nodes of v
    for $\omega \in \Omega$ **do** ▷ For each character in the set of states, Ω
      $\phi_\omega \leftarrow \sum \boldsymbol{z}^*_\omega(D_v)$ ▷ Sum of probabilities of character ω among children
      $z_{\omega,v} \leftarrow \frac{\phi_\omega}{|D_v|}$ ▷ Probability of character ω in v is average of children's
    end for
    if v is root **then**
      $cost_v \leftarrow |D_v| - \phi_{\omega=\omega_{root}}$
    else
      $cost_v \leftarrow |D_v| - \max_{\omega \in \Omega}(\phi_\omega)$ ▷ Sum of children's disbelief in most probable $\omega \in \Omega$
    end if
  end for
  return $\sum_{v \in N} cost_v$
end procedure

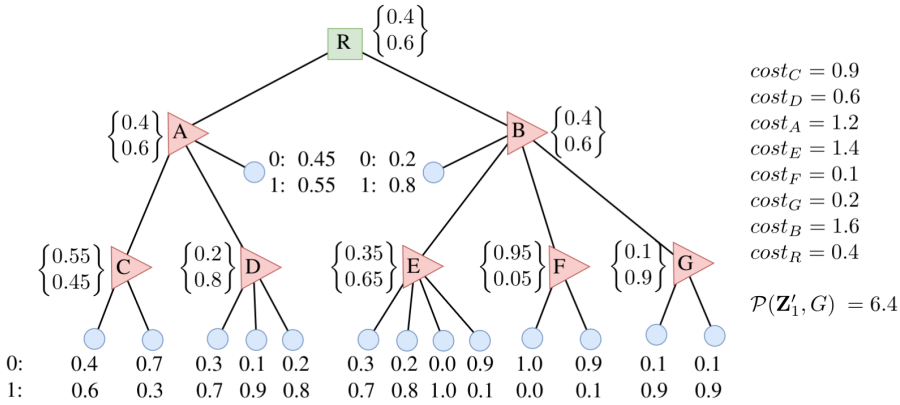


Figure 9.6.3: An example of how to compute the optimal parsimony score for a specific tree G using algorithm 9.3. Here, a single time point in the input data sequences $\boldsymbol{Z}'$ is considered. This data consists of probabilities of each of the two states in the alphabet for each customer (blue nodes). The red nodes are splitters/amplifiers, and the green node is the CMC. The computed probabilities of each state for the internal nodes are listed in curly brackets. The optimal parsimony score for the example tree is 6.4. The cost in each node corresponds to the sum of the probabilities for the least probable state as believed by the children. For instance, in node C, the most probable state is 0, hence the cost becomes the sum of probabilities of state 1 as observed in the children; $0.6 + 0.3 = 0.9$.

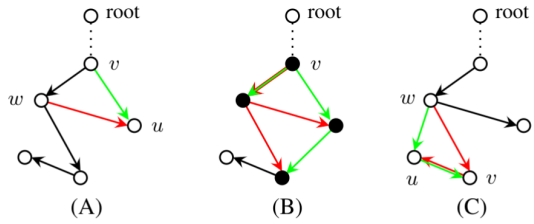


Figure 9.6.4: Examples of the three neighborhood functions used to generate negative samples, G_A^n, G_B^n, and G_C^n, of a topology G. (A) *Random swap*. (B) *Subtree shuffle*. (C) *Parent-child swap*. The green edges are new edges introduced by each neighborhood function, the red edges are edges removed by each neighborhood function. Figure taken from [138].

9.7 Experiments

In order to accurately assess the performance of our methodology, we have chosen to conduct the following set of experiments, designed to show:

- The legitimacy of our proposed modified version of the parsimony algorithm, i.e. Conjecture 1 (see section 9.7.3 and section 9.8.1 for experiment details and results, respectively).
- The performance of our approach for small trees, where it is feasible to consider all possible tree topologies during both training and validation. Including the impact of event sparsity/density in the data and the influence of multi-faults, as well as the performance under what is found to be the ideal data conditions (section 9.7.4 and section 9.8.2).
- The performance of our approach on bigger trees where only a subset of topologies can be considered for computational reasons, as well as the effect of different sampling strategies for generating the considered subset of topologies (section 9.7.5 and section 9.8.3).
- The performance of a single generalized encoder's ability to encode data for a wide range of tree sizes (section 9.7.6 and section 9.8.4).

Because this work intends to assess the feasibility of our contrastive approach, we will keep both the simulated data and the encoder model simple. For all experiments, we simulate independent training and testing sets each containing a set of instances, where an instance constitutes one true tree topology, a number of customer modems, and modem time series simulated based on the true tree.

We train our model on the training sets and assess the performance on the test sets. Inspired by Franceschi et. al in [64], but originally proposed by van den Oord et. al in [187], our encoder model will use a sequence of one-dimensional convolutional layers with exponentially increasing dilation to handle the time series. Since the topology reconstruction problem is not imminent, our model is not chosen to be causal - oppositely to the work of van den Oord et. al. and Franceschi et al. This means that our model includes data in time both prior and posterior to the prediction point under consideration. After a sequence-to-sequence encoding of the input time series, our model applies an identical fully connected layer to every time point to summarize the output of the convolutional channels into a single vector of output values for each time point. These outputs will act as an event's encoded probabilities after being sent through a sigmoid activation function. Recall that the parsimony algorithm treats every time point

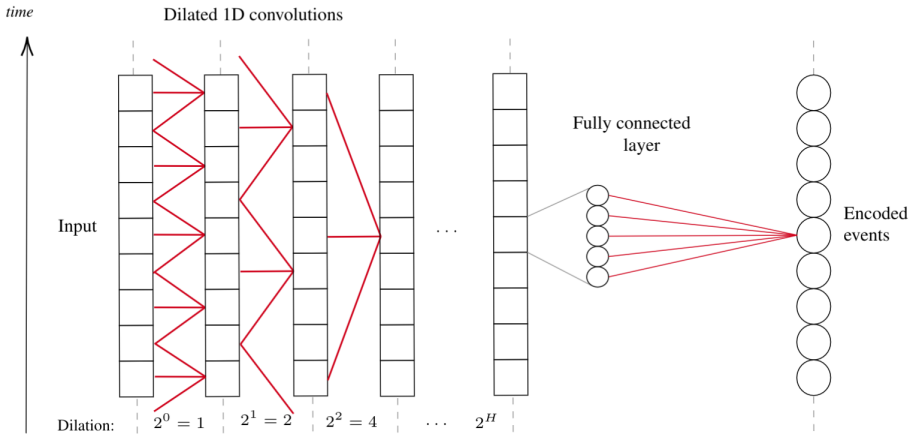


Figure 9.7.1: Illustration of the architecture used in the experiments. The use of dilated convolutions is inspired by Franceschi et. al in [64] and van den Oord et. al in [187], but differs in that a fully connected layer is applied to the output at every time point, retaining the time component in the output. H is the number of convolutional layers (hence also the coarsest dilation) in the architecture.

independently; hence multiple event vectors can be concatenated to form the input for the parsimony algorithm. An illustration of the architecture is shown in Figure 9.7.1. In the experiments conducted in this study, we opt for a value of 3 for H, a kernel size of 5 for each convolution, 16 channels for each convolutional layer, and eight neurons for the final fully connected layer.

It should also be mentioned that, in the case of binary events, there are two close-to-identical local minima in the encoder parameter space: the one that correctly identifies the presence or absence of events (encoded to be 1s and 0s in the output, respectively), and the one where the two are interchanged. This is not a problem in the traditional parsimony formulation, however, using our alternative formulation of the parsimony score given in (9.3), an interpretation of the output of the encoder is introduced. Since the root-level base signal is assumed to be all 0s, this slightly affects the value of the parsimony score. We get around this by simply checking after training (but still using the training data) which of the two cases provides the lowest modified parsimony score and thereby choose whether or not to flip all outputs of the encoder in the validation stage.

9.7.1 Data simulation

Since the goal of this paper is to assess the ability of the proposed methodology to find inherited events in time series data, we will do the assessment in a controlled fully simulated setting. Thus, we simulate both the considered tree topologies as well as the continuous time series data. For each observation, we first simulate a true network topology with a predetermined number of internal nodes, N. We sample a number of child modems uniformly in the range from $5 \cdot (|N| - 1)$ to $9 \cdot (|N| - 1)$, both values included, and divide them among the internal nodes, except for the root, making sure that none of these nodes has less than two modems connected to it.

For small trees with less than seven internal nodes, we sample the true topology uniformly from the set of all possible topologies given a tree of that size. For bigger trees, we sample the true topology using a random sampling scheme discussed in the following section.

Lastly, we select the set of negative topologies. For small trees, this is the set of all possible topologies, while for bigger trees, the negative set of topologies will be sampled using one of the schemes discussed in the following.

9.7.1.1 Topology sampling

We define two different sampling schemes, *Smart* and *Random*, for topologies of a given number of internal nodes, $|N|$. The *random sampling* scheme is used for sampling both true topologies and topologies in the negative topology set, whereas, *smart sampling* is only used for sampling topologies in the set of negative topologies.

Random sampling We begin with the set of internal nodes, N, but without any edges connections between them (each of the customer leaf nodes in M has a preexisting parent in N, as described in section 9.5). One node in N is initially marked as the *root*. We then alternate between two ways of adding an edge, (u, v), to the graph until all internal nodes, excluding the root, have a parent, and consequently also a path to the root. In both approaches, we first sample a node $v \in N \setminus root$, that does not yet have a parent when considering the edges in E. The parent node, u, of v is sampled in one of two different ways:

- Sample u from the set of nodes in N for which the addition of the edge (u, v) would not create a cycle in G.

- Sample u from the set of descendants of the root node or the root node itself, i.e. nodes in N that already have a path to the root node, when considering the edges in E.

Smart sampling A set of smart samples is generated based on a true topology, G^p, each through a series of *neighborhood moves* as described in section 9.6.3. For each move, one of the three neighborhood functions is selected with a 45%, 10%, and, 45% probability, respectively. Each smart sample consists of $k \in [1, 3N]$ neighborhood moves chosen with exponentially increasing distance between the number of moves, however, sampled with uniform probabilities. Topologies sampled more than once are discarded, resulting in a naturally skewed distribution of the number of moves[3].

9.7.1.2 Time series simulation

For the time series data, we simulate perfectly distinguishable events by splitting the sequence of time points into several chunks of a predetermined size, τ, each with a start time, t_i, and an end time, $t_{i+\tau}$. For each chunk, we first simulate the presence or absence of an event in the network by sampling from a Bernoulli distribution with a 50% event occurrence probability. In other words, we determine the set of affected time points, T^*. If an event occurs in a given time chunk, we then sample a random edge uniformly from the set of internal edges in the network and from this edge determine the set of modems, M_t^*, affected by the event. Inspired by De Ryck et. al [42], Chang et. al [29], Kawahara & Sugiyama [84], Liu et al.[104], and originally proposed by Yamanishi & Takeuchi [197], we use the AR(2) process [112] as a base signal for our simulated time series data for a modem, m:

$$z_{m,t} = 0.6z_{m,t-1} - 0.5z_{m,t-2} + \epsilon_{m,t}, \qquad \text{where} \qquad \epsilon_t \sim \mathcal{N}(\mu_{m,t}, 0.5^2) \quad (9.8)$$

We choose $\mu_{m,t} = 3$ if modem $m \in M_t^*$, i.e. is affected by a fault at time t. Otherwise, we set $\mu_{m,t} = 0$; i.e. if modem $m \notin M_t^*$, which means that it is *not* affected by a fault at time t. An example of the simulated data is shown in Figure 9.7.2.

[3]The probability of permuting to the same sample multiple times decreases as the number of moves increases.

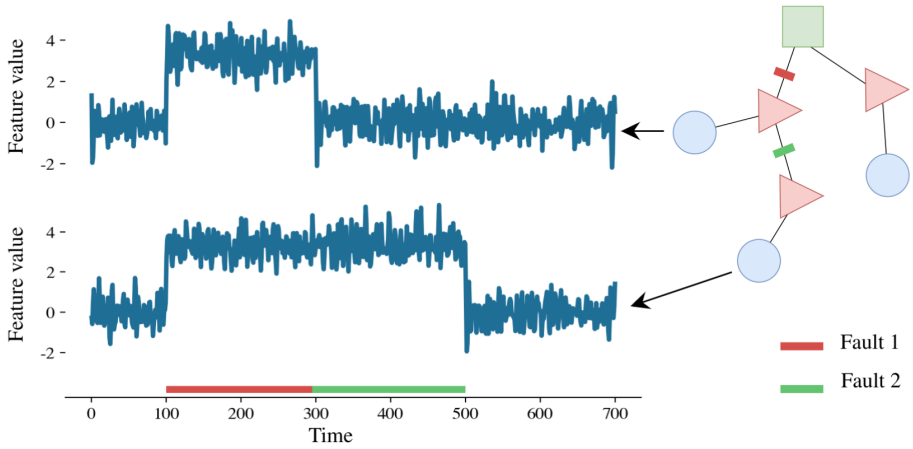


Figure 9.7.2: An example of part of a data instance with a tree of size $N = 4$ shown on the right (only three out of 29 modems are shown). A snapshot of the time series data is shown for two of the modems in which two faults occur on two different edges. These faults cause the time series for the modems to be affected in two different ways, as dictated by the tree topology. The base level signal has a mean of roughly zero.

9.7.2 Evaluation metrics

To properly assess and compare the performance of our methodology in different settings, we define two accuracy metrics to be used across all of our experiments. One metric relates to our ultimate goal of accurately inferring the true topology of a network, and the other to the end goal of training an encoder that properly learns to identify the true underlying simulated events.

9.7.2.1 Topology Accuracy

We define the *topology accuracy* of an encoder to be the ratio of observations for which the *true topology* (i.e. the topology used to generate the modem time series) leads to the lowest parsimony score out of *all topologies* in a test set $\mathcal{G} = \{G^p\} \cup \mathcal{G}^n$. This means that for a set of observations, $s = 1 \dots S$, where each observation consists of a discrete-valued matrix, $\boldsymbol{Z}_s$, and a set of topologies,

$\mathcal{G}_s$, whereof one is the true topology, G_s^p, the topology accuracy is given by:

$$\text{topology accuracy} = \frac{\sum_{s=1}^{S} \mathbb{1}\left(\boldsymbol{Z}_s, G_s^p, \mathcal{G}_s^n\right)}{S} \qquad \text{where}$$
$$\mathbb{1}(\boldsymbol{Z}, G^p, \mathcal{G}^n) = \begin{cases} 1 & \text{if} \quad \hat{\mathcal{P}}(\boldsymbol{Z}, G^p) < \hat{\mathcal{P}}(\boldsymbol{Z}, g), \quad \forall g \in \mathcal{G}^n \\ 0 & \text{otherwise} \end{cases} \tag{9.9}$$

If $\mathcal{G}$ contains all possible topologies for a given tree size we call it the *true topology accuracy*. This metric, however, quickly becomes cumbersome to calculate as the size of a tree grows, since the number of possible topologies increases exponentially with respect to the tree size as given by Cayley's formula [24]. For this reason, we also define the *estimated topology accuracy*, to be the ratio of observations for which the true topology leads to the lowest parsimony score considering $\mathcal{G}^n$ to be some (sampled) subset of all possible topologies for a given tree size.

9.7.2.2 Event Accuracy

We define the *event accuracy* of an encoder as the mean accuracy of the encoder over a set of observations. This refers to the mean ratio of time points for which the encoder correctly identifies the presence or absence of an event, relative to the true underlying data simulation. This means that for a set of observations, $s = 1, \ldots, S$, where each observation consists of a discrete-valued matrix, $\boldsymbol{Z}_s$, of encoded events and a matrix, $\boldsymbol{Z}_s^*$, of true underlying simulated events, the event accuracy is given by:

$$\text{event accuracy} = \frac{\sum_{s=1}^{S} \sum_{m=1}^{|M|} d_H\left(\boldsymbol{z}_{s_m}, \boldsymbol{z}_{s_m}^*\right)}{S \cdot |M|} \tag{9.10}$$

where M is the set of distinct modems and d_H is the Hamming distance given in (9.2) between two vectors, in this case, the row vectors of $\boldsymbol{Z}_s$ and $\boldsymbol{Z}_s^*$, both of length $|B|$. This metric can also be used on a single observation (i.e. a single network) to assess how well the encoder extracts the true underlying discrete events from the continuous time series data.

9.7.3 Uniqueness conjecture experiments

The validity of Conjecture 1 is challenged through a series of rigorous experiments designed to find a counterexample to the claim. The experiments constitute of two parts;

1. *Full* testing for trees with three to six internal nodes.
   It is computationally viable to compute and compare all possible topologies and thus prove that Conjecture 1 holds true up to and including trees of size six.

2. *Partial* sampled testing for selected trees with more than six internal nodes.
   A subset of all possible topologies for trees of size seven to ten as well as fifteen, will be tested against a sampled sub-set of topologies.

We use a tree size of $|N| = 3$ as an example to elaborate on part 1 of the experiment. There exist three unique topologies for a tree of size three. One of these topologies is picked to represent the true topology G^p, and based on G^p the modem data is simulated. As described in Condition 1, we simulate the occurrence of exactly one independent event per edge per time point, i.e. a total of $|E|$. Since G^p in the case of $|N| = 3$ contains two internal edges, the modem data strings also contain two data points. Considering a binary alphabet of discrete states, the root data string is chosen as a string of zero bits, without loss of generality. Then, we let each data point correspond to an event occurring on each of the two edges and flip the bit for the modems affected by each event.

Using the Modified Parsimony Score, as computed by Algorithm 9.2, the parsimony score is calculated for each of the possible topologies using the same simulated data. For Conjecture 1 to hold true, G^p must yield a unique best score out of the three possible topologies. Repeating this process three times, each time letting a new topology represent G^p, proves that Conjecture 1 holds true for all trees of size three.

Similarly, we use a tree size of $N = 10$ as an example to elaborate on part 2 of the experiment. A tree of size ten has 100 million possible topology configurations. Consequently, we choose to only let 1000 randomly chosen topologies represent the true topology G^p and simulate modem data for each of them in the same way as described in Part 1.

For each G^p we now *sample* a sub-set of 10,000 topologies, $\mathcal{G}^n$, for which we compute the modified parsimony scores and compare them to the score of G^p. For $\mathcal{G}^n$ to best represent the total set of 100 million topologies, we obtain half of the samples in $\mathcal{G}^n$ by *Smart Sampling* and the other half by *Random Sampling* (see section 9.7.1.1 Topology sampling). This ensures that $\mathcal{G}^n$ includes topologies that are both similar and dissimilar to the true topology G^p.

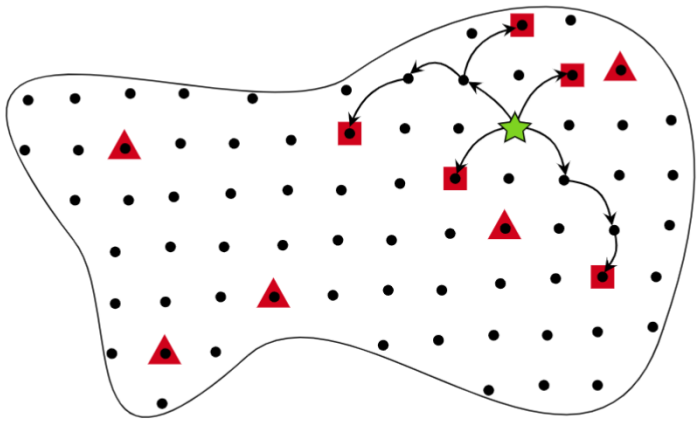


Figure 9.7.3: An illustration of *Random* and *Smart* sampling. The abstract shape illustrates the entire topology space for a network of size N. Each black dot inside the space represents a unique topology. The green star represents the true topology G^p and the selected set of samples $\mathcal{G}^n$ are marked with red, where triangles correspond to random samples and squares to smart samples. The arrows illustrate the neighborhood moves performed to reach each smart sample.

9.7.4 Training the encoder on the full set of topologies

We evaluate the performance of our proposed methodology when the model takes all possible topologies for a given tree size into account. As previously mentioned, we assume that the last-line amplifier/splitter is known, hence only the topology of the internal nodes will be inferred. In this setting, we test the performance of our approach for trees of varying sizes, however, keeping them small makes it feasible to train and test over multiple repetitions. Simultaneously, we assess the robustness of our methodology as data conditions become sub-optimal. We presume based on Conjecture 1, that the true topology, G^p, leads to the unique lowest parsimony score if the data contains at least one event on every edge in the network and that there is at most one event at a given time point.

For each data set, we simulate 1,000 observations of which half will be used as a test set. In all cases, an observation is given by, firstly; a true topology, G^p, sampled at random, uniformly from the set of possible topologies for a tree of a chosen size. Secondly; a set of negative samples, $\mathcal{G}^n$, is chosen to be the set of all possible topologies excluding G^p. Lastly; an array of time series data for the modems of the network, simulated specifically according to each specific experiment explained in the following.

9.7.4.1 Number of events in the time series data

We simulate ten different data sets for both trees of size four, five, and six, respectively, i.e. a total of 30 data sets. Each of the ten data sets will correspond to an *average* number of η events per internal edge in E. We fix the chunk size to $\tau = 30$ and simulate time series of length $|T| = 2\tau\eta \cdot |E| = 60\eta \cdot |E|$ for each average number of events, η, in the set $\{1, 2, \ldots, 10\}$. We do, however, not perform experiments using all ηs for bigger trees due to the infeasibility of computation when trees, thus also their time series, increase in size. We multiply by two to account for the fact that events only occur in a given chunk with a probability of 50% as described in section 9.7.1 Data simulation.

9.7.4.2 Introducing multi-faults

We simulate 11 different data sets for both trees of sizes four and five, i.e. a total of 22 data sets. Each of the 11 data sets corresponds to *a proportion of events* in the time series, ρ, for which two faults occur simultaneously on two different internal edges instead of one. We investigate values of ρ in the set

$\{0\%, 10\%, 20\%, \ldots, 100\%\}$. For each data set, we fix the chunk length to $\tau = 50$ and the average number of events to $\eta = 10$ to make sure that the encoder has good conditions for learning. These numbers are chosen generously to make the influence of multifaults transparent. This results in a time series length of $|T| = 2\tau\eta \cdot |N| = 1,000 \cdot |N|$.

When simulating an event (after checking if an event is present) in a given chunk as described in section 9.7.1 Data simulation, we sample from a Bernoulli distribution with probability ρ of a multi-fault. If a multi-fault should be present, we sample two different internal edges, instead of just one, uniformly from the set of internal edges and modify the time series of the modems beneath both faults accordingly.

9.7.4.3 Performance under ideal data conditions

We consider the case with ideal data conditions (i.e. only singular faults, long time series consisting of long-lasting faults) for our approach and simulate time series under these conditions for trees of size four and five. Already for trees of size six, it is too demanding to train the encoder considering ideal data conditions. This is due to the loss function (9.7), in this case, requiring $6^4 = 1,296$ (see section 9.3.2) parsimony evaluations for each forward pass in the training loop, because all possible topologies are considered in the set of negative samples. We fix the chunk length to $\tau = 50$ to be able to computationally evaluate gradients in the process of learning the encoder. We simulate time series with $\eta = 10$ events per internal edge in the tree on average, to allow for accurate modeling. This results in a time series of length $20\tau|E|$. For these results, we exclude the results where the encoder ends up in the pitfall of learning the zero-encoder because it is easy to check if that is the case. This is still a possibility even though we are using a contrastive approach to avoid it.

9.7.5 The influence of sampling

As the size of a network increases beyond six, it is no longer computationally viable to train the encoder on a set of a negative sample $\mathcal{G}^n$ which includes all possible topologies. Realistically, $\mathcal{G}^n$ must therefore be a sub-set representing the full *topology space*. We aim to compare the effectiveness of two different sampling techniques as well as investigate how many samples are needed to represent the full topology space. The effectiveness will be evaluated using the topology accuracy given by (9.9).

The two sampling techniques considered are *Random sampling* and *Smart sampling* which were introduced in section 9.7.1.1. For larger trees, it quickly becomes impractical to sample a fixed proportion of the full topology space, to represent the negative samples $\mathcal{G}^n$. Instead, we investigate the relationship between the number of negative samples and the size of the trees, denoted as $|N|$. Specifically, we explore the size of $\mathcal{G}^n$ within the range from 1 to $3|N|$, both values included, as this range is computationally feasible for trees of up to approximately size 400.

The experiment includes trees of size six, seven, nine, and fifteen, where size six has the unique benefit that both the *true* and *estimated* topology accuracy (9.9) can be computed and consequently compared. Additionally, for size six it is feasible to also *train* the encoder letting $\mathcal{G}^n$ include all possible topologies, making this a baseline for evaluating the effect of letting $\mathcal{G}^n$ be a sampled subset of all possible topologies. This is, however, computationally very heavy. For trees of size five and smaller, the size of the topology space is at most 125, and consequently, $3|N|$ constitutes a significant fraction of the full space, making such results non-generalizable. This is why we only consider trees of size six or bigger.

To avoid sampling bias, each experiment for each tree size will be conducted on 500 different true topologies G^p. We compare the two sampling methods using the same set of true simulated topologies and time series, hence only the set of negative samples are simulated differently for the two methods. Both sampling methods are tested on the same test using the estimated topology accuracy defined in (9.9). A test set constitutes an equal mix of both random and smart sampling as described in section 9.7.1.1. An illustration of the full experiment setup is shown in Figure 9.7.4. We set the length of the data chunks to $\tau = 30$ and the length of the time series to $|T| = 12\tau N$ for a mean number of events per edge of six.

9.7.6 A generalized encoder for multi-sized trees

Since HFC networks rarely have the same size, i.e. the same number of amplifiers/splitters, we check the ability of our approach to learn an encoder when the test samples stem from observations with trees of different sizes. We simulate 500 training samples each consisting of a randomly sampled true topology, G^p, with a number of internal nodes, $|N|$, in the range 4 to 20, both values included. The set of negative topologies, $\mathcal{G}^n$, will be based on $3|N|$ sampled topologies, sampled using the smart sampling scheme, as described in section 9.7.1.1. For the test set, we simulate 500 observations based on the same method, however, letting the set of negative topologies be based on 500 samples, sampled using a

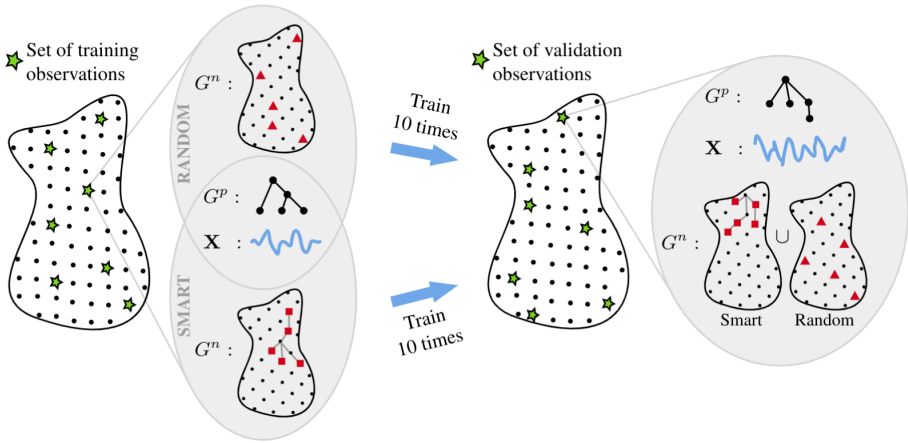


Figure 9.7.4: The data generation scheme for the sampling influence experiment. Both sampling strategies are trained on the same set of true topologies and time series, hence only the set of negative samples vary between the two datasets. Both strategies are validated based on the estimated topology accuracy using the same validation set. Each observation in this set consists of a mixture of both random and smart samples, as described in section 9.6.3.

mixture between smart and random sampling, to ensure that the full topology space is well represented.

In this case, we simulate fewer customer modems, simulating the number in the range from $2 \cdot (|N| - 1)$ to $5 \cdot (|N| - 1)$, ensuring that all internal nodes, except for the root, each have at least two adjacent modems. We fix the chunk length to $\tau = 30$ and simulate on average $\eta = 10$ events per internal edges, such that the modem time series have lengths $|T| = 2 \cdot 10 \cdot 30 \cdot |N| = 600|N|$.

9.8 Results

9.8.1 Uniqueness conjecture experiments

Table 9.8.1 shows the results of the tests designed to disprove Conjecture 1. For tree sizes from three and up to and including seven, a full test—where all possible topologies have been tested against all possible topologies—was conducted. In all cases, the true topology leads to a unique lowest modified parsimony score.

Table 9.8.1: Results of tests designed to find counterexamples to Conjecture 1. All tests failed to disprove the conjecture.
Here, $|N|$ is the tree size, $\Psi(|N|)$ the number of possible topologies of size $|N|$ (Cayley's formula), *# Tested* the number of unique topologies tested as being true (i.e. used to simulate data from) followed by the corresponding fraction of $\Psi(N)$ given in parentheses, *# Tested against* the number of unique topologies each true topology is tested against, *Successful* the number of tests which successfully disproves Conjecture 1, and *Proved* marks which test fully prove Conjecture 1 by exhaustive search.

$\|N\|$	$\Psi(\|N\|)$	# Tested (%)	# Tested against	Successful	Proved
3	3	3 (100.00)	*All*	0	✓
4	16	16 (100.00)	*All*	0	✓
5	125	125 (100.00)	*All*	0	✓
6	1,296	1,296 (100.00)	*All*	0	✓
7	16,807	2,004 (11.92)	*All*	0	
8	262,144	1,002 (0.38)	10,000	0	
9	4,782,969	882 ($1.84 \cdot 10^{-2}$)	10,000	0	
10	100,000,000	786 ($7.86 \cdot 10^{-4}$)	10,000	0	
15	1,946,195,068,359,375	528 ($2.71 \cdot 10^{-11}$)	10,000	0	

For reference, case $N = 7$ took around 12 hours to compute using six parallel threads on an *Intel Xeon E5-2620 v4 (2.1 GHz)* CPU.

For the remaining cases in Table 9.8.1, only a subset of all topologies was tested against a subset of topologies. However, all tests still show that the true topology leads to a uniquely best parsimony score.

Given the mix of a *random* and a *smart* sampling technique, the subset tests still provide compelling evidence for Conjecture 1 for larger trees. Random sampling ensures a wide range of differing topologies is considered while smart sampling ensures topologies similar to the true topology are also considered. Thus, the full *topology space* is well represented in the sampled sub-set tests.

9.8.2 Training the encoder on the full set of topologies

Using the full set of possible topologies as the negative samples in our contrastive approach gives the model the maximum amount of information possible and, hence serves as a baseline for the best possible performance.

Figure 9.8.1 shows the true topology and event accuracy, respectively, of our contrastive approach as a function of the mean number of events per internal edge simulated in the time series. Figure 9.8.1a shows a clear trend with the performance increasing as the information (number of events) in the simulated time series increases. It also shows that the topology accuracy converges at 100%, affirming the potential of our contrastive approach in learning an encoder that can extract inherited events from time series.

Interestingly, from Figure 9.8.1b it seems that the trend of the event accuracy as a function of the mean number of events is rather constant. Moreover, the increased topology accuracy cannot be directly correlated to a simultaneous increase in the event accuracy. In other words; it seems that the encoder does not need to extract all the events in the time series to learn an informative encoder with the ultimate goal of being able to do accurate topology reconstruction.

Additionally, Figure 9.8.1a also shows that, even though the approach works well most of the time, there is still a possibility of the encoder learning the trivial case; the zero-encoder, resulting in a topology accuracy of 0 (recall that the topology accuracy is based on the true topology being uniquely best). This is, however, not believed to be problematic, since it is easy to check if this is the case after training.

9.8.2.1 Introducing multi-faults

We test the robustness of our approach with respect to the challenge posed by *multi-faults* as introduced in section 9.5.4. Figure 9.8.2 shows the performance of our approach as a function of the proportion of simulated events that are multi-faults, i.e. indistinguishable events happening simultaneously at two locations in the topology, affecting the descendant modems of both locations. From Figure 9.8.2a it seems that our approach is somewhat robust to multi-faults up to a level of approximately 20-30%. The figure also suggests a similar robustness for trees of sizes four and five, respectively, due to the curves showing close-to-identical trends. In Figure 9.8.2b a seemingly negative trend is visible that ends at approximately 50% (corresponding to a random encoder) when all the generated faults are multi-faults. It should be mentioned that the number of simulated events in this case is rather high (ten events per internal node), to allow for higher granularity in the tested proportions.

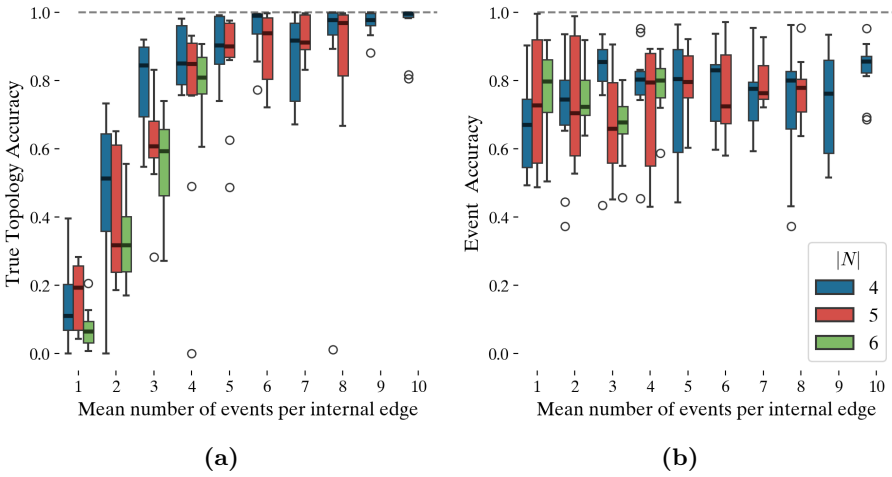


Figure 9.8.1: Performance of the proposed approach as a function of the number of events generated in the simulated data. Reported using a grouped boxplot of ten repeated experiments for each number of events per internal node and tree size combination. Each repetition is randomly initialized with a new encoder. Results are truncated at the point where we did not find it feasible to train due to memory computation constraints (length of time series, thus also computational demand increases with the number of events).

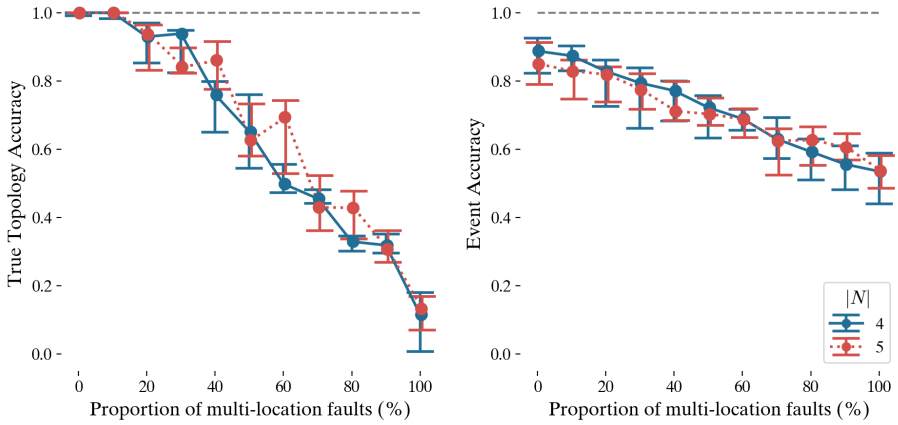


Figure 9.8.2: Performance of the proposed approach as a function of an increasing amount of multi-faults, i.e. faults that happen on multiple edges at the same time. Results are reported as the median and 50% confidence interval (CI) of ten experiments, each with a new and randomly initialized encoder.

Table 9.8.2: Results for trees of size four and five when all possible topologies are used as negative samples in the contrastive approach. The results are reported as the mean and standard deviation of 20 repetitions where a new random initialization of the encoder is performed. The length of the chunks, τ, is chosen to be 50, to allow for accurate modeling. Because it is easy to check, zero-encoders have been excluded from these results which means that experiments have been repeated until 20 encoders have been learned that did not end up in this pitfall.

$\|N\|$	Topology accuracy	Event accuracy
4	98.27 % (±3.01)	86.16 % (±9.82)
5	97.92 % (±3.36)	82.88 % (±10.13)

9.8.2.2 Performance under ideal data conditions

Table 9.8.2 shows the results for trees of sizes four and five in an ideal setting without multi-faults and with time series containing the information necessary to learn an optimal encoder, based the experiment in Figure 9.8.1. These results have been reported as the mean and standard deviation of 20 repetitions, however, removing samples in which the zero-encoder was learned.

We observe that the accuracy is very high in both cases. This suggests that the learned encoder successfully generated representations that, in almost all instances, resulted in the topology from which the data was sampled, having the lowest parsimony score among all possible topologies.

Similar conclusions as the ones made for Figure 9.8.1 can be drawn; i.e. the event accuracy stagnates at a sub-optimal level of approximately 85%, which is inconsequential to the topology accuracy which is near perfect at a level of approximately 98%.

9.8.3 The influence of sampling

The topology accuracy for trees of sizes 6, 7, 9, and 15 is depicted in Figure 9.8.3. Here, the negative set of topologies $\mathcal{G}^n$, utilized during training the encoder, is a sampled subset. Assuming the results from Table 9.8.2 can be generalized to larger trees, we know that it is theoretically possible to train an encoder that delivers a topology accuracy of around 98%. Since this result was obtained when

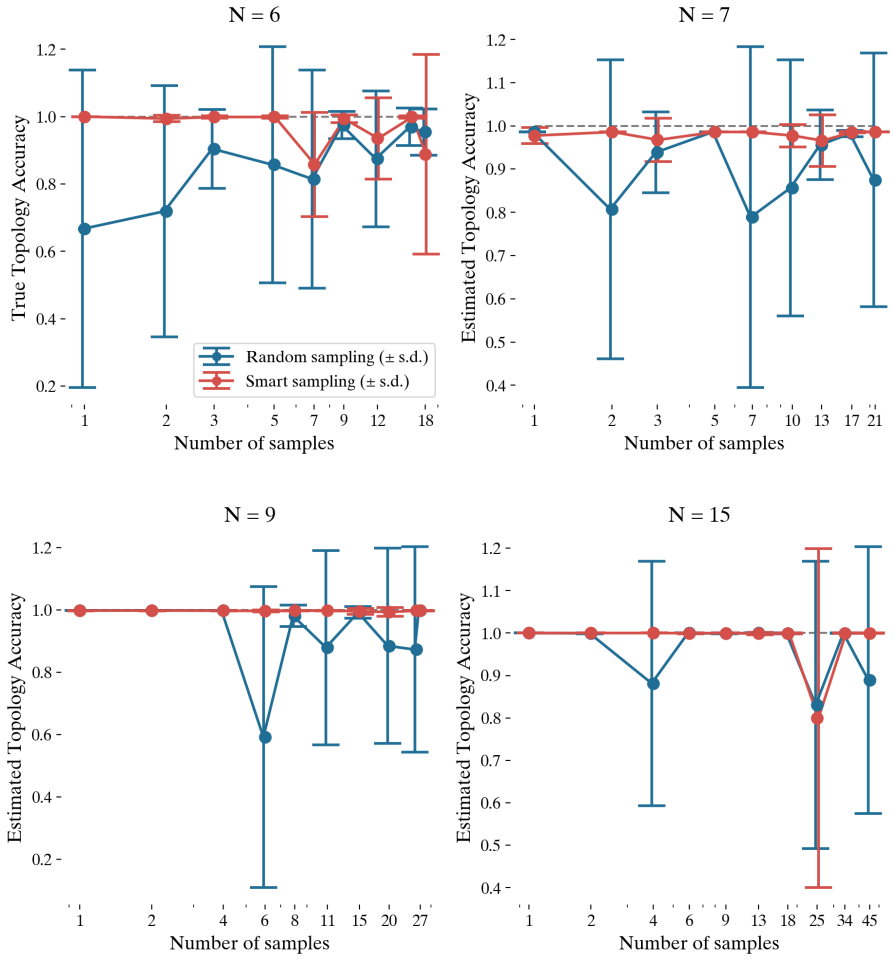


Figure 9.8.3: The above results illustrate how the *topology accuracy* of trees of size 6, 7, 9, and 15 is influenced by each of the two sampling strategies considered as well as the sample size. Each data point represents an average of ten repetitions of training and validating the encoder on 500 observations, respectively, where each observation is based on a unique combination of a true topology, G^p, and modem data. See Figure 9.7.4 for an illustration of the data generation scheme. The whiskers on each data point denote the standard deviation.

Continued on the next page

Figure 9.8.3 ***(continued)*****:** The x-axis (log scale) shows the number of samples in $\mathcal{G}^n$ in the range from 1 to $3N$, used to train the model in each observation.
The trained encoders are validated on a validation set containing 500 observations each consisting of both random and smart samples. Zero-encoders have been excluded from the results.
For tree size $N = 6$, the true topology accuracy is computed, whereas the remainder of the results are based on the estimated topology accuracy. In all experiments, employing the smart sampling scheme for sampling the negative set of topologies used in training outperforms the random sampling scheme.

training and testing on the full set of possible topologies, it serves as a baseline when evaluating the influence of sampling.

The figure furthermore compares the topology accuracy when sampling $\mathcal{G}^n$ during the training stage using either a fully *Smart* or fully *Random* sampling scheme. Moreover, Figure 9.8.3 shows how the number of samples used when training the encoder, i.e. the size of $\mathcal{G}^n$, affects the topology accuracy.

It is immediately clear that Smart sampling outperforms Random sampling in all four experiments in Figure 9.8.3. Moreover, for $N = 6$ there is a positive correlation between the true topology accuracy and the number of samples in $\mathcal{G}^n$. This trend is not apparent for the remaining values of N considering the estimated topology accuracy. However, it is important to note that these results contain an increasing level of uncertainty. For $N = 6$, the full topology space is small enough for us to compute the *true* topology accuracy, i.e. where the test set contains all possible topologies of size six. For the remaining experiments, it is only possible to compute the *estimated* topology accuracy, i.e. only comparing the true topology to a subset of possible topologies during testing. Since the size of the test set compared to the full set of topologies dramatically decreases for increasing tree sizes, consequently, so does the validity of the estimated topology accuracy.

Despite this evaluation uncertainty, we have attempted to sample a fair test set of topologies using both Smart and Random samples. In Figure 9.8.3, however, the topology accuracy appears to improve for increasing tree sizes. This suggests that the estimated topology accuracy is optimistic for bigger trees, as the full variation of the exponentially increasing topology space might not be captured by the fixed-sized sampled test set. However, an alternative explanation is that the topology accuracy improves for bigger trees due to an increase in the event accuracy, which we found to improve by approximately 10% from the $N = 6$

Table 9.8.3: Results for generalized encoder. Results, especially event accuracy are better than for when an encoder is only trained on one tree size.

Topology accuracy	Event accuracy
99.89 % (±0.23)	93.44 % (± 5.87)

case to the $N = 15$ case.

Overall, the smart sampling scheme during training results in consistently high topology accuracies with respect to both the size of the tree and the number of training samples, $\mathcal{G}^n$. Moreover, this is achievable with considerably fewer computational resources.

9.8.4 A generalized encoder for multi-sized trees

Lastly, Table 9.8.3 shows the results of training and testing a generalized encoder on sets of trees of varying sizes up to 20. Although one would expect this learning task to be more challenging, the results in the table suggest otherwise. Here, zero-encoders have been excluded from the results enabling a fair comparison with results in Table 9.8.2. However, it must be noted that in Table 9.8.2 the true topology accuracy is reported, whereas in Table 9.8.3 the topology accuracy is partially estimated.

As compared to the results of training and testing an encoder under the best possible conditions (see Table 9.8.2), the event accuracy improves from approximately 85% to more than 90% in the case of the generalized encoder. As the event accuracy is never an estimation and thus a reliable measure, this suggests that when the encoder is trained on only a single tree size, it is prone to overfitting.

Despite the increased event accuracy, Table 9.8.3 shows that the generalized encoder obtains only a slightly improved topology accuracy of approximately 99% as compared to the 98% observed in Table 9.8.2. Here, it is important to mention that this is only an *estimated* topology accuracy, since for larger trees, the test set only includes a sampled subset of all possible topologies.

9.9 Discussion

Overall, we find the results of this preliminary study of our proposed methodology promising. We demonstrated that a topology prediction accuracy of around 99% and an event accuracy of up to 93% are attainable for simplified time series data.

Additionally, we designed a set of experiments that: studied the robustness of the methodology with respect to various data assumptions; investigated performance under ideal data conditions; considered the impact of sampling; and investigated how generalizing the methodology impacted performance. We found the performance of the encoder to be best when trained on a set of trees of various sizes as opposed to one single size (see Table 9.8.2 and Table 9.8.3). When investigating the influence of multi-faults, we found that as long as multi-faults do not constitute more than 20% of all data events, the topology accuracy does not worsen by more than 10% (see Figure 9.8.2). Furthermore, we found that the minimum number of events needed to achieve the best possible topology accuracy depended on the size of the tree. As events were simulated to occur on random edges, this result intuitively makes sense, as it for larger trees become progressively less likely to sample each unique edge at least once due to the increasing total number of edges.

As trees grow in size, the proposed methodology is only able to consider a set of training and test topologies constituting a diminishing fraction of the complete topology space. We analyzed two sampling schemes and found the *smart* sampling scheme based on neighborhood moves to be best performing during training (see Figure 9.8.3). In testing, we let the topology test set be a mix of both *smart* and *random* samples. We computed the *true* topology accuracy for trees of size 6, where the full topology space contains 1,296 topologies, as a function of using various sizes of sampled training sets. The *estimated* topology accuracy of larger trees was computed as a function of the same training scheme, allowing for a credible comparison between the *true* and *estimated* topology accuracy as well as the two sampling schemes (see Figure 9.7.4).

We experimentally proved Conjecture 1 to be true for tree sizes up to and including six and used the same training and testing sampling scheme as above to challenge the validity of the conjecture for larger trees (see Table 9.8.1). Moreover, the 99% topology accuracy obtained in other experiments, meaning a unique best topology was almost always found, adds to the plausibility of Conjecture 1 although it remains to be formally proved.

For this preliminary study, data events were simulated to be simple and distin-

guishable and to occur on a single edge at a time (except for when investigating the influence of multi-faults). While real modem data is not this ideal, the encoder should still be able to identify the positions within the data where the behavior is suitable with respect to the contrastive loss function. Useful data behavior does not have to be actual events that modify the signal in a simple way as visualized in Figure 9.2.2, but can also be an underlying correlation structure, e.g. the same general behavior but scaled differently, or more complex events such as local instantaneous spikes or cyclic behavior. As modems collect various data metrics at every time point, some metrics may not be informative on their own but could be valuable collectively. Our proposed encoder can be readily adapted to handle multiple time series inputs on different scales, producing multiple latent variables. This collection of latent variables could form an informative ensemble.

Due to the proposed loss function including a minimization problem building on a bottom-up (poster-order) traversal of the internal nodes of the network, the evaluation of this is computationally expensive and not readily improvable by e.g. the use of GPUs. This is, however, not a problem for the ISPs (Internet Service Providers) as they would only need to train a generalized model once to infer all the missing topology. After this point, the loss function does not need to be evaluated again.

For the proposed methodology to give good results for ISPs, one of the main challenges would be to obtain a descriptive and extensive dataset that could be used for benchmarking. In training the model, the true topology of a given last-mile network would need to be known and as mentioned earlier, many ISPs often only partially know their own topologies. In the case of TDC NET; while the whole topology is only known for a small set of last-mile networks, it is far from all networks in which nothing is known. This means that ISPs could use the known part (and corresponding modem time series) of a network to train the encoder model, and then use the trained model to infer the missing part of the topology.

One of our main assumptions in our work is that every internal node needs to have at least one modem child. While this assumption might not hold for real networks, ISPs could train an encoder using only the parts of a set of real networks in which this assumption holds. Additionally, deployment of the full approach would mean using the encoded events to search for the optimal topology including geographic distances using e.g. the algorithm proposed by Pisinger & Sørensen in [138]. This means that the assumptions does not necessarily need to hold true for good deployment results in real life.

Our proposed loss function aims to simultaneously minimize the parsimony score of the true topology while maximizing the dissimilarities between distinct topolo-

gies. However, it may be oversimplified to solely focus on maximizing the distances between any two pairs, as similar topologies could yield small maximum distances. Therefore, incorporating a more sophisticated loss function that also considers the degree of similarity among all pairwise sets of topologies could prove beneficial.

Although we obtain promising results, there is still the possibility of the encoder ending up in the pitfall of learning a zero-encoder, which, in some cases, is sensitive to weight initialization. Even though it is easy to check if the encoder ends up in this pitfall, it could also be addressed by doing unsupervised pretraining of the weights of the network, making this less likely.

9.10 Conclusion

We propose a novel contrastive approach to performing discrete encoding of continuous time series while confirming the feasibility of integrating an optimization algorithm into a deep learning loss function. We present an encoder, f, tailored for HFC networks, which takes a set of continuous time series as input and generates binary sequences, implicitly learning data events in order to solve the topology reconstruction problem. Our approach leverages a contrastive loss function, incorporating a modified algorithm for computing the *parsimony score*. The modified parsimony score is designed to enable unique optimal solutions and we introduce a conjecture that states this is possible for all trees. We prove the conjecture for trees with up to six internal nodes and provide compelling evidence for bigger trees. To address the non-differentiability of binary sequences during training in the deep learning context, we also propose a continuous variant of the new modified parsimony algorithm.

Our results are promising, achieving an estimated accuracy of 99.89 % in topology reconstruction and an accuracy of 93.44 % in data event identification for a generalized encoder considering trees of multiple sizes. Additionally, we propose a sampling scheme to ensure comprehensive coverage of the full topology space, providing a solid foundation for training an encoder and generating accuracy estimates for larger trees.

As this work is only a preliminary pilot study, the natural next step will be to evaluate the methodology in a more realistic setting, potentially using real modem data. Nevertheless, our results still confirm the potential for infrastructure owners to infer missing topology using continuous time series from the customer modems on the network.

Although our proposed methodology was designed around a telecommunication framework, its applicability may extend to other fields. For instance that of phylogenetic inference in biology where despite genetic material being inherently discrete, the uncertainty and ambiguity in the data may justify adopting a continuous perspective. E.g. by considering the probability of each of the four nucleotides found in DNA occurring at each position in a genetic sequence.

Overall, our work highlights the advantage of integrating an optimization problem in a deep learning setting rather than using each of the methods individually. Despite the increased complexity introduced by this approach, its relevance remains significant, particularly for models requiring only a single training phase.

Acknowledgement

This project was supported by Innovation Fund Denmark, project 0224-00055B, GREENFORCE. David Pisinger was funded by ERC-2022-ADG, project 101093188, DECIDE.

Appendices

9.A Proof of Algorithm 9.2

Firstly, Algorithm 9.2 only modifies how the parsimony score is calculated in the top layer of a tree, namely when calculating the cost of the root node with respect to its immediate children. By letting the children of the root node be represented by the set D_{root}, then every subtree rooted in a node $u \in D_{root}$, must be optimal. This follows directly from the proof of the original parsimony algorithm since algorithm 9.2 calculates the cost of these subtrees in exactly the same manner.

Now, it remains to be proven that it is optimal to pay the cost of the root-node assumption in the top layer of the tree, and that it cannot be cheaper to move the top-layer data mutations further down the tree.

This part of the proof can be split up into two cases. For every node $u \in D_{root}$ we have that either:

- Case 1: the optimal state set of node $u \in D_{root}$ includes the state of the root node.
- Case 2: the optimal state set of node $u \in D_{root}$ does *not* include the state of the root node.

It is straightforward to prove that Case 1 is optimal. Since the state of the root is included in the optimal state set of node v, then there is no data mutation on edge ($root$, v), and the cost associated with this edge is then zero. Since mutation costs must be positive, a cost of zero in the top layer cannot be improved. Moreover, we have already argued that the subtree rooted in node v is optimal, and consequently also not improvable. In conclusion, in Case 1, algorithm 9.2 will compute the best possible parsimony score.

In Case 2, the state of the root node is *not* included in the state set of node v. As a result, algorithm 9.2 pays a cost of one for letting a data mutation occur on edge ($root$, v). The total cost of this part of the tree, according to algorithm 9.2, is then; $1 + \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$. The question is then, if this can be done cheaper by letting the mutation occur somewhere else in the subtree $G(v)$. i.e. obtain a cost $< 1 + \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$. This could potentially be achieved in two ways:

i) cost is $\leq 0 + \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$, i.e. the cost of edge ($root$, v) disappears while the cost of the subtree rooted in v stays the same or becomes less.

ii) cost is $\leq 1 + \Big(\mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v)) - 1\Big)$, i.e. the cost of edge ($root$, v) remains one, while the cost of the subtree rooted in v is decreased by at least one.

In i), if the cost of edge $(root, v)$ must disappear, then it means that the state of the root node must also be the state of node v. But given that we are in Case 2, then the root node state is *not* in the *optimal* state set of v. Consequently, when the state of v is forced to equal the root node state, then the total cost of the subtree rooted in v must be suboptimal, i.e. $> \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$, since at least one additional mutation must occur in the subtree rooted in v.

In ii), the cost of edge $(root, v)$ remains one—the state of v can be chosen freely—but consequently the subtree rooted in v must be less than $\mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$. Since $\mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$ is already optimal, this is impossible. This concludes the proof.

■

Bibliography

[1] C. C. Aggarwal. *Outlier Analysis*, pages 237–263. Springer International Publishing, Cham, 2015.

[2] M. Ahmed, A. Naser Mahmood, and J. Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[3] H. Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17(4):589–602, oct 1970.

[4] H. N. Akouemo and R. J. Povinelli. Probabilistic anomaly detection in natural gas time series data. *International Journal of Forecasting*, 32(3):948–956, 2016.

[5] C. Apte, H. Park, K. Wang, and M. J. Zaki. *Proceedings of the 2009 SIAM International Conference on Data Mining (SDM)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009.

[6] O. Ayoub, N. Di Cicco, F. Ezzeddine, F. Bruschetta, R. Rubino, M. Nardecchia, M. Milano, F. Musumeci, C. Passera, and M. Tornatore. Explainable artificial intelligence in communication networks: A use case for failure identification in microwave networks. *Computer Networks*, 219:109466, 2022. `https://www.sciencedirect.com/science/article/pii/S138912862200500X`.

[7] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv*, 2018. `https://arxiv.org/abs/1803.01271`.

[8] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271, 2018.

[9] S. Bansod and A. Nandedkar. Crowd anomaly detection and localization using histogram of magnitude and momentum. *The Visual Computer*, 36:609–620, 03 2020.

[10] L. Basora, X. Olive, and T. Dubot. Recent advances in anomaly detection methods applied to aviation. *Aerospace*, 6:117, 09 2019.

[11] Z. Bischof, F. Bustamante, and N. Feamster. Characterizing and improving the reliability of broadband internet access. *SSRN Electronic Journal*, 09 2017.

[12] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347, 2022.

[13] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[14] Broadcom Corporation. Digital transmission: Carrier-to-noise ratio, signalto-noise ratio, and modulation error ratio. White paper, Broadcom Corporation and Cisco Systems, 2012. `https://www.digikey.in/htmldatasheets/production/3367981/0/0/1/digital-transmission-white-paper.html`.

[15] T. Bujlow, T. Riaz, and J. M. Pedersen. A method for assessing quality of service in broadband networks. In *2012 14th International Conference on Advanced Communication Technology (ICACT)*, pages 826–831, 2012.

[16] P. Buneman. The recovery of trees from measures of dissimilarity. In F. Hodson, D. Kendall, and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, 1971.

[17] Børsen. Kunstig intelligens undgår omveje for tdc nets bilflåde. `https://borsen.dk/nyheder/baeredygtig/cases/2022/kunstig-intelligens-undgar-omveje-for-tdc-nets-bilflade`, 2022. (Danish - accessed on 15 May 2023).

[18] CableLabs. Cable broadband technology gigabit evolution. `https://www.cablelabs.com/insights/cable-broadband-technology-gigabit-evolution`, 2016. (accessed on 12 December 2023).

[19] CablesLabs. DOCSIS 3.1 Techonology. `https://www.cablelabs.com/technologies/docsis-3-1#What_is_DOCSIS_31_technology`, 2019. Accessed 25 June 2024.

[20] Y. Cai, M.-L. Shyu, Y.-X. Tu, Y.-T. Teng, and X. hu. Anomaly detection of earthquake precursor data using long short-term memory networks. *Applied Geophysics*, 16:257–266, 11 2019.

[21] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen. Unsupervised time series outlier detection with diversity-driven convolutional ensembles - extended version. *CoRR*, abs/2111.11108, 2021.

[22] A. Carreño, I. Inza, and J. Lozano. Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework. *Artificial Intelligence Review*, 53:3575–3594, 06 2020.

[23] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra. Deep learning for time series forecasting: Advances and open problems. *Information*, 14(11), 2023.

[24] A. Cayley. A theorem on trees. *Quart. J. Math.*, 23:376–378, 1878.

[25] J. Chagas, R. Ivo, M. Guimarães, D. Rodrigues, E. Rebouças, and P. P. Filho. Fast fully automatic skin lesions segmentation probabilistic with parzen window. *Computerized Medical Imaging and Graphics*, 85:101774, 08 2020.

[26] R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey. *ArXiv*, abs/1901.03407, 2019. `https://api.semanticscholar.org/CorpusID:57825713`.

[27] C. Challu, P. Jiang, Y. N. Wu, and L. Callot. Deep generative model with hierarchical latent factors for time series anomaly detection. In *International Conference on Artificial Intelligence and Statistics*, 2022.

[28] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58, jul 2009.

[29] W.-C. Chang, C.-L. Li, Y. Yang, and B. Póczos. Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*, 2019.

[30] B. Chen, Z. Zhang, N. Langrené, and S. Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv*, 2023. `https://arxiv.org/abs/2310.14735`.

[31] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. `http://doi.acm.org/10.1145/2939672.2939785`.

[32] W. Chen and K. Shi. Multi-scale attention convolutional neural network for time series classification. *Neural Networks*, 136:126–140, 2021.

[33] Y.-C. Chen, G. M. Lee, N. Duffield, L. Qiu, and J. Wang. Event detection using customer care calls. In *2013 Proceedings IEEE INFOCOM*, pages 1690–1698, 2013. https://ieeexplore.ieee.org/document/6566966.

[34] H. Choi, E. Jang, and A. A. Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv*, 2019. https://arxiv.org/abs/1810.01392.

[35] F. Chung, M. Garrett, R. Graham, and D. Shallcross. Distance realization problems with applications to internet tomography. *Journal of Computer and System Sciences*, 63(3):432–448, 2001.

[36] M. K. Chung. Introduction to logistic regression. *arXiv*, 3 2020. https://arxiv.org/abs/2008.13567.

[37] E. Commision. Annex to the commission recommendation on critical technology areas for the eu's economic security for further risk assessment with member states. https://defence-industry-space.ec.europa.eu/document/download/d2649f7e-44c4-49a9-a59d-bffd298f8fa7_en?filename=C_2023_6689_1_EN_annexe_acte_autonome_part1_v9.pdf, 2023. (accessed 23 April 2024).

[38] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[39] E. Dai and J. Chen. Graph-augmented normalizing flows for anomaly detection of multiple time series. *arXiv*, 2022. https://arxiv.org/abs/2202.07857.

[40] A. K. David M. Blei and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[41] W. H. Day, D. S. Johnson, and D. Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 81(1):33–42, 1986.

[42] T. De Ryck, M. De Vos, and A. Bertrand. Change point detection in time series data using autoencoders with a time-invariant representation. *IEEE Transactions on Signal Processing*, 69:3513–3524, 2021.

[43] H. M. DEITEL and B. DEITEL. Chapter 7 - data communications. In H. M. Deitel and B. Deitel, editors, *An Introduction to Information Processing*, pages 154–181. Academic Press, 1986.

[44] N. Devroye, N. Mohammadi, A. Mulgund, H. Naik, R. Shekhar, G. Turán, Y. Wei, and M. Žefran. Interpreting deep-learned error-correcting codes. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2457–2462, 2022.

[45] A. Dhingra, S. Kumar, Payal, D. Sharma, and S. Dahiya. A comprehensive review of qam-ofdm optical networks. *International Journal of Computer Sciences and Engineering*, 2018. `https://www.ijcseonline.org/full_paper_view.php?paper_id=3248`.

[46] K. Dhou. An innovative design of a hybrid chain coding algorithm for bi-level image compression using an agent-based modeling approach. *Applied Soft Computing*, 79:94–110, 2019.

[47] L. Dinh, D. Krueger, and Y. Bengio. NICE: non-linear independent components estimation. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[48] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[49] A. Dobson and A. Barnett. *An Introduction to Generalized Linear Models*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2018.

[50] Y. Du, W. Chen, K. Cui, Z. Guo, G. Wu, and X. Ren. An exploration of the military defense system of the ming great wall in qinghai province from the perspective of castle-based military settlements. *Archaeological and Anthropological Sciences*, 13(3):46, Feb 2021.

[51] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv*, 2018. `https://arxiv.org/abs/1603.07285`.

[52] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. Kwoh, X. Li, and C. Guan. Self-supervised contrastive representation learning for semi-supervised time-series classification. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(12):15604–15618, dec 2023.

[53] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359, 2021.

[54] European Commision. Communication from the European Commision - The European Green Deal. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2019:640:FIN`, 2019. Accessed 25 June 2024.

[55] European Commision. Broadband: Technology overview. https://digital-strategy.ec.europa.eu/en/policies/broadband-technology-overview, 2022. (accessed on 16 April 2024).

[56] O. S. Faragallah, G. Abdel-Aziz, and H. M. Kelash. Efficient cardiac segmentation using random walk with pre-computation and intensity prior model. *Applied Soft Computing*, 61:427–446, 2017.

[57] Federal Communications Commission. Getting Broadband Q&A. https://www.fcc.gov/consumers/guides/getting-broadband-qa, 2024. (accessed on 15 April, 2024).

[58] J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, 2003.

[59] P. Filonov, A. Lavrentyev, and A. Vorontsov. Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. *ArXiv*, abs/1612.06676, 2016.

[60] M. Fischer. On the uniqueness of the maximum parsimony tree for data with up to two substitutions: An extension of the classic buneman theorem in phylogenetics. *Molecular Phylogenetics and Evolution*, 137:127–137, 2019.

[61] W. M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20(4):406–416, 1971.

[62] P. Forsare Källman. Unsupervised anomaly detection and root cause analysis in hfc networks : A clustering approach. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2021. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1612886.

[63] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch. Somvae: Interpretable discrete representation learning on time series. In *7th International Conference on Learning Representations (ICLR)*. ICLR, May 2019.

[64] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi. Unsupervised scalable representation learning for multivariate time series. *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 1 2019.

[65] E. Gibellini and C. E. Righetti. Unsupervised learning for detection of leakage from the hfc network. In *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, pages 1–8, 2018.

[66] T. R. Gregory. Understanding evolutionary trees. *Evol Educ Outreach*, 1:121–137, 04 2008.

[67] S. Grover, M. S. Park, S. Sundaresan, S. Burnett, H. Kim, B. Ravi, and N. Feamster. Peeking behind the nat: an empirical study of home networks. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, page 377–390, New York, NY, USA, 2013. Association for Computing Machinery.

[68] S. Guan, Z. He, S. Ma, and M. Gao. Conditional normalizing flow for multivariate time series anomaly detection. *ISA Transactions*, 143:231–243, 2023. https://www.sciencedirect.com/science/article/pii/S0019057823004020.

[69] J. A. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29(1):53–65, 1973.

[70] V. Hautamaki, I. Karkkainen, and P. Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 430–433. IEEE, 2004.

[71] G. Heiler, T. Gadermaier, T. Haider, A. Hanbury, and P. Filzmoser. Identifying the root cause of cable network problems with machine learning. *arXiv*, 3 2022. `https://arxiv.org/abs/2203.06989`.

[72] J. Henríquez and W. Kristjanpoller. A combined independent component analysis–neural network model for forecasting exchange rate variation. *Applied Soft Computing*, 83, 2019.

[73] Y. Heryadi and Dandalina. The effect of several kernel functions to financial transaction anomaly detection performance using one-class svm. In *2019 International Congress on Applied Information Technology (AIT)*, pages 1–7, 2019.

[74] E. Hoogeboom, R. van den Berg, and M. Welling. Emerging convolutions for generative normalizing flows. *arXiv*, 2019. `https://arxiv.org/abs/1901.11137`.

[75] R. Hranac. Modulation error ratio. `https://broadbandlibrary.com/modulation-error-ratio/`, 2023. (accessed on 15 April 2024).

[76] J. Hu, Z. Zhou, and X. Yang. Characterizing physical-layer transmission errors in cable broadband networks. *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation*, 2022.

[77] J. Hu, Z. Zhou, X. Yang, J. Malone, and J. W. Williams. Cablemon: Improving the reliability of cable broadband networks via proactive network maintenance. In *Proceddings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020. `https://www.usenix.org/conference/nsdi20/presentation/hu-jiyao`.

[78] M. Hu, Z. Ji, K. Yan, Y. Guo, X. Feng, J. Gong, X. Zhao, and L. Dong. Detecting anomalies in time series data via a meta-feature based approach. *IEEE Access*, 6:27760–27776, 2018.

[79] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395, 2018.

[80] A. Hyvärinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3772–3780, Red Hook, NY, USA, 2016. Curran Associates Inc.

[81] International Telecommunication Union and Michaelis, A.R. *From Semaphore to Satellite*. International Telecommunication Union, 1965.

[82] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, Jul 2019.

[83] J. Kacprzyk, J. W. Owsiński, D. A. Viattchenin, and S. Shyrai. *A New Heuristic Algorithm of Possibilistic Clustering Based on Intuitionistic Fuzzy Relations*, pages 199–214. Springer International Publishing, Cham, 2016.

[84] Y. Kawahara and M. Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(2):114–127, 2012.

[85] C. D. Keeling. Climate change and carbon dioxide: An introduction. *Proceedings of the National Academy of Sciences*, 94(16):8273–8274, 1997.

[86] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 206–215, New York, NY, USA, 2004. Association for Computing Machinery.

[87] D. Khurana, A. Koli, K. Khatter, and S. Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, 82, 07 2022.

[88] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[89] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[90] P. Kirichenko, P. Izmailov, and A. G. Wilson. Why normalizing flows fail to detect out-of-distribution data. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[91] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015.

[92] M.-A. Kourtis, G. Xilouris, G. Gardikis, and I. Koutras. Statistical-based anomaly detection for nfv services. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 161–166, 2016.

[93] M. Kramer. Autoassociative neural networks. *Computers & Chemical Engineering*, 16(4):313–328, 1992. Neutral network applications in chemical engineering.

[94] F. Lartey. Proactive network and technical facilities monitoring using standardized scorecards. In *Proceedings of the Fall Technical Forum*, Denver, CO, 2017. SCTE, ISBE, NCTA. `https://www.nctatechnicalpapers.com/Paper/2017/2017-proactive-network-and-technical-facilities-monitoring-using-standardized-scorecards`.

[95] P. H. Le-Khac, G. Healy, and A. F. Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[96] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[97] S. Lee, K. Levanti, and H. S. Kim. Network monitoring: Present and future. *Computer Networks*, 65:84–98, 2014.

[98] W.-H. Lee, J. Ortiz, B. Ko, and R. Lee. Time series segmentation through automatic feature learning. *arXiv*, 11 2018. `https://arxiv.org/abs/1801.05394`.

[99] W. Lehr, M. Heikkinen, D. Clark, and S. Bauer. Assessing broadband reliability: Measurement and policy challenges. In *Research Conference on Communication, Information and Internet Policy (TRPC)*, 09 2011.

[100] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon. Similarity preserving representation learning for time series clustering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 2845–2851. AAAI Press, 2019.

[101] J. Li, H. Izakian, W. Pedrycz, and I. Jamal. Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*, 100:106919, 2021.

[102] J. Li, W. Pedrycz, and I. Jamal. Multivariate time series anomaly detection: A framework of hidden markov models. *Applied Soft Computing*, 60:229–240, 2017.

[103] J. Li, W. Pedrycz, and I. Jamal. Multivariate time series anomaly detection: A framework of hidden markov models. *Applied Soft Computing*, 60:229–240, 2017.

[104] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[105] G. Louppe. *Understanding Random Forests: From Theory to Practice*. PhD thesis, Department of Electrical Engineering and Computer Scienc at the University of Liège, 2014. `https://arxiv.org/abs/1407.7502`.

[106] H. Lu, J. Shi, Z. Fei, Q. Zhou, and K. Mao. Measures in the time and frequency domains for fitness landscape analysis of dynamic optimization problems. *Applied Soft Computing*, 51:192–208, 2017.

[107] Y. Lu, B. Cao, C. Rego, and F. Glover. A tabu search based clustering algorithm and its parallel implementation on spark. *Applied Soft Computing*, 63:97–109, 2018.

[108] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. `https://papers.nips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html`.

[109] O. Lüdtke, A. Robitzsch, and S. West. Regression models involving nonlinear effects with missing data: A sequential modeling approach using bayesian estimation. *Psychological Methods*, 25:157–181, 09 2019.

[110] A. M and E. C. I. Large scale mine. algorithm for mining local outliers. In *Proceeding of the 15th information resource management association international conference, New Orleans*, pages 5–8, 2004.

[111] D. Ma. Recent advances in deep learning based computer vision. In *2022 International Conference on Computers, Information Processing and Advanced Education (CIPAE)*, pages 174–179, 2022.

[112] H. Madsen. *Time series analysis*. Chapman & Hall, 2007.

[113] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *arXiv*, 2016. https://arxiv.org/abs/1607.00148.

[114] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff. Timenet: Pretrained deep recurrent neural network for time series classification. In *25th European Symposium on Artificial Neural Networks, ESANN 2017, Bruges, Belgium, April 26-28, 2017*, 2017.

[115] Y. Mao, H. Jamjoom, S. Tao, and J. M. Smith. Networkmd: topology inference and failure diagnosis in the last mile. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, page 189–202, New York, NY, USA, 2007. Association for Computing Machinery.

[116] A. Maćkiewicz and W. Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[117] G. Mclachlan. Mahalanobis distance. *Resonance*, 4:20–26, 06 1999.

[118] B. Mehlig. *Machine Learning with Neural Networks*. Cambridge University Press, 10 2021. https://www.cambridge.org/core/books/machine-learning-with-neural-networks/0028D1883AF842C81340508119AB6491.

[119] B. Memarian and T. Doleck. Chatgpt in education: Methods, potentials, and limitations. *Computers in Human Behavior: Artificial Humans*, 1(2):100022, 2023.

[120] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 26. Curran Associates, Inc., 2013.

[121] J. A. Miller, M. Aldosari, F. Saeed, N. H. Barna, S. Rana, I. B. Arpinar, and N. Liu. A survey of deep learning and foundation models for time series forecasting. *arXiv*, 2024. https://arxiv.org/abs/2401.13912.

[122] Ministeriet for Digitalisering of Ligestilling. Lov om net- of informationssikkerhed for domænenavnssystemer og visse digital tjenester. https://www.retsinformation.dk/eli/lta/2018/436, 2018. (Danish - accessed on 16 April 2024).

[123] U. Mori, A. Mendiburu, and J. A. Lozano. Distance measures for time series in r: The tsdist package. *The R Journal*, 8(2):451–459, 2016.

[124] J. Määttä and T. Roos. Maximum parsimony and the skewness test: A simulation study of the limits of applicability. *PloS one*, 11:e0152656, 04 2016.

[125] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? In *3rd Workshop on Bayesian Deep Learning (NeurIPS 2018)*, 2019.

[126] E. Nalisnick, A. Matsukawa, Y. W. Teh, and B. Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality. In *4th workshop on Bayesian Deep Learning (NeurIPS 2019)*, 2019.

[127] Network Operations Subcommittee. Mathematics of cable. Standard SCTE 270 2021r1, Society of Cable Telecommunications Engineers, Inc., 2021. `https://www.scte.org/standards/library/catalog/scte-270-mathematics-of-cable/`.

[128] OpenAI. *ChatGPT response to: "Please write a poem about representation learning in time series and fault identification in hybrid-fiber coaxial networks in general"*, 2024.

[129] R. Padmanabhan, A. Schulman, D. Levin, and N. Spring. Residential links under the weather. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 145–158, New York, NY, USA, 2019. Association for Computing Machinery. `https://dl.acm.org/doi/10.1145/3341302.3342084`.

[130] L. Pan, J. Zhang, P. P. Lee, H. Cheng, C. He, C. He, and K. Zhang. An intelligent customer care assistant system for large-scale cellular network diagnosis. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1951–1959, New York, NY, USA, 2017. Association for Computing Machinery.

[131] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*, pages 315–326. IEEE, 2003.

[132] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[133] M. Parthasarathy. A dissimilarity measure for comparing subsets of data: Application to multivariate time series. *Temporal Data Min. Alg. Theory Appl*, 101:1–12, 2005.

[134] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[135] R. Patterson and E. Rolland. Hybrid fiber coaxial network design. *Operations Research*, 50:538–551, 06 2002.

[136] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, pages 2825–2830, 2011. `https://jmlr.org/papers/v12/pedregosa11a.html`.

[137] T. X. Pham, P. Siarry, and H. Oulhadj. Integrating fuzzy entropy clustering with an improved pso for mri brain image segmentation. *Applied Soft Computing*, 65:230–242, 2018.

[138] D. Pisinger and S. Sørensen. Topology reconstruction using time series data in telecommunication networks. *Networks*, 83(2):408–427, 2024.

[139] Promptlink Communications, Inc. What is broadband network noise and why is it difficult to find? `https://www.promptlink.com/media-library/blog/what-is-broadband-network-noise-and-why-is-it-difficult-to-find.html`, 2019. (accessed on 13 December 2023).

[140] Promptlink Communications, Inc. What is cable ingress? `https://www.promptlink.com/media-library/blog/what-is-cable-ingress.html`, 2023. (accessed on 13 December 2023).

[141] Z. Qiao, J. He, J. Cao, G. Huang, and P. Zhang. Multiple time series anomaly detection based on compression and correlation analysis: A medical surveillance case study. In Q. Z. Sheng, G. Wang, C. S. Jensen, and G. Xu, editors, *Web Technologies and Applications*, pages 294–305, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[142] H. Qiu, Y. Liu, N. A. Subrahmanya, and W. Li. Granger causality for time-series anomaly detection. In *2012 IEEE 12th International Conference on Data Mining*, pages 1074–1079, 2012.

[143] R. Raj, A. Singh, V. Kumar, and P. Verma. Analyzing the potential benefits and use cases of chatgpt as a tool for improving the efficiency and effectiveness of business operations. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 3(3):100140, 2023.

[144] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.

[145] T. Rasmussen, L. Clemmensen, and A. Baum. Compressing cnn kernels for videos using tucker decompositions: Towards lightweight cnn applications. *Proceedings of the Northern Lights Deep Learning Workshop*, 3, 2022. Northern Lights Deep Learning Workshop 2022, NLDL 2022 ; Conference date: 10-01-2022 Through 12-01-2022.

[146] T. E. Rasmussen. Evaluation of synergistic effects of tensor decomposition methods within (deep) neural network applications, 2021. MSc thesis. Technical University of Denmark.

[147] T. E. Rasmussen and S. Sørensen. Encoding binary events from continuous time series in rooted trees using contrastive learning. *arXiv*, 2024. `https://arxiv.org/abs/2401.01242`.

[148] T. E. Rasmussen, S. Sørensen, D. Pisinger, T. M. Jørgensen, and A. Baum. Topology reconstruction in telecommunication networks: Embedding operations research within deep learning, 2024. Preprint available at SSRN: `https://ssrn.com/abstract=4757707`.

[149] K. Rasul, A.-S. Sheikh, I. Schuster, U. Bergmann, and R. Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows. *arXiv*, 2021. `https://arxiv.org/abs/2002.06103`.

[150] P. P. Ray. Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, 3:121–154, 2023.

[151] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. `https://doi.org/10.1137/0108018`.

[152] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. In *Encyclopedia of Database Systems*, pages 532–538. Springer US, 2009. `https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_565`.

[153] J. Ren, H. Li, C. Hu, and H. He. Odmc: Outlier detection on multivariate time series data based on clustering. *Journal of Convergence Information Technology*, 6:70–77, 02 2011.

[154] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

[155] H. Rezgui. An overview of optical fibers. *Global Journal of Science Frontier Research*, 21:14–20, 01 2022.

[156] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[157] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct. 1986.

[158] J. Rupe. *PNM Current Methods and Practices in HFC Networks (DOCSIS 3.1)*. Cable Television Laboratories, 5th edition, 2023. `https://www.cablelabs.com/specifications/CM-GL-PNM-3.1`.

[159] J. Rupe and J. Zhu. Kickstarting proactive network maintenance with the proactive operations platform and example application. In *Proceedings of the Fall Technical Forum*, New Orleans, LA, 2019. SCTE, ISBE, NCTA. `https://www.nctatechnicalpapers.com/Paper/2019/2019-kickstarting-proactive-network-maintenance`.

[160] I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational statistics & data analysis*, 23(1):153–168, 1996.

[161] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv*, 2024. `https://arxiv.org/abs/2402.07927`.

[162] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975.

[163] R. Schmidt-Fajlik. Chatgpt as a grammar checker for japanese english language learners: A comparison with grammarly and prowritingaid. *AsiaCALL Online Journal*, 14:105–119, 06 2023.

[164] SCTE. *Test Procedure for Common Path Distortion (CPD)*. SCTE and ISBE, 2019. `https://www.scte.org/standards/library/catalog/scte-109-test-procedure-for-common-path-distortion/`.

[165] J. Serrà, D. Álvarez, V. Gómez, O. Slizovskaia, J. Núñez, and J. Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In *The 8th International Conference on Learning Representations (ICLR 2020)*, 03 2020.

[166] A. Shafaei, M. W. Schmidt, and J. Little. A less biased evaluation of out-of-distribution sample detectors. In *British Machine Vision Conference*, 2018.

[167] N. Sharma, V. Jain, and A. Mishra. An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132:377–384, 2018. International Conference on Computational Intelligence and Data Science.

[168] K. Shaukat, T. M. Alam, S. Luo, S. Shabbir, I. A. Hameed, J. Li, S. K. Abbas, and U. Javed. *A Review of Time-Series Anomaly Detection Techniques: A Step to Future Perspectives*, pages 865–877. Springer International Publishing, Cham, 2021.

[169] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, S. Chen, D. Liu, and J. Li. Performance comparison and current challenges of using machine learning techniques in cybersecurity. *Energies*, 2020.

[170] C. Shi, J. Lu, Q. Sun, J. Zhou, W. Huang, and R. Xia. Multi-scale autoencoder for edge detection. *IEEE Access*, 10:116253–116260, 2022.

[171] A. Shokrollahi. Ldpc codes: An introduction. In K. Feng, H. Niederreiter, and C. Xing, editors, *Coding, Cryptography and Combinatorics*, pages 85–110, Basel, 2004. Birkhäuser Basel.

[172] M. Simakovic, Z. Cica, and D. Drajic. Big-data platform for performance monitoring of telecom-service-provider networks. *Electronics (Switzerland)*, 11(14), 2022. Cited by: 2; All Open Access, Gold Open Access.

[173] M. Simakovic and Z. čiča. Detection and localization of failures in hybrid fiber–coaxial network using big data platform. *MDPI Electronics*, 2021. `https://www.mdpi.com/2079-9292/10/23/2906`.

[174] M. N. Simaković, I. B. Masnikosa, and C. G. Zoran. Performance monitoring challenges in hfc networks. In *2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, pages 385–388, 2017.

[175] Starlink. How starlink works. `https://www.starlink.com/technology`, 2024. (accessed 23 April 2024).

[176] M. Steel and D. Penny. Parsimony, Likelihood, and the Role of Models in Molecular Phylogenetics. *Molecular Biology and Evolution*, 17(6):839–850, 06 2000.

[177] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2828–2837, 2019.

[178] S. Sun. OFDM - Orthogonal Frequency Division Multiplexing. `https://www.youtube.com/watch?v=KCHO7zlU25Q`, 10 2018. YouTube video.

[179] Y. Tang, F. Ren, and W. Pedrycz. Fuzzy c-means clustering through ssim and patch for image segmentation. *Applied Soft Computing*, 87:105928, 2020.

[180] TDC NET. Who are tdc net? `https://tdcnet.com/about-us/who-are-tdc-net/`. (accessed on 15 May 2023).

[181] The Editors of Encyclopaedia Britannica. Talking drum. Encyclopedia Britannica, `https://www.britannica.com/technology/talking-drum`, October 2015. (accessed on 15 April 2024).

[182] A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti. Explainable ai for time series classification: A review, taxonomy and research directions. *IEEE Access*, 10:100700–100724, 2022.

[183] M. Thill, W. Konen, H. Wang, and T. Bäck. Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing*, 112:107751, 2021.

[184] R. Thompson and R. Hranac. *PNM Best Practices: HFC Networks (DOCSIS 3.0)*. Cable Television Laboratories, 3rd edition, 2016. `https://www.cablelabs.com/specifications/proactive-network-maintenance-using-pre-equalization`.

[185] U.S. Department of Homelend Security: Cybersecurity & Infrastructure Security Agency. Information technology sector. `https://www.cisa.gov/topics/critical-infrastructure-security-and-resilience/critical-infrastructure-sectors/information-technology-sector`. (accessed on 16 April 2024).

[186] G. Vadlamudi, N. Vemuru, S. Vangapalli, R. K. Surapaneni, and S. Nimmagadda. Meeting summarizer using natural language processing. In *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1610–1614, 2022.

[187] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, page 125, 2016.

[188] A. van den Oord, O. Vinyals, and k. kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[189] B. Volpe. Docsis pre-equalization: Vastly powerful, often undervalued [white paper]. *ZCorum*, 2014. `https://api.semanticscholar.org/CorpusID:17932883`.

[190] X. Wang, F. Yu, and W. Pedrycz. An area-based shape distance measure of time series. *Applied Soft Computing*, 48:650–659, 2016.

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

[192] L. Weng. Flow-based deep generative models. `https://lilianweng.github.io/posts/2018-10-13-flow-models/`, 2018. (Accessed on 29 August 2024).

[193] T. Williams, B. Hamzeh, and R. Hranac. Field measurements of nonlinear distortion in digital cable plants. In *Proceedings of the Fall Technical Forum*. SCTE, ISBE, NCTA, 2014. `https://www.nctatechnicalpapers.com/Paper/2014/2014-field-measurements-of-nonlinear-distortion-in-digital-cable-plants`.

[194] T. H. Williams. System and method to locate common path distortion on cable systems. Patent US 2004O245995A1, Arcom Digital Patent LLC, 12 2004. `https://patents.google.com/patent/US20080319689`.

[195] L. Wolcott, M. O'Dell, P. Kuykendall, and V. Gopal. A pnm system using artificial intelligence, hfc network impairment, atmospheric and weather data to predict hfc network degradation and avert customer impact. In *Proceedings of the Fall Technical Forum*. SCTE, ISBE, 2018. `https://www.nctatechnicalpapers.com/Paper/2018/2018-a-pnm-system`.

[196] H.-S. Wu. A survey of research on anomaly detection for time series. In *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 426–431, 2016.

[197] K. Yamanishi and J.-i. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 676–681, New York, NY, USA, 2002. Association for Computing Machinery.

[198] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv*, 2015.

[199] M. Yu, A. Greenberg, D. Maltz, J. Rexford, L. Yuan, S. Kandula, and C. Kim. Profiling network performance for multi-tier data center applications. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011. https://www.usenix.org/conference/nsdi11/profiling-network-performance-multi-tier-data-center-applications.

[200] S.-S. Yu, S.-W. Chu, C.-M. Wang, Y.-K. Chan, and T.-C. Chang. Two improved k-means algorithms. *Applied Soft Computing*, 68:747–755, 2018.

[201] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1409–1416, Jul. 2019.

[202] Y. Zhang, J. Pan, W. Liu, Z. Chen, K. Li, J. Wang, Z. Liu, and H. Wei. Kullback-leibler divergence-based out-of-distribution detection with flow-based generative models. *IEEE Transactions on Knowledge and Data Engineering*, 36(4):1683–1697, 2024.

[203] J. Zhu, K. Sundaresan, and J. Rupe. Proactive network maintenance using fast, accurate anomaly localization and classification on 1-d data series. In *Proceedings of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2020. https://ieeexplore.ieee.org/document/9187045.

[204] D. Zwillinger and S. Kokoska. *CRC Standard Probability and Statistics Tables and Formulae*. Chapman & Hall, 2000. https://www.taylorfrancis.com/books/mono/10.1201/b16923/crc-standard-probability-statistics-tables-formulae-student-edition-stephen-kokoska-daniel-zwillinger.

Technical
University of
Denmark

Richard Petersens Plads, Building 324
2800 Kgs. Lyngby
Tlf. 4525 3031

www.compute.dtu.dk