technische universität
dortmund

Bachelor Thesis

ML-Driven Classification of Link Components in Passive Optical Networks

Maximilian Brügge September 2022

Supervisors:
Prof. Dr. Dr. Klaus-Tycho Förster
M. Sc. Stephanie Althoff

Technical University of Dortmund
Faculty of Computer Science
Chair for Computer Networks and
Distributed Systems (LS-4)
https://ls4-www.cs.tu-dortmund.de

Acknowledgement

First and foremost, I have to thank my supervisors at Technical University of Dortmund, Prof. Dr. Dr. Klaus-Tycho Förster and Stephanie Althoff. Without their assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished. I would like to thank you very much for your support and understanding over these past three months.

I would also like to give special thanks to my supervisors at ADVA Optical Networking SE, Dr. Sander Jansen and Jasper Müller. Both of them spent much time and were very caring by giving suggestions for improvement and helping me with problems during my thesis.

I would also like to show gratitude to ADVA Optical Networking SE, who offered me this thesis topic and supported me financially. Not only did this help focussing on my thesis, it also allowed me to build new relationships with colleagues and gain insight into working in the research field.

Most importantly, none of this could have happened without my family, especially my parents. They enabled me going to university and helped me to make my dream of becoming a computer scientist come true. In the tenth grade, my parents made it possible for me to take part in a student exchange. This helped me to consolidate my English and helped me grow as a person. I would like to deeply thank you for everything including the emotional support.

Abstract

Telecom operators deploy and operate large amounts of passive optical networks (PONs) delivering high-speed broadband internet to homes and small businesses. The maintenance and high-reliability requirements for such networks is a challenging task, helped by specialized fiber monitoring equipment such as optical time domain-reflectometer (OTDR). This thesis is focused on analyzing and interpreting OTDR traces using machine learning (ML) techniques. An OTDR trace data set of varying PON architectures is collected in a laboratory setup using commercial OTDR equipment. Two ML approaches for event identification and localization in OTDR traces are compared. The results show that a two-stage approach using a deterministic event localization algorithm and an ML based identification model comprehensively outperforms the one-stage approach of using a single ML model to localize and identify events. An accuracy of 98% is achieved for the important task of identifying optical network units (ONUs) while the accuracy over all types of events lies at ~90% . The gained information can be used to find problems and failures to reduce costs and increase work efficiency. Current state-of-the-art techniques mostly only use deterministic algorithms to detect and classify events happening in PONs. This approach is not ideal for PONs because with slope calculation a classification cannot be made. Additionally, events can overlap in PONs and various levels of attenuation and noise result in different looking traces, making analysis difficult. Convolutional neural networks, however, can extract common characteristic to correctly classify each event. In this work, a deterministic algorithm is used to propose a position for a potential event which is to be classified by the convolutional neural network.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Objectives and Contributions . . . 2
1.3 Structure of this Thesis . . . 2

2 Theoretical Background **3**
2.1 Passive Optical Networks . . . 3
2.2 Machine Learning . . . 4
2.3 Convolutional Neural Networks . . . 5
2.4 Optical Time-Domain Reflectometer . . . 6

3 Related Work **9**
3.1 Time Series Classification . . . 9
3.2 OTDR Trace Analysis . . . 9

4 Implementation **11**
4.1 Lab Setup . . . 11
4.2 Data Representation, Visualization and Labeling . . . 11
4.3 Deterministic Detection . . . 12
4.4 Data Generation . . . 13
4.4.1 Constructing Artificial Traces . . . 16
4.4.2 Moving Events in Input Window . . . 16
4.5 Data Loading . . . 17
4.6 Classification . . . 19

5 Analysis **21**
5.1 Analysis of Constructing Artificial Traces . . . 21
5.2 Analysis of Moving Window Approach . . . 21
5.3 Performance on a Test Trace . . . 23
5.4 Comparison to State-of-the-Art Techniques . . . 25

6 Conclusion **27**
6.1 Resume . . . 27
6.2 Outlook . . . 27

Table of Figures **29**

List of Algorithms **30**

Bibliography **31**

Declaration **33**

Nomenclature

Sets

$\mathbb{N}$	Natural numbers
$\mathbb{N} \setminus \{0\}$	Natural numbers without 0
$\mathbb{R}$	Real numbers
$\mathbb{R}_{>0}$	Real numbers greater than 0

Abbreviations and Acronyms

PON	Passive Optical Network
NN	Neural Network
ML	Machine Learning
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
OTDR	Optical Time Domain-Reflectometer
OLT	Optical Line Terminal
ONU	Optical Network Unit
PC	Physical Connector
APC	Angeled Physical Connector
IoT	Internet of Things
GB	Gigabyte

1 Introduction

1.1 Motivation

The need for high-quality Internet with low latencies, great bandwidth and a high reliability is constantly increasing. This is not only because of the growing demand of high-resolution video streaming but also to cover the traffic from millions of IoT-devices and people working from home. This is why telecom operators such as British Telecom or Deutsche Telekom started speeding up the process of building passive optical networks (PONs), a cost effective fiber optic access technology where a single fiber from a network operator is split passively and fed to up to 128 house holds. A countries network is split into subnetworks called core, metro and access. PONs are predominantly used in the access part of the network. Currently and in most cities, only backbone networks are optical fiber, while the last meters to the customer's house are copper cable. Due to speed speed limitations, PONs replace this network architecture by having an optical fiber instead of copper cable to the house.

In PONs, different kinds of faults can occur (e.g. faulty installation, faulty equipment, construction caused fiber cut). Maintaining these networks to ensure reliability becomes a challenging and time-consuming task for operators. Today, many companies use optical time domain reflectometers (OTDR), such as *ADVA's ALM* (2016), to locate and analyse faults and events inside their fiber optic network. OTDRs exploit the physical effect of Rayleigh-backscattering to build a profile of the optical fiber network by sending a laser pulse into the fiber and detecting the backscattered light with a photo diode. Reflective events such as network components (connectors, splitters etc.) can be identified in the OTDR traces. In a PON, the fiber splits up into multiple fibers, hence the backscattering light overlays. The traces become harder to analyse by a human or deterministic algorithms. Having an expert analysing traces is not only time consuming but also expensive. The goal of this work is to find a satisfying and automatic solution analysing OTDR-traces for the special case of PONs.

1.2 Objectives and Contributions

The main goal of this work is to develop a fast and efficient way to find and classify events in PONs. It should outperform state-of-the-art deterministic approaches and be reliable with its classification. To achieve this, a laboratory setup is designed for training and verification data generation. To generate more data, two algorithms which generate unique traces out of recorded events were developed and are presented in section 4.4. To visualize and label all data generated by either the ALM or the trace generating programm, a custom tool is build for this purpose. It also includes a deterministic event detector which visualizes potential events and therefore supports the user for fast manual labeling. After generating large set of training data, a convolutional neural network is trained for classifying and distinguishing between potential events. Continuing, for the neural network to load new traces to classify, the data must be preprocessed and then given to the network which classifies the correct event type accordingly.

The event classification can be used to detect faults and problems in links and PONs, respectively. The classification should be reliable having a high accuracy but, most importantly, it should be able to correctly classify optical network units (ONUs). They are a critical part in PONs and often remain undetected when ONUs have distribution fibers with similar length. The classification accuracy of ONUs is reliably about 98% in training evaluation.

1.3 Structure of this Thesis

In the following there is a short introduction to neural networks and OTDR-traces which are crucial to understand the problem definition (chapter 2). In chapter 3, related work done in time-series classification and OTDR trace analysis is discussed. Furthermore in chapter 4, the implementation which is done by using state-of-the-art convolutional neural networks and new developed algorithms is shown. Two approaches of generating training data for the convolutional neural network is presented in section 4.4 which are compared in chapter 5. In this chapter, the overall performance of the network is shown including a comparison between two data generation approaches presented in 4.4. Finally, the work will be concluded with a summary of the achieved results and an outlook on future work.

2 Theoretical Background

This chapter provides the necessary background knowledge on machine learning, neural networks and optical time domain reflectometers. This knowledge is crucial to understand the problem definition as well as its implementation and proposed solution.

2.1 Passive Optical Networks

Over the last decades, the focus of network operators has shifted from telephony to internet and the bandwidth demands have risen dramatically. As a result, the telephone cable does not scale with the bandwidth demand anymore such that telecom companies are migrating their network to PONs. Light, with its short wavelength, has huge advantages over electrical communication in terms of bandwidth. That is why today nearly every backbone cables in core or metro networks consists of optical fibers. An optical connection is realized by connecting multiple optical Fibers together. These connections are realized with either splices or connectors. In this thesis, we will focus on two connectors, which are fiber optic connectors with a threaded body, which are commonly used in datacom and telecom and are important for this project. There are different types of connectors, such as SC, FC, LC, E2000, etc. However, regardless which connector is used, all of these connectors are either angled (APC) or straight (PC). In the following, PC (physical connectors) and APC (angled physical connector) is used for simplicity. All connectors suffer from small attenuation since a connector is an interruption in the cable. PC and APC differ by their shape. Because the PC is not angled, the reflections in terms of backscattering light (see section 2.4) are much higher compared to APCs which can be seen in figure 4.7. APCs, however, do not have a measurable reflection because they are angled by 8°. Consequently, only the insertion loss of 0 to $0.5[dB]$ can be found on traces. (Keiser 2003)

A Passive Optical Network (Figure 2.1) is a optical fiber technology which typically is used to deliver high-speed internet to the customers home. As the name indicates, these optical access networks consist of passive components in the area between the central office and the customer premises. A service starts at the optical line terminal (OLT) and is then transmitted over a typically 10-15km feeder fiber to a power splitter.

Figure 2.1 depicts a typical PON network with fiber monitoring. As can be seen in this figure, a power splitter is a passive component which splits power onto N different fibers resulting in less than $1/N$ power on a split cable due to loss. This is connected to an ONU installed at the customer's house. The ONU converts the optical into electrical signal (such as Ethernet). Because ONUs often do not have great reflection, reflectors are used together with ONUs to make them visible on the OTDR trace. Power splitters are passive components which means data is transmitted via time division multiplexing. Distribution fibers between power splitters and ONUs are generally not longer than 5km. (Lam 2007)

Fiber optic communication has great advantages in comparison to coaxial copper

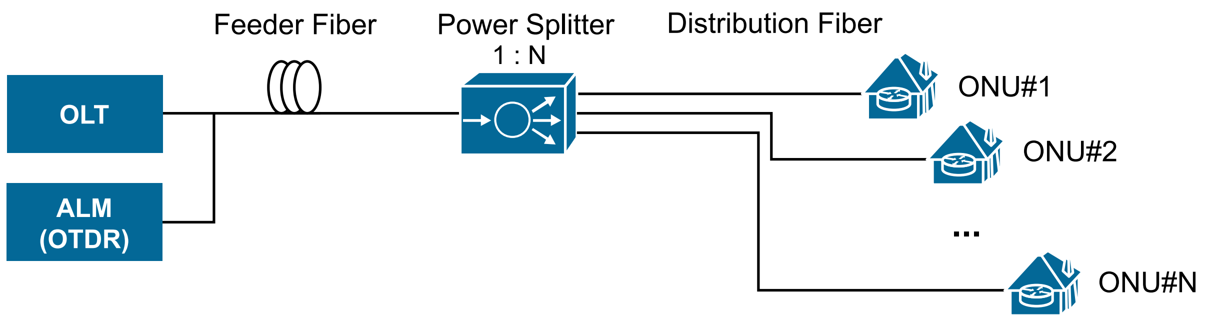

Figure 2.1: Simplified Passive Optical Network

fibers. They have a higher bandwidth and can transport data over long distances. (Babani et al. 2014)

2.2 Machine Learning

Machine learning is a generic term for a machine inferring knowledge out of experience and a sub-field of artificial intelligence. At training, data is given to the system which uses statistical models to find patterns. In machine learning, a feature is an individually measurable characteristic, e.g. wind force in weather forecasting. (Janiesch, Zschech, and Heinrich 2021) In this work, we categorize data or events into different classes. This process called classification often starts with predicting the class of given data points. Classes are often referred to as target, label or categories.

Neural networks (see section 2.3) which are a subset of machine learning must often be trained on massive data sets. There are various methods of training in machine learning, however, this work discusses supervised and unsupervised training. Having an algorithm searching for patterns and distributions in training data and classifying it e.g. into clusters is called unsupervised training. Clusters are groups of elements that share common properties. It is called unsupervised because the training processed has not be monitored. Consequently, the training data is unlabled leaving this process to the clustering or extracting data algorithm. O'Shea and Nash (2015) and Rutigliano et al. (2021) used unsupervised training for detecting events in OTDR traces. Because of the lack of interaction and checking in the training process, unsupervised training usually takes less time. One problem to mention is the performance of the neural network (NN) is strongly dependent on the dataset and data pre-processing. Supervised training, on the other hand, leads to more accurate results but takes also more time to prepare. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. This type of learning can only be done when knowing the correct label to a training element. Having accurately labeled training data is crucial for an algorithm being able to correctly predict new (unlabeled) cases (Berry, Mohamed, and Yap 2019; Janiesch, Zschech, and Heinrich 2021; O'Shea and Nash 2015).

2.3 Convolutional Neural Networks

A neural network (NN) consists of neurons n_i for $i \in \mathbb{N}$ which together represent parametric functions $L \in \mathbb{N}$ called layers, such as $l_j \in L$ and $j \in \{1, ..., L\}$. Each neuron is a non-linear function which has an input domain where $k \in \mathbb{N}$. $w_{i,k} \in \mathbb{R}$ denotes an edge containing transformation parameters called weights. The output of a neuron can be calculated by its activation function which is also influenced by the neurons transformation parameters. An activation function defines the neurons output by applying the function on the input. Each neuron is connected to at least one output of a previous layer neuron or an input respectively. If every neuron is connected to every neuron of the previous layer the network is called "fully-connected". A network is considered "deep" when there are at least two layers of neurons. (Fauvel et al. 2021)

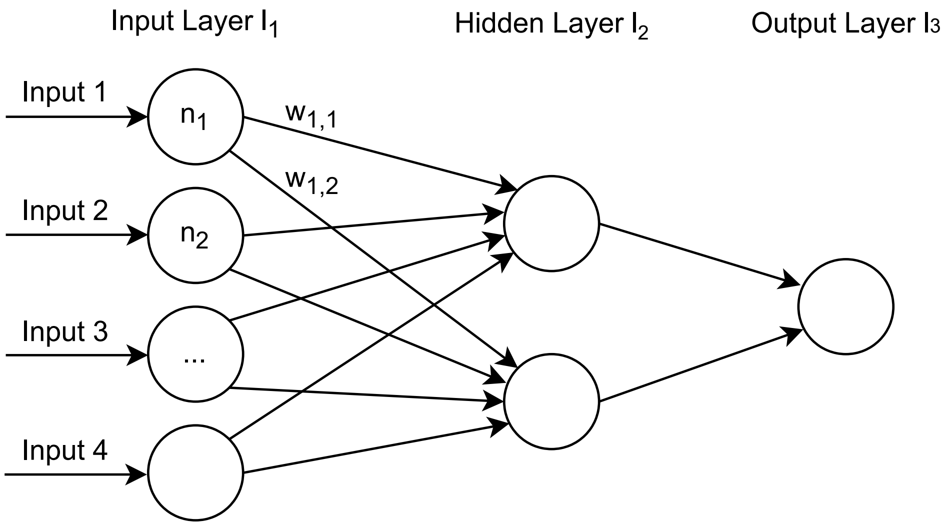

Figure 2.2: Simple three layered fully-connected neural network based on Fauvel et al. (2021) and O'Shea and Nash (2015)

Giving the success of CNNs in image detection and speech recognition, it is natural to think about using it in time series classification. Being a relatively lightweight NN compared to recurrent neural or long short-term memory networks, it is also perfect for commercial use. In a CNN, instead of being fully-connected, the neurons in a layer are only connected to a small region of the previous layer. A layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume. For explaining CNNs in detail, it is often referred to images since they can be plotted easily. In neural networks, features are the elements of the input vectors, meaning the number of pixel in images. Convolution is the process of applying a filter to an input. Often the resulting data is then scaled up or down, so that information is compressed or expanded. Pooling, on the other hand, can be understood as enhancing the power of convolutions. This is, we are going to take groups of pixels and perform an aggregation over them. One of the possible aggregations which can be done is taking the maximum value of the pixels. This concept is called "MaxPooling". When information is compressed with pooling, the wanted effect called feature extraction is to filter out unnecessary information.

The pooling and the convolution layer are operations that are applied to each of the input pixel (see figure 2.3), word or in this case time series data. The size of the matrix within the convolution or pooling operation is called kernel (O'Shea and Nash 2015; Tan et al. 2021). Although this is explained on a two dimensional vector, it can also be applied to vectors with other dimensions, e.g. with one dimension in the case of time series.

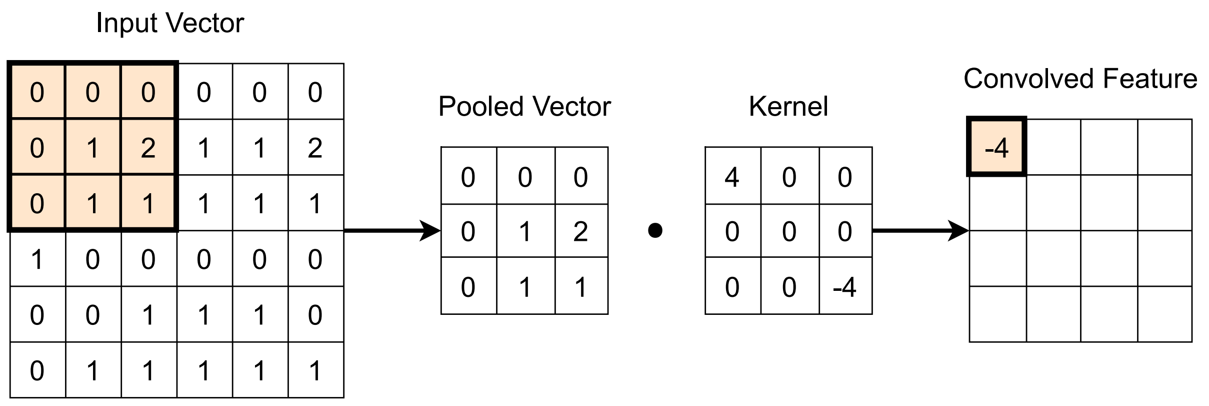

Figure 2.3: Visual representation of a convolutional layer based on O'Shea and Nash (2015)

2.4 Optical Time-Domain Reflectometer

Optical time-domain reflectometry (OTDR) is the state-of-the-art technique monitoring optical networks. An OTDR works with a physical phenomenon called Rayleigh scattering which is the backscattering or reflection of light by particles much smaller than the wavelength of the radiated light. It is measurable in optical fiber when a pulsed laser is coupled into a glass fiber. The backscattering light is then detected by a photo diode. Barnoski and Jensen invented OTDRs in 1976 (Barnoski and Jensen 1976).

ADVA's ALM (2016) can emit pulse widths down to 5ns which is the laser impulse duration. An OTDR, like the ALM, is not only sending one light pulse. It sends thousands of impulses during the previously set measurement time and averages all traces to a single trace. This reduces noise and leads to a higher accuracy and resolution. Different compromises are made when selecting the right pulse-width and measurement times. Having a short pulse width leads to higher resolution since the backscattering light stimulates the diode only for a short time period. At the same time, with short light impulses the total power emitted is also lower leading to high noise and a low signal after the power splitter which can effect the visibility of components on the trace. Often, in telecom networks, a technician works with a low pluse width (10ns) and a relatively high pulse width ($\sim 100 - 200ns$). Due to measurement times, cost is a crucial factor. Having short measurement times increase fast fault detection but also decrease trace quality. The ALM uses different pulse widths, depending on the application. For PON networks, a 10ns pulse width is typically used. The ALM supports distances up to 160km nominal reach for access, metro and core applications. The measurement signal has a wavelength of 1650nm which is outside user traffic wavelengths.

3 Related Work

Research in automized OTDR-trace analysis has been mainly focused on deterministic algorithms. The state-of-the-art in this area is presented in sections 3.2. Methods and learnings from the related popular research areas of time series classification as well as object detection in images can be transferred to the task of OTDR-trace analysis. Relevant work is listed in section 3.1.

3.1 Time Series Classification

Neural networks are commonly associated with media such as video, audio and images. In the case of gray-scale images, we can represent them by using a 2D-Array. Time series, however, can be represented by a 1D-Array. Like in Xiao et al. (2020) convolutional neural network are widely use to detect objects in images. Hence, a convolutional neural network made for time-series-classification is used for this work. Often, events in time series are detected but not classified meaning there is no specification about the event type. In this work, however, we want to detect and classify events.

There are some constraints for choosing the right NN model. The model should reflect the state of the art and ideally be used often. It should also be optimized for time series. The paper by Ismail Fawaz et al. (2019) is accessed over 60k and cited 894 times on Springer Link (Sept. 13. 2022) making it relevant and state-of-the-art. Oguiza (2022) is an implementation based on papers by Ismail Fawaz et al. (2019) and Wang, Yan, and Oates (2017) which is optimized for time series classification and hence used for this work.

3.2 OTDR Trace Analysis

In this thesis, the main objective is to find and classify all events within an OTDR trace. An event or component reflects the backscattering light differently than an ordinary optical fiber cable. Most approaches, as in Fernández et al. (2018) and Laferrière, Saget, and Cherncevère (1997), follow a deterministic approach meaning that detection works by either looking at the strength of attenuation or the slope of peaks. Both techniques combined have a relatively sufficient and reliable event detection quality when it comes to Point to Point Networks. But with PONs, however, these algorithms suffer from the fact that they cannot cope with overlapping events. Overlapping events can occur as ONUs can be placed with the same or slightly changed distance to a power splitter resulting in different looking traces. Often it is not apparent if multiple ONUs are overlapping which makes fault detection even more difficult.

Some works, such as Nedelkoski, Cardoso, and Kao (2019), use anomaly-detection which can only find the position of an event. The biggest problem with this method is the need of a second network to classify the type of event. Rutigliano et al. (2021) also uses two neural networks. Here, the first network extracts features from a trace which are fed into a region proposal network and a 1D event-detection convolutional neural network. The regional proposal network also forwards its output to the detection network. Every implementation done in this paper is explained on high level which purports that it cannot be reimplemented and tested against the proposed solution in this work. Results from Rutigliano et al. (2021) were tested against existing solution currently embedded in "Cisco NCS-1001". As it is hard to get such devices it is also not possible to also test against it.

All paper briefly explain how they perform event detection but do not provide an implementation for it. In case of deterministic detection, a slope-based algorithm is used. The same applies to *ADVA's ALM* (2016) which has a built in event detection feature called "fingerprint". With this method it is not possible to classify different component as the slope based algorithm only uses a threshold to detect events.

4 Implementation

This chapter describes the implementation and specific use of the previously presented theoretical background for the classification of events in PONs. In the first part of this chapter, the laboratory work is presented. All implemented algorithms are presented and explained afterwards. Consequently, the generation of training data is shown. This data is then handed to the CNN which can be seen in section 4.6.

4.1 Lab Setup

For the purpose of this work, many traces have been produced in order to understand the function and behaviour of *ADVA's ALM* (2016) and components. To test, verify and produce traces, a laboratory setup is designed and used which includes the mentioned ALM, different optical fibers with physical connectors (PCs) and angeled physical connectors (APCs), power splitters, reflectors, attenuators and optical network units (ONU). All components were present multiple times ensuring a widely spread and representative data set from which more artificial events can be created.

The ALM supports the simple network management protocol (*snmp* 1990) which has been used together with a python script to set parameters, start measurements and save the resulting file containing the trace. The trace file contains a list of all parameters which have been set as well as the trace information containing x and y value separated by a comma.

4.2 Data Representation, Visualization and Labeling

The OTDR traces contain time-series data representing reflected power (see section 2.4) at a given distance from the measurement device normalized to the reflected power received at distance zero. These information contained in the trace file are difficult to interpret for humans. In order to supervise the training and visualize results from the classifier, it is useful to have a tailored tool which helps representing, labeling and visualizing data. A tool called "Smart-ALM-UI" (figure 4.2) was developed with the help of ReactJS and NodeJS which makes it very lightweight and easy to distribute.

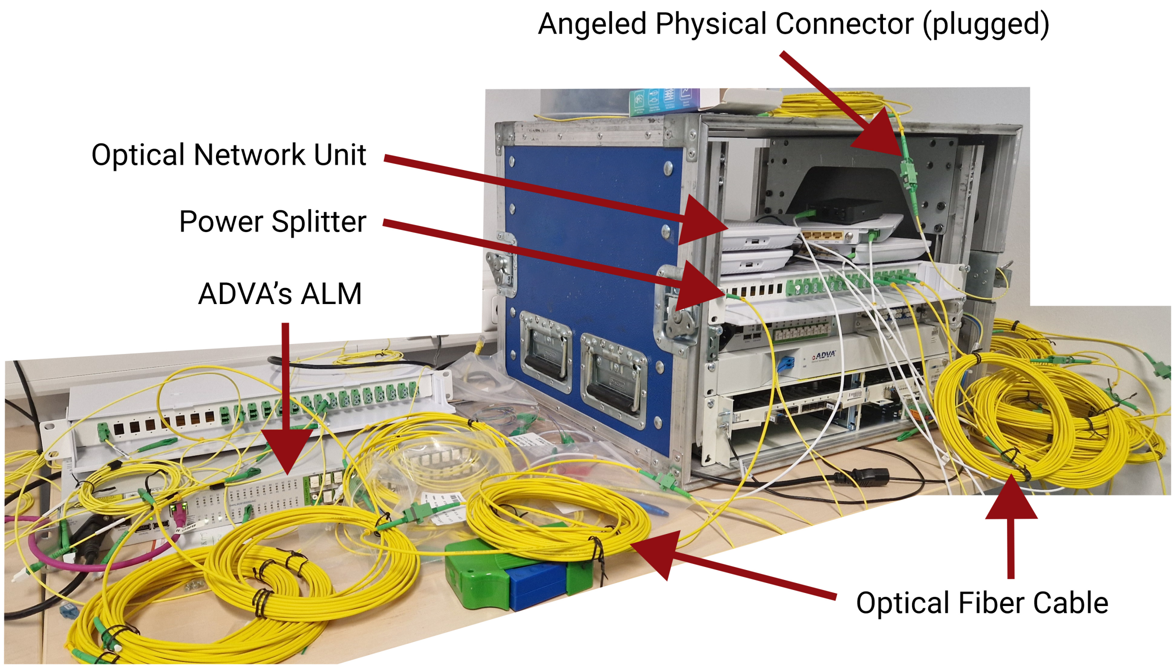

Figure 4.1: Laboratory setup for test trace (see figure 5.5)

This program not only helps generating training data through a manual labeling function, it can also detect events by automatically calculating the slope of a point in the trace. This is the same algorithm used for region proposal explained in section 4.3. In the figure 4.2 it can be seen how all events were manually labeled. To label an event, it has to be clicked at the start and end position resulting in an interval which then can be specified with the correct event. This work is essential as the events are used for artificially creating events later in this process. After saving the file, the loaded CSV contains now all trace information as well as the event intervals and event identification numbers.

4.3 Deterministic Detection

A deterministic algorithm for event detection is used to find the position of events. Unlike other works, the resulting intervals containing start and end points of potential events are given to the CNN which must then classify the correct event discussed in section 4.6.

Having a deterministic algorithm for region proposing has great advantages. One to mention is that the training of a second (region-proposal-)network is not necessary which reduces complexity and the need of generating extra training data. In this specific case, the algorithm is of such great effectiveness, a NN would add more error to the system. Another advantage is the elimination of a none-class in the classifier (see section 4.6). If we had no region proposing, on the other hand, the classifier must additionally distinguish between events and no event at all, since a sliding input window

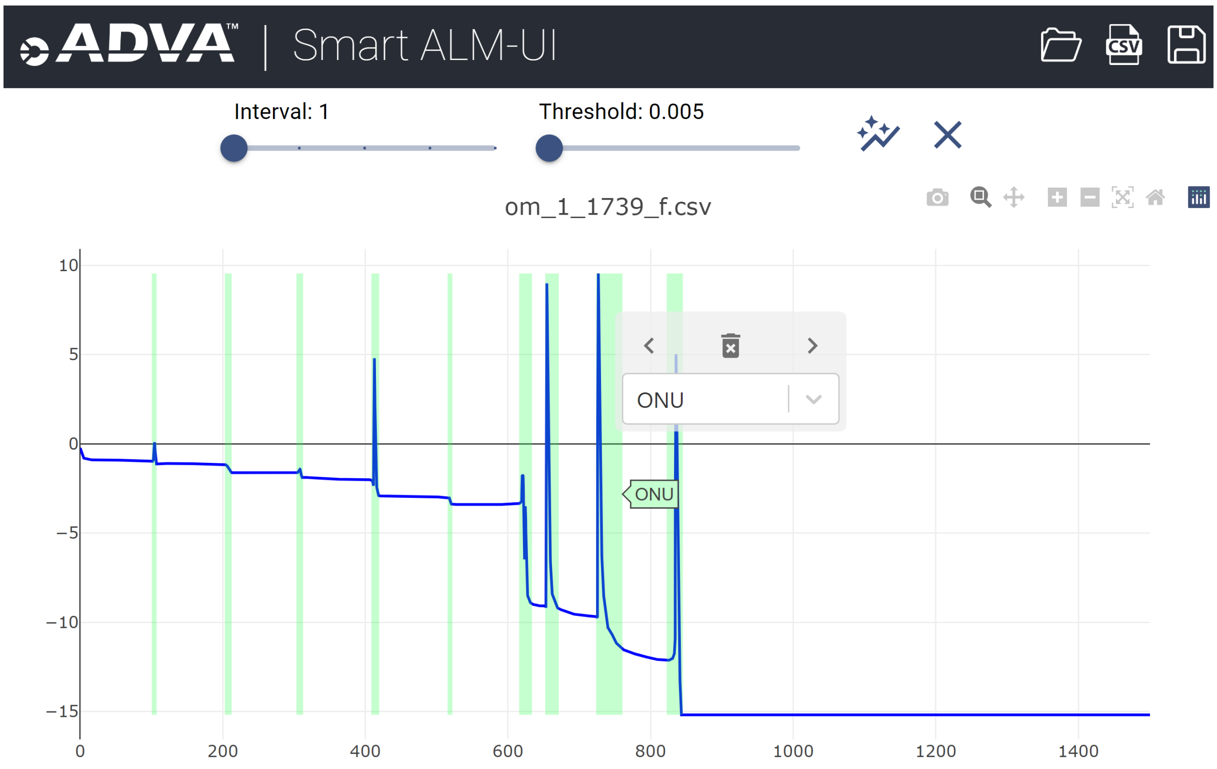

Figure 4.2: Smart-ALM-UI with a labelled trace (Web-browser View)

could also contain no event.

The algorithm works by looking at the slope s of a point. If it is over a certain threshold $t \in \mathbb{R}_{>0}$, such that $s > t$, it will be considered as an event. It makes sense to have events marked as intervals instead of single point events. Let X be an array containing the x-values of a trace and $X[x_1, x_1]$ be an interval considered as an event whereas x_1 denotes an index of X. Let $i \in \mathbb{N} \setminus \{0\}$. If at least one of the next i points is considered as an event, then $X[x_1, x_2]$ is the interval of an event where $x_2 \in [x_1 + 1, x_1 + i]$ (see algorithm 4.1).

To test the effectiveness of the proposed deterministic algorithm, it was implemented in the Smart-ALM-UI. The results can be seen in figure 4.3. The red marked areas are proposed events detected by the algorithm using a slope threshold of 0.0041 dB/m. During tests with the Smart-ALM-UI it was shown, a threshold of 0.001 and 0.05 $[dB/m]$ works best for event position proposing.

4.4 Data Generation

This section explains how data for the CNN was generated. For this purpose, a tailored tool called “Smart-ALM-UI” was developed helping visualizing and labeling events for training data (section 4.2). Now, two approaches for generating data are presented

Algorithm 4.1.: Find Event Positions

Require: X: array of x values of trace, Y: array of y values of trace, t: slope threshold, i: interval-threshold

```
1:  function FINDEVENTS($X, Y, t, i$)
2:      $E \leftarrow \emptyset$                                  ▷ Initializing empty events array
3:      for $i = 0$ To length($Y$) $-1$ do
4:          $s \leftarrow \mid Y[i+1] - Y[i] \ / \ X[i+1] - X[i] \mid$          ▷ Calculate slope
5:          if $s > t$ then                                  ▷ If slope greater than threshold
6:              $l \leftarrow$ length($E$)
7:              if $l > 1$ and $i - E[l-1][1] - i <= 0$ then
8:                  $E[l-1][1] = i$
9:              else
10:                 $e \leftarrow [i, i]$          ▷ $e[0]$ denotes start, $e[1]$ end x value of an interval
11:                 $E$.append($e$)
    return $E$
```

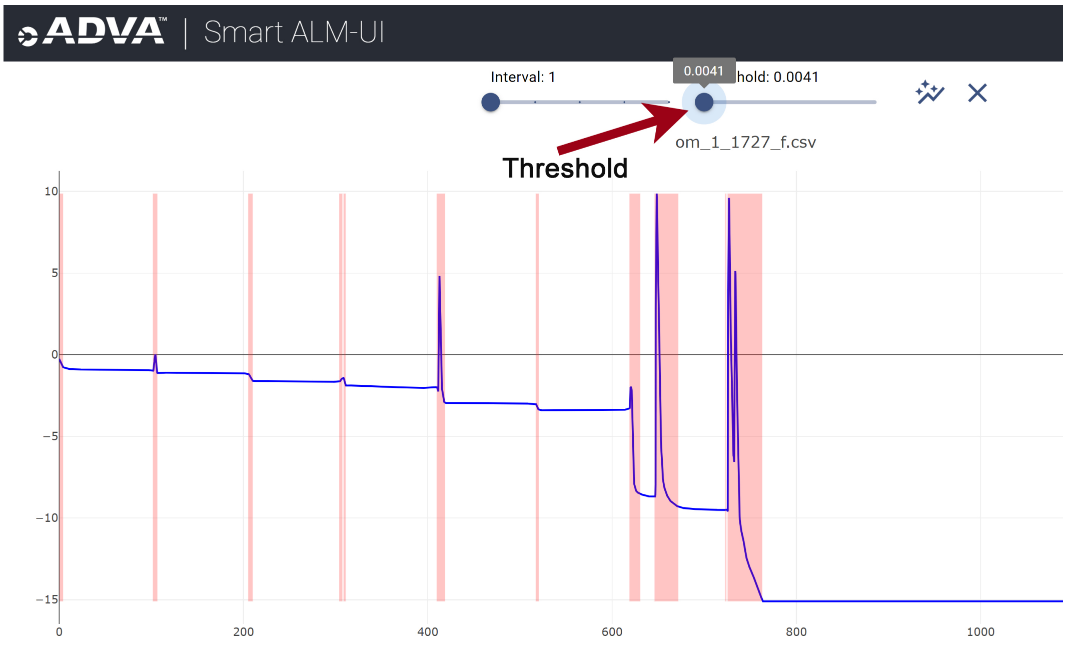

Figure 4.3: Detection algorithm in Smart-ALM-UI

which results are analysed in chapter 5.

In order to gain high accuracy, NNs must be trained on relatively large data sets. In lab, a variety of traces cannot be produced in reasonable time. Having someone creating and label traces is an unmanageable task. Accordingly, an algorithm has been devised to generate a sufficient amount of realistic training data in a reasonable time frame. Therefore, I created a laboratory setup with ADVA's ALM and the required hardware i.e. connectors, power splitter as well as ONUs. The classifier should be able to distinguish between angled physical connectors (APC), physical connectors (PC) power splitters and ONUs.

During trace experiments it has been discovered that no matter how much attenuation or noise is inserted, the characteristic of an event remains the same. This can be seen in figure 4.4 and 4.5, whereas both APC events have been cut out and zeroed for comparison. The laboratory setup for both events can be seen in figure 4.6.

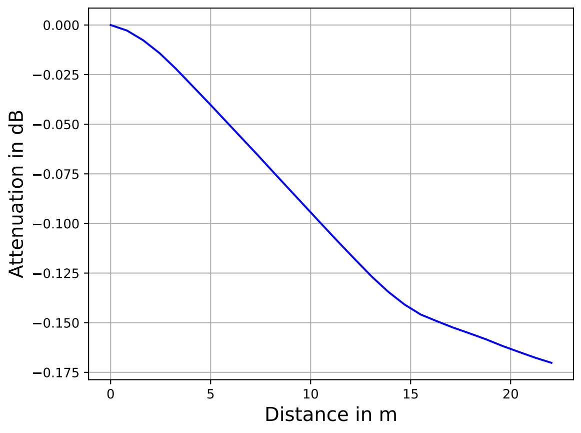

Figure 4.4: APC after 2dB attenuation

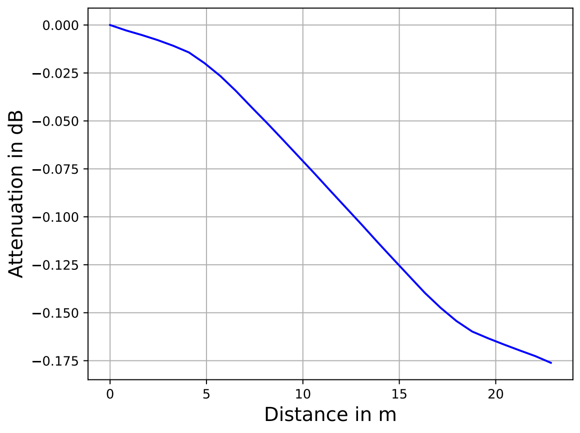

Figure 4.5: APC after 10dB attenuation

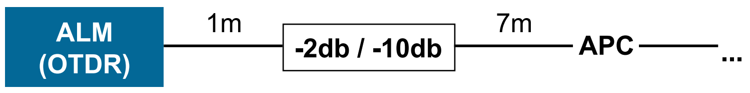

Figure 4.6: Laboratory setup for recording an APC trace

With this knowledge, many traces with different levels of attenuation as well as positions were generated. After having all traces present, the Smart-ALM-UI is used to label each event accordingly. Aiming to generate artificial traces, each event has to be slightly changed to simulate in field noise, disturbances as well as different attenuation levels. This happens by changing the length of an event as well as its amplitude level. Since all encapsulated events are zeroed, a scale factor s can be applied to the set of y-values changing the amplitude level. In order to stretch an event on the x-axis, cubic interpolation was used to readjust to the ALMs resolution.

Algorithm 4.2.: Create New Events by Scaling

Require: e: event to be changed, n: Number of scaled events to be generated, P: Scaling factor interval

```
1: function scaleEvent($e, n, P$)
2:     $E \leftarrow \emptyset$                                  ▷ Create empty events array
3:     $F \leftarrow$ changeEventAmplitude($e, n, p$)    ▷ Create $n$ new events out of $e$ by applying a factor out of $P$
4:     for $f$ in $F$ do
5:         g $\leftarrow$ changeEventLength($e, n, p$)
6:         $E$.append(g)
7:     return E                                  ▷ Return new events with $|E| = n^2$
```

4.4.1 Constructing Artificial Traces

A PON has a certain structure which has an importance for artificially creating traces. This constraints, e.g. ONUs can only be placed after power splitters, have to be implemented into the algorithm.

The main idea is to cut out labeled events from traces which has been recorded in the laboratory (see sections 4.1 and 4.2). The next step is to change the size and amplitude for each event (algorithm 4.2) and putting them back together on random positions in a new trace by adhering the constraints mentioned. For the data being readable by the CNN, a preprocessing is needed. Hence, a technique called "sliding window" is applied. This turns the OTDR trace which is a time series into a supervised learning problem. A fixed window $w \in \mathbb{N}$ specifies the size of the data points processed at the same time. The window can be understood as a frame which moves over the trace from the start to the end. The step $s \in \mathbb{N}$ denotes the size of how much the window moves in order to get the next window to process. Having a trace with length n leads to $\lfloor (n - w)/s \rfloor$ frames or windows each having a length of w. In order to label the windows with the correct event for the training, the overlap of an event being inside the window is calculated. If more than one event is present inside a window, the window gets assigned to the event with the biggest overlapping.

4.4.2 Moving Events in Input Window

Another approach is to construct only a part trace like in 4.4.1. For a NN we need fixed input sizes, while an event can vary in its size. To solve this problem, we can add lines with a slope of $-0.0003835684\ dB/m$ in front and behind the encapsulated events (transition from figure 4.7 to 4.8). The line slope was calculated using traces recorded by ADVA's ALM. The starting position can also be changed such that the NN can learn to see events on different positions (figures 4.8, 4.9). Like in section 4.4.1, there will be also an amplitude changing, so that different attenuation level can be simulated. Algorithm 4.3 describes this process in more detail.

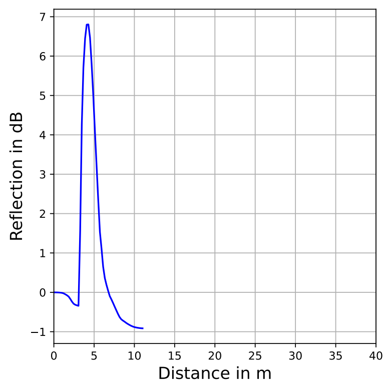

Figure 4.7: Encapsulated PC-event

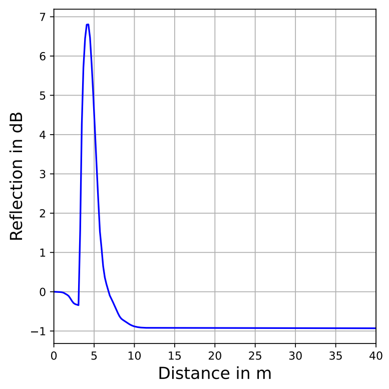

Figure 4.8: PC event, line added

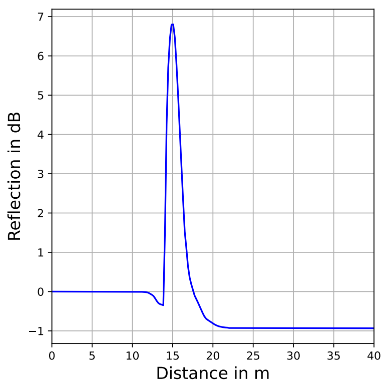

Figure 4.9: PC event, lines added, moved on x-axis

4.5 Data Loading

In section 4.4 it is explained how training data is generated and loaded for training the convolutional NN. In this section, it is going to be shown how data is loaded to the CNN which must be classified.

In the case of generating data by constructing artificial traces, we can use the sliding window technique mentioned in section 4.4.1 to load data.

With the moving event approach mentioned in section 4.4.2, however, we have to load data differently since we need to propose regions for events. Let s be the start and e the end index position of a proposed event resulting in the interval $[s, e]$. Let w be the window size and therefore the input size of the NN. Let $p = (w - e - s)/2$. We assume that $w > e - s$. To center the event, the y-values of the interval $[s - p, e + p]$ are forwarded to the NN.

Xiao et al. (2020) use the same structure with a region selector, feature extractor and classifier like it was proposed in this work which can be seen in Figure 4.10. The input is a time-series trace from ADVA's ALM. Then, a deterministic algorithm (section 4.3) is used to propose regions for the NN to classify.

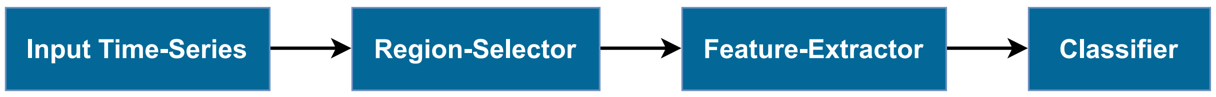

Figure 4.10: Basic architecture of traditional object detection algorithms based on Xiao et al. (2020)

Algorithm 4.3.: Generate Training Data - Moving Events

Require: T: Labeled OTDR traces, n: Number of scaled events to be generated, p: Scaling factor, i: Input size of neural network, s: Step size

```
1: function GENERATETRAININGDATA($T, n, p, i, s$)
2:     assert $i$ mod $s = 0$
3:     $E \leftarrow$ encapsulateEvents($T$)            ▷ Parse by label info of file content
4:     $Events \leftarrow \emptyset$                    ▷ Define empty Events array
5:     for $e$ in $E$ do
6:         $F \leftarrow$ scaleEvent($e$, $n$, $p$)      ▷ Algorithm 4.2
7:         $Events$.concat(F)                           ▷ Merge arrays $E$ with $F$
8:     $Traces \leftarrow \emptyset$
9:     for $ev$ in $Events$ do
10:        $r \leftarrow i/s$
11:        for $k = 0$ To $r - r/5$ do
12:            $t \leftarrow$ addLine(0, $k * s$)        ▷ Add line from position $x = 0$ to $x = i$
13:            $t$.addEvent($ev$)
14:            $l \leftarrow$ length($t$)
15:            $t \leftarrow$ addLine($l$, $i - l + 1$)
16:            $t \leftarrow$ cutTrace(0, $i$)           ▷ Cut created trace to match window size $i$
17:            $Traces$.append(t)
18:    return $Traces$
```

4.6 Classification

When people think about NNs, media such as video, audio and images come into their minds. In the case of gray-scale images, we can represent them by using a 2D-Array. Time series, however, can be represented by a 1D-Array. So it is more than important looking at work which was done in object detection in images. Since a CNN was used, the network tries to extract features from the input trace and then classifies it accordingly (see section 2.3). The generated data, explained in section 4.4, were used to train the proposed CNN. For the CNN, a PyTorch implementaion by Oguiza (2022) called FCN (fully-connected convolutional neural network) was used. It was explicitly made for time-series-classification. It is used in its standard configuration with three `ConvBlocks` each containing a 1D-convolutional and 1D-BatchNorm layer. ReLU (rectified linear unit) is used as activation function. The input size equals the window of generated traces. One-hot-encoding is used for representing each event class in a unbiased binary format. If we used ordinal integer instead, we would have different distances between classes, thus having greater errors when predicting wrongly. One-hot-encoding works by having n bits of classes and setting only one bit to 1 which corresponds to the “hot” state while having the others set to 0. The distance between all classes are equally spaced with one-hot-encoding (e.g. 10 and 01 for $n = 2$).

5 Analysis

In this chapter, the implementations presented in chapter 4 are tested and analysed. First, the two different approaches from sections 4.4.1 and 4.4.2 are tested against each other. All following performance tests done are based on the moving event approach (see section 4.4.2), since the constructing trace approach did not perform well as can be seen in section 5.1.

5.1 Analysis of Constructing Artificial Traces

In this section the first approach mentioned in section 4.4.1 is tested and analysed in its performance. Artificial traces are constructed out of encapsulated events recorded by ADVA's ALM and labeled with the Smart-ALM-UI. A sliding window is used to load the data and then used to train the neural network (section 4.5). The loaded data is then divided into a test and training set to prevent overfitting.

During test, a sliding window size of 60 brings the training loss to a minimum. The data generation algorithm were used to generate 15,000 unique traces which were loaded with a sliding window to train the CNN. 16 epochs with a learning rate of 0.0001 were set to bring the loss to its local minimum. The confusion matrix (see figure 5.1) shows how the network has problems distinguishing between the "none"-class and actual events. What seems most unintuitive is how ONUs are often classified as no event. By sliding over the entire trace with a fixed window size, there is a chance some windows will not contain an event. Thus, an additional "none" class must be added to the classifier. The neural network must then distinguish between four event types and the none-case. The output size is therefore $n = 5$.

Because of the poor results it has been decided not to pursue this approach further.

5.2 Analysis of Moving Window Approach

In this section, the second approach mentioned in section 4.4.2 is tested and analysed in its performance. Using moving events not only eliminates the none class but also

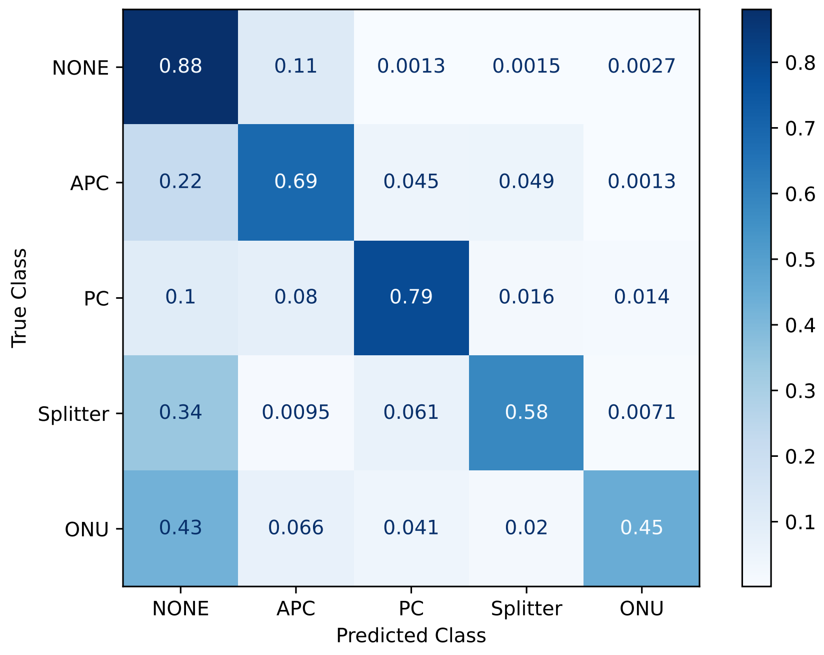

Figure 5.1: Confusion matrix after training with artificial traces

results in a decrease of event classes to distinguish between. It is always good to have as few classes as possible because this reduces the probability predicting the wrong class by $\frac{n-1}{n} - \frac{n-2}{n-1}$ where n denotes the number of classes.

The next step is to transmit specific locations of potential events to the network. This is done by a deterministic algorithm (4.1) which detects events by its slope (see section 4.3). The positions are used to load the data (section 4.5) which are passed to the CNN.

For the network, an input size of $i = 50$ is used (see algorithm 4.3). The input data is not normalized since this leads to poor results. In PONs, events often have their shape characteristics by how much attenuation occurs. When normalizing events, this information is lost. The data generation algorithm is used to generate 10,000 unique events which is loaded to train the CNN. 50 epochs with a learning rate of 0.0001 were set to bring the loss to its local minimum. One thing to note is that an increase of training data does not lead to better performance. Since the size of the loaded training data set is smaller as in section 5.2, more epochs are needed to bring the loss to its local minimum. The model is trained and executed on a windows machine equipped with an Intel Core i9® Core™ i9-10940X CPU, a NVIDIA GeForce RTX 3080 Ti GPU and 32GB of RAM, although it needs only 5GB on training. The training time is approximately 210 seconds on average. The execution time is around 0.0129 seconds on average. Since the calculation does not require a lot of resources, a regular computer is sufficient for this task and should perform at a similar speed.

Although, the classifier performs good with an accuracy of over 90%, figure 5.2 shows that it still has problems to distinguish between a power splitter and ONUs. However,

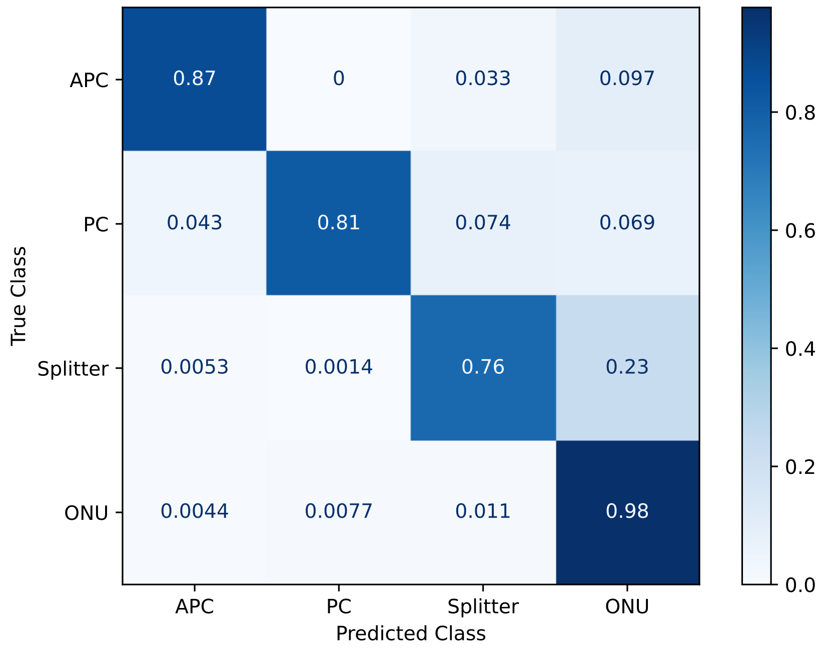

Figure 5.2: Confusion matrix after training with moving window approach

the objective of classifying ONUs correctly is met with this work of having an accuracy of 98%.

5.3 Performance on a Test Trace

For sample proof, a PON-setup is built and recorded with ADVA's ALM. These setup is built similarly to those found in the field. Different cables as well as a different splitter are used for this setup. The resulting trace can be seen in figures 5.3 and 5.4 whereas only relevant parts are plotted for reasons of clarity. Figure 5.3 contains the PC event at position $x = [0, 13]$ and APC event at position $x = [15, 24]$. Figure 5.4 also contains a PC event at $x = [15216, 15225]$, APC at $x = [15233, 15235]$, power splitter at $x = [15244, 15250]$. Four peaks representing ONUs can be seen in the interval $x = [15254, 15293]$. When talking about distance in terms of ONU, the absolute difference from two ONUs to the power splitter is meant. Because the first two ONUs ($x = [15254, 15268]$) are placed closely together with a distance of six meters, the events start overlaying. The small differences in distance to figure 5.5 are due to inaccuracies of the photo diode.

The trace was explicitly not used for training. The network was trained with the moving events data generation approach (section 4.4.2). The trace was loaded as described in section 4.5. The confusion matrix of this training can be seen in figure 5.2. The network is able to both find the correct positions as well as classifying all events marked in green in figure 5.5.

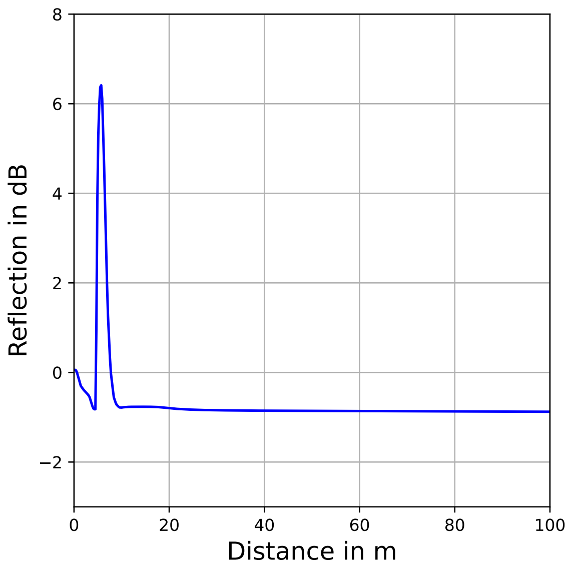

Figure 5.3: Test trace for $x = [0, 100]$

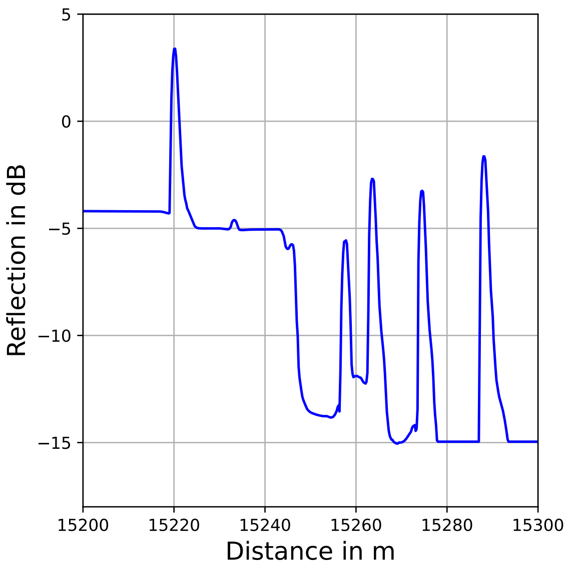

Figure 5.4: Test trace for $x = [15200, 15300]$

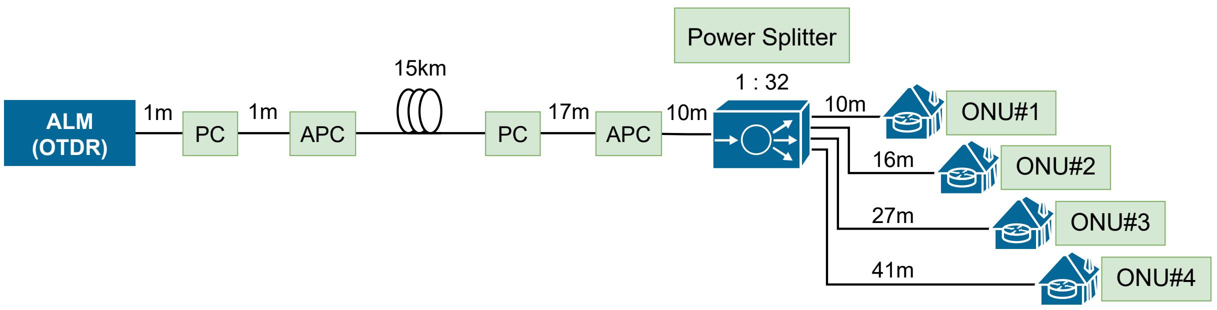

Figure 5.5: Test trace

5.4 Comparison to State-of-the-Art Techniques

OTDR-trace analysis is also used in other devices apart from *ADVA's ALM* (2016) e.g. in Ciscos "NCS-1001" mentioned in Rutigliano et al. (2021). Because of high company secret policies, it is hard to get those devices. However, in this project, ADVA's ALM is used as a reference because it is easily accessible. The ALM has a feature called Fingerprint which automatically detects events. This algorithm works similar to the proposed method in section 4.3. Additionally, a measurement distance has to be set while it is not required for proposed solution in this work. The use of the ALMs implemented algorithm, which works with the calculation of the slope, also means that there is no event classification. The ALM Fingerprint was able to find all events in the test trace (section 5.3). Comparing this to the results from this work, it is easy to see how the proposed classifier extends functionality. It is possible to not only find event positions but also to classify the event type. With this ability, the tool becomes more useful for areas such as fault detection.

6 Conclusion

This chapter summarizes the analysis results and the insights that can be derived from them. This is followed by an outlook showing weaknesses and further approaches.

6.1 Resume

The worked shows a new fast and efficient way to find and classify events happening in PONs by using deterministic event position proposal and deep learning. It outperforms and extends state-of-the-art devices and techniques by not only finding events but also classifying the corresponding event type. For this, a data generation algorithms for training CNNs is developed and compared. Furthermore, a tool for labeling traces and visualizing results was developed for this project. The accuracy of the CNN in classifying the correct event did not exceed 90% for all events, such as power splitters. However, most events could be reliably detected with an accuracy around 90%. Most importantly, ONUs were correctly classified in 98% of cases during training evaluation making it already useful for in field ONU fault detection. The performance on a test trace, presented in 5.3, also shows the effectiveness of the proposed technique in this work. Through extended optimization and more research, the effectiveness and accuracy of the network can increase.

6.2 Outlook

This work showed other approaches to deterministic algorithms and to Rutigliano et al. (2021) are possible. Through extended optimization and more research the effectiveness and accuracy of the network can increase. Especially the splitter classification must improve to guarantee reliability of the tool. It is worth mentioning that only a single convolutional neural network was tested for a proof of concept in this work. It is also possible to try other implementations of CNNs as well as other types of networks such as recurrent neural networks.

ADVA's ALM (2016) was used in this work. In future, the presented technique could be implemented to the ALM, helping telecom companies to increase fault detection capabilities.
In PONs, more events exist than are covered in this paper. The classifier could be extended to include events such as open physical connectors, fiber bendas or fiber cuts. This would increase the use of this tool in terms of fault detection.

When ONUs are connected with nearly the same distance to power splitters, these events start to overlay leading to higher peaks or a change in their characteristic look. For humans, this traces become hard to read and undistinguishable. A second network could be used to determine how many ONUs are overlaying on the same position. The proposed network in this work would give the position of classified ONUs to the second network which specifies the number of ONUs greater than zero. PONs include devices called optical line terminals (OLT) which safe information about connected ONUs. Having the number and position of ONUs detected by the networks, it can be compared to the previous mentioned OLT which would also the use of this tool in terms of fault detection.

List of Figures

2.1 Simplified Passive Optical Network 4
2.2 Simple three layered fully-connected neural network based on Fauvel et al. (2021) and O'Shea and Nash (2015) 5
2.3 Visual representation of a convolutional layer based on O'Shea and Nash (2015) 6

4.1 Laboratory setup for test trace (see figure 5.5) 12
4.2 Smart-ALM-UI with a labelled trace (Web-browser View) 13
4.3 Detection algorithm in Smart-ALM-UI 14
4.4 APC after 2dB attenuation 15
4.5 APC after 10dB attenuation 15
4.6 Laboratory setup for recording an APC trace 15
4.7 Encapsulated PC-event 17
4.8 PC event, line added 17
4.9 PC event, lines added, moved on x-axis 17
4.10 Basic architecture of traditional object detection algorithms based on Xiao et al. (2020) 17

5.1 Confusion matrix after training with artificial traces 22
5.2 Confusion matrix after training with moving window approach 23
5.3 Test trace for $x = [0, 100]$ 24
5.4 Test trace for $x = [15200, 15300]$ 24
5.5 Test trace 24

List of Algorithms

4.1 Find Event Positions 14
4.2 Create New Events by Scaling 16
4.3 Generate Training Data - Moving Events 18

Bibliography

ADVA's ALM (2016). URL: `https://www.adva.com/de-de/products/network-infrastructure-assurance/alm` (visited on 09/02/2022).

Babani, S, A. Bature, M. Faruk, and N. Dankadai (2014): "Comparative study between fiber optic and copper in communication link". In: *Int. J. Tech. Res. Appl* 2.2, pp. 59–63.

Barnoski, M. K. and S. M. Jensen (1976): "Fiber waveguides: a novel technique for investigating attenuation characteristics". In: *Appl. Opt.* 15.9, pp. 2112–2115. URL: `http://opg.optica.org/ao/abstract.cfm?URI=ao-15-9-2112`.

Berry, M. W., A. Mohamed, and B. W. Yap (2019): *Supervised and unsupervised learning for data science*. Springer.

Fauvel, K., T. Lin, V. Masson, Fromont, and A. Termier (2021): "XCM: An Explainable Convolutional Neural Network for Multivariate Time Series Classification". In: *Mathematics* 9.23. URL: `https://www.mdpi.com/2227-7390/9/23/3137`.

Fernández, M. P., L. A. B. Rossini, J. P. Pascual, and P. A. C. Caso (Oct. 2018): "Enhanced fault characterization by using a conventional OTDR and DSP techniques". In: *Opt. Express* 26.21, pp. 27127–27140. URL: `http://opg.optica.org/oe/abstract.cfm?URI=oe-26-21-27127`.

Ismail Fawaz, H., G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller (July 1, 2019): "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4, pp. 917–963. URL: `https://doi.org/10.1007/s10618-019-00619-1`.

Janiesch, C., P. Zschech, and K. Heinrich (Sept. 1, 2021): "Machine learning and deep learning". In: *Electronic Markets* 31.3, pp. 685–695. URL: `https://doi.org/10.1007/s12525-021-00475-2`.

Keiser, G. (2003): *Optical communications essentials*. McGraw-Hill Education.

Laferrière, J., M. Saget, and A. Cherncevère (1997): "Original method for analyzing multipaths networks by OTDR measurement". In: *Conference on Optical Fiber Communications*. Optica Publishing Group, TuT4. URL: `http://opg.optica.org/abstract.cfm?URI=OFC-1997-TuT4`.

Lam, C. (Jan. 2007): *Passive optical networks: Principles and practice*.

Nedelkoski, S., J. Cardoso, and O. Kao (2019): "Anomaly Detection and Classification using Distributed Tracing and Deep Learning". In: *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 241–250.

Oguiza, I. (2022): *tsai FCN*. URL: `https://timeseriesai.github.io/tsai/models.FCN.html` (visited on 09/08/2022).

O'Shea, K. and R. Nash (2015): "An Introduction to Convolutional Neural Networks". In: *CoRR* abs/1511.08458. arXiv: `1511.08458`. URL: `http://arxiv.org/abs/1511.08458`.

Rutigliano, D., G. Boracchi, P. Invernizzi, E. Sozio, C. Alippi, and S. Binetti (June 1, 2021): "Event-Detection Deep Neural Network for OTDR Trace Analysis". In: *Proceedings of the 22nd Engineering Applications of Neural Networks Conference*. Ed. by L. Iliadis, J. Macintyre, C. Jayne, and E. Pimenidis. Cham: Springer International Publishing, pp. 190–201.

snmp (1990): *RFC 1157*. URL: `https://datatracker.ietf.org/doc/rfc1157/` (visited on 09/02/2022).

Tan, C. W., A. Dempster, C. Bergmeir, and G. I. Webb (2021): "MultiRocket: Effective summary statistics for convolutional outputs in time series classification". In: *CoRR* abs/2102.00457. arXiv: `2102.00457`. URL: `https://arxiv.org/abs/2102.00457`.

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

Xiao, Y., Z. Tian, J. Yu, Y. Zhang, S. Liu, S. Du, and X. Lan (Sept. 1, 2020): "A review of object detection based on deep learning". In: *Multimedia Tools and Applications* 79.33, pp. 23729–23791. URL: `https://doi.org/10.1007/s11042-020-08976-6`.