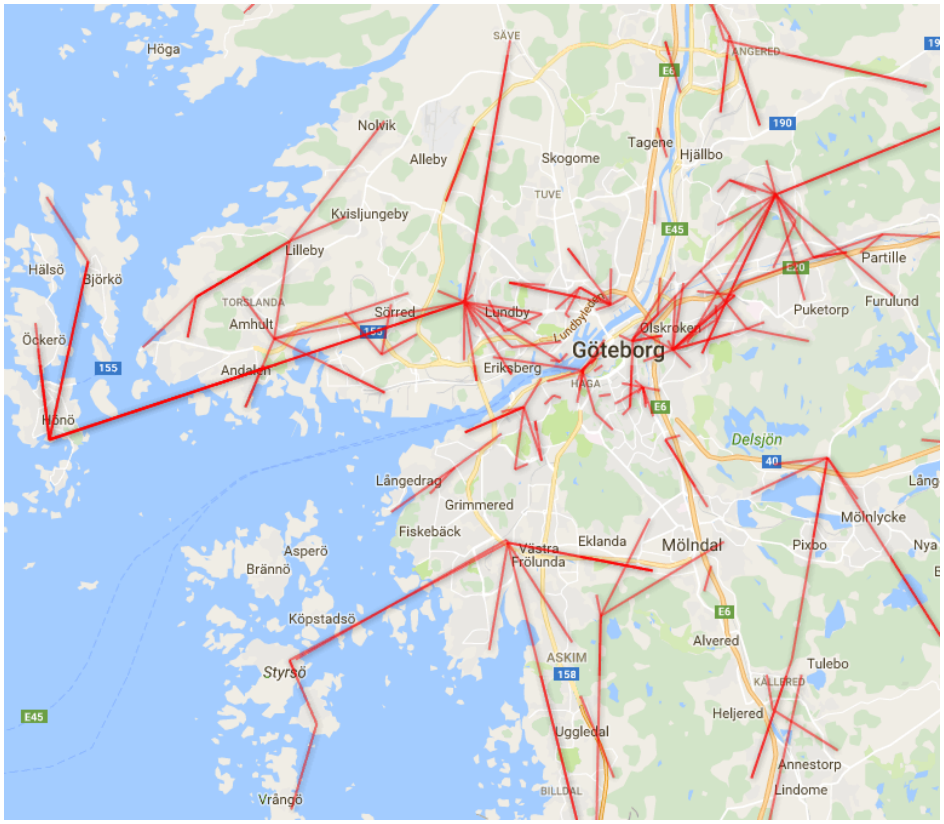
CHALMERS
UNIVERSITY OF TECHNOLOGY



Disturbance Detection and Classification in Large Microwave Networks

Time series classification using deep convolutional neural networks

Master's thesis in Computer Science

TOBIAS OLAUSSON
VICTOR SANDELL

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:105

Disturbance Detection and Classification in Large Microwave Networks

Time series classification using deep convolutional neural networks

Tobias Olausson
Victor Sandell

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communications and Antenna Systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Disturbance Detection and Classification in Large Microwave Networks
Time series classification using deep convolutional neural networks
Tobias Olausson & Victor Sandell

Supervisor: Jonas Hansryd, Ericsson
Supervisor: Alireza Sheikh, Department of Electrical Engineering, Chalmers University of Technology
Examiner: Alexandre Graell i Amat, Department of Electrical Engineering, Chalmers University of Technology

Master's Thesis 2017:105
Department of Electrical Engineering
Division of Communications and Antenna Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A map showing Ericsson's microwave network in Gothenburg.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2017

Disturbance Detection and Classification in Large Microwave Networks
Time series classification using deep convolutional neural networks
Tobias Olausson
Victor Sandell
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis explores how disturbances in a microwave network can be detected and classified using neural networks. The data was segmented into chunks consisting of one day measurements in one link. Each segment was then classified as normal behavior, weather disturbances, or disturbances caused by the construction cranes. Additionally, a general class for other disturbances was also used.
Two convolutional neural network structures were evaluated. One structure has a single link input, while the other uses the data of the nearby links. The networks were able to achieve an accuracy of 100% and 98%, respectively. This confirms that convolutional neural networks can be used to classify disturbances in a microwave network.

Keywords: Machine Learning, Neural Network, Classification, Time Series

Acknowledgements

First and foremost, we would like to thank our supervisor at Ericsson, Jonas Hansryd, for his guidance throughout the project. We would like to thank our supervisor at Chalmers, Alireza Sheikh, for his insight surrounding the thesis. We would also like to acknowledge Lei Bao and Sima Shahsavari for their support and expertise. A thanks goes out to Olof Mogren, from the Computing Science division at Chalmers, for his support regarding machine learning. We would like to thank the owners of the microwave network, Hi3G Access AB, for providing the data for this thesis. Last but not least, we would like to express our gratitude toward Ericsson and also everyone else involved with the project.

Tobias Olausson & Victor Sandell, Gothenburg, June 2017

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Background . . . 1
1.2 Purpose . . . 2
1.3 Limitations . . . 2
1.4 Related Work . . . 2
1.5 Outline . . . 3

2 Theory **5**
2.1 Microwave Transmission . . . 5
2.1.1 Quadrature Amplitude Modulation . . . 5
2.1.2 Attenuation . . . 6
2.2 Artificial Neural Networks . . . 6
2.2.1 Supervised Learning . . . 7
2.3 Convolutional Neural Networks . . . 7
2.3.1 Convolutional Layer . . . 8
2.3.2 Pooling Layer . . . 9
2.4 Preprocessing . . . 9
2.4.1 Time Series . . . 9
2.4.2 Normalization . . . 9
2.5 Training . . . 10
2.5.1 Backpropagation . . . 10
2.5.2 Learning Rate . . . 11
2.5.3 Cross Entropy . . . 11
2.5.4 Dropout . . . 11
2.6 Validation . . . 12

3 Methods **13**
3.1 Data . . . 13
3.1.1 Expected Behaviour . . . 13
3.1.2 Common Disturbances . . . 14
3.1.3 Normalization . . . 14
3.1.4 Labelling . . . 15
3.2 Machine Learning Approach . . . 15

3.2.1 TensorFlow 15
3.2.2 Neural Network Design 16
3.2.3 Parameter Settings 17

4 Results **19**
4.1 Classification Performance 19
4.1.1 Normalization 19
4.1.2 Network Comparison 19

5 Conclusion **25**

6 Future Work **27**
6.1 Worldwide Deployment 27
6.1.1 Scalability 27
6.1.2 Location Dependence 28
6.2 Network Modifications 28
6.2.1 Semi-supervised 28
6.2.2 Additional Data 29

List of Figures

2.1 The binary points of 16-QAM. 6
2.2 An example of a feed forward neural network. The circles represent neurons and the lines between the neurons are the weighted connections. This network consists of three input neurons, one hidden layer with four neurons and the output layer with three neurons. 7
2.3 A segment, defined by the filter size, is extracted and given as an input to a neuron in the feature map. Subsampling using pooling is then performed and followed by inputting the results of the feature maps to a fully connected layer. 8
2.4 An example of maximum pooling. 9

3.1 The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day. 13
3.2 An example of a weather disturbance. The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day. 14
3.3 An example of a crane disturbances. The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day. 15
3.4 The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day. 16

4.1 Comparison of zero-mean normalization (green line), with the zero-mean unit-variance normalization (blue line) for the input data of the single link structure. 20
4.2 Comparison of the zero-mean normalization (green line), with the zero-mean unit-variance normalization (turquoise line) for the nearby link structure. 20
4.3 The orange lines show the best obtained accuracy and loss when using the single link structure. The turquoise lines show the result of using the same structure, but with the addition of nearby links. 21
4.4 The neural network structures with the highest obtained accuracy and lowest loss. The turquoise line represents the structure using the nearby links, and the orange line represents the structure using a single line link as an input. 22

4.5 This figure shows the structure of a convolutional neural network using time series data of the main link and its nearby links as an input data. The neural network consists of three convolutional layers, followed by dropout, and then supplied to a fully connected layer which returns a classification. 23

List of Tables

4.1 The results obtained from the best convolutional neural network structures. 19
4.2 The parameters for the best performing neural network using a single link as an input. 21
4.3 The parameters for the best performing neural network by addition of the nearby links as an input. 22

Acronyms

1-NN 1-Nearest Neighbour
Adam Adaptive moment estimation
ANN Artifical Neural Network
API Application Programming Interface
BIDMC Beth Israel Deaconess Medical Center
CIFAR-10 Canadian Institute for Advanced Research 10
CNN Convolutinal Neural Network
DNN Deep Neural Network
GHz Gigahertz
MC-DCNN Multi-Channels Deep Convolution Neural Networks
MLP Multilayer Perceptron
MNIST Modified National Institute of Standards and Technology
PAMAP2 Physical Activity Monitoring Data
QAM Quadrature Amplitude Modulation
ReLU Rectified Linear Unit
SGD Stochastic Gradient Descent

1
Introduction

In recent years there has been a rapid increase in the amount of collected and stored data. The analysis and manipulation of big data is one of the important aspects of the ever growing computer industry. Companies collect large amounts of data from various services potentially, for further analyzing.
Machine learning algorithms provide solutions which are efficient at analyzing and finding hidden data patterns. Machine learning techniques such as neural networks is used for a large variety of tasks related to data analysis such as image and speech recognition. Developing and applying machine learning algorithms to automate and improve data analytics is beneficial for companies, as it unlocks the potential use of the unused data. The continuing increase in computing power combined with the progress in the field of machine learning extends these possibilities further.
There are currently (2016), approximately 4 million operating microwave links all over the world. These microwave links connect radio sites, and is used in addition to fiber and copper links [1]. The use of fiber is expected to increase, while copper is being phased out. Microwave links will still connect 65% of radio sites in 2021. Due to the size of the microwave networks located all over the world, large amounts of data is generated and machine learning solutions can potentially be beneficial to the network operators.

1.1 Background

Ericsson collects data from a network of microwave links located in Gothenburg. The data includes the transmitted and received power for each link. The attenuation, deviation from the expected received power, is currently used to approximate precipitation in Gothenburg. When the received power is lower than the expected power, it is expected that the power attenuation was due to the rain or snow.
While the changes in the received power are often caused by the weather, it is not always the case. Some of these disturbances are instead caused by construction cranes, trees, and other unknown disturbances. At this time, Ericsson does not have a way of knowing what is the cause of fluctuations and drops in the received power without manual analysis of the link's data.
In order to find any potential problems in the network, large amounts of data must be analyzed manually which is time-consuming.
Time series analysis extracts patterns in time series data, in order to obtain useful features. The data gathered by Ericsson falls into the field of time series analysis since the data for the received power is directly linked to a point in time, forming

time series data.
This thesis explores the application of machine learning algorithms on time series data in order to classify and separate anomalies from normal behaviour within a large microwave network.

1.2 Purpose

The primary aim of this thesis is to explore the respective benefits of different machine learning algorithms for anomaly detection and classification for a single-variable time series. Furthermore, the findings will be used to simplify and improve efficiency of fault detection and handling in a microwave network.

Our thesis will seek to answer the following questions:

- How can problems be detected in a microwave network using the collected data about the received power over an extended period of time?
- Which kind of input data and neural network structure produces the best classification results?

1.3 Limitations

The data used for this thesis is only gathered for Gothenburg. As such, the program will only be required to work for this limited area. While some thought may be given to the future scalability, it will not be a high priority.
Predicting faults may be even more useful than detecting them in real-time. However, due to the limitations in time and data quality, this would be out of the scope of this thesis.
The thesis will examine ways to differentiate between normal disturbances caused by precipitation and more significant disturbances. This thesis will be limited to identifying a few prominent categories among these significant disturbances.

1.4 Related Work

There are a wide range of research topics surrounding the applications of machine learning for time series analysis. The main machine learning approach that will be explored in this thesis is the convolutional neural networks (CNNs), an extension of neural networks. Classification tasks usually require knowledge about the data in order for feature extraction by human experts. CNNs have the advantage of extracting and learning features, thus not requiring human experts [2]. This feature extraction capability is useful for all kinds of data. Since the data used for this thesis likely contains patterns and behaviour only found in microwave data, the automatic feature extraction of CNNs is applicable.
CNNs have successfully been applied to time series data within a range of fields, including speech recognition and audio classification. The results obtained by [3]

show that the CNN, compared to a corresponding deep neural network (DNN), performed much better for the same speech recognition task.
Furthermore, CNNs have been proven to be successful in image classification tasks. To make use of these results when classifying time series, the data can be transformed and represented as an image in order to treat the problem as an image classification task [4].
Another widely used method for time series classification is 1-nearest neighbour (1-NN) with dynamic time warping as a similarity measurement. This approach has been found to produce similar results to a multi-layer perceptron (MLP) with fully connected layers. However, in comparison to the convolutional networks, they have been shown to achieve lower classification accuracy when tested on 44 different time series data sets [5].
Multi-Channels Deep Convolution Neural Networks (MC-DCNN) [6], divides a multivariate time series into univariate time series and performs feature learning on each time series separately using CNNs. The feature learning for each time series is then concatenated into a MLP, in order to carry out classification. MC-DCNN outperform both 1-NN and MLP used on health care data. The health care data sets used are Physical Activity Monitoring Data (PAMAP2) and the Beth Israel Deaconess Medical Center (BIDMC) data set [7], containing multi-variate time series data.

1.5 Outline

In section 2, the reader is introduced to the general concepts of microwave transmission. This chapter presents the theory behind artificial neural networks and more specifically convolutional neural networks. Section 2 also presents the concepts used for preprocessing, and finally how training and validation is performed in an artificial neural network. Section 3 introduces the methodology. This section begins with a description of the analyzed data, including classification and preprocessing. Section 3.2 presents the machine learning approach, the utilized machine learning library and how the network was set up. Section 4 presents the results of this thesis. The different network structures are compared with regards to accuracy, confidence, and time consumption. Section 5 includes a discussion and conclusion of the results. Finally, in section 6, the potential future work is discussed.

2
Theory

In this section the theory behind this thesis is presented, focusing on artificial neural networks and in particular convolutional neural networks. Additionally, the theory behind data preprocessing used for this thesis is introduced.

2.1 Microwave Transmission

Microwave transmission is used to transfer information between two points, and is used in communication systems such as cellular and telecommunications. Microwave frequencies range from 3 Gigahertz (GHz) to 300 GHz, which is equivalent to wavelengths between 10 and 0.1 centimeters [8]. Antennas are used to direct and transmit the microwave signal. A microwave network is constructed by placing antennas, forming point-to-point communication links. Careful placement of the antennas allow for several links to use the same frequency. This is due to the fact that the microwave beams are highly narrow and precisely oriented, thus not interfering with other beams. Microwave transmission is done using high frequency which in turn leads to have a high bandwidth. However, due to the high frequency transmission, the signals have difficulties passing through mountains and other terrains. Therefore, it is essential to place the antennas strategically in order to avoid obstacles.

2.1.1 Quadrature Amplitude Modulation

Quadrature amplitude modulation (QAM) is a method of encoding information in a microwave signal. Using QAM the microwave signal is composed of two underlying signals, cosin and sinusoidal functions, given in (2.1). The symbol ω denotes the angular velocity, t is the time, ϕ is the phase delay, Q, I, and A are the amplitudes. The resulting signal is a sinusoidal function with varying phase and amplitude based on the underlying values of Q and I. Either using the phase and amplitude of the final signal or by decoding the values of Q and I, one can infer the actual value of the signal as can be seen in Fig. 2.1.

$$Q * \cos(\omega t) + I * \sin(\omega t) = A * \sin(\omega t + \phi) \tag{2.1}$$

However, much higher modulations are possible. Since microwave signals are analog, they can encode an arbitrary number of values. This is, however, only true in a theoretical environment. In reality, there will always be noise in a signal, and as the modulation increases, the difference between signals become smaller. As such,

a higher order of QAM demands a higher signal-to-noise ratio. Nowadays, some microwave transmissions are based on the 2048-QAM.
The quality of microwave transmission varies significantly over time, especially when environmental factors such as rain is taken into account. Therefore, one can vary the modulation order to combat these factors and maintain the low error rate.

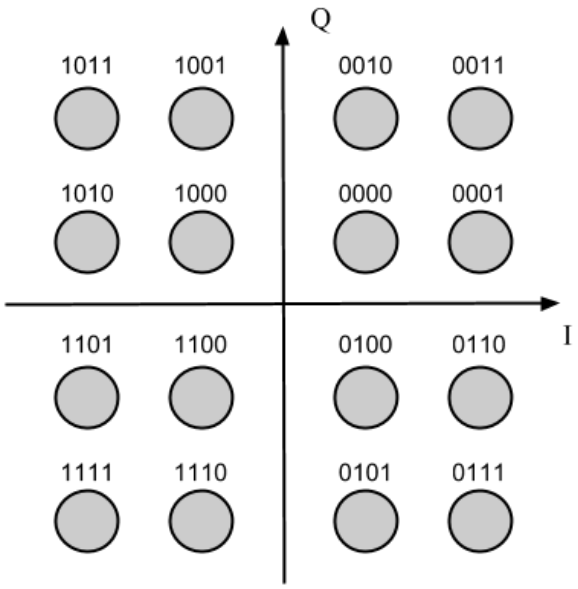


Figure 2.1: The binary points of 16-QAM.

2.1.2 Attenuation

Attenuation is the loss in power of a signal transmitted through a medium. Compared to lower radio frequencies, signals in the microwave range suffer more from attenuation caused by the rain. This becomes especially apparent at frequencies above 10 GHz [9]. Terrain, buildings and trees are obstacles that can cause attenuation, thus the path between the transmitting and receiving antennas must remain unobstructed.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs) is a prominent computational model in machine learning. ANNs are inspired by the network of neurons in a biological brain. An ANN consists of an input layer, an output layer and several hidden layers. The neurons are connected between the layers by weights that determine the output of each layer to the next. ANNs are trained instead of explicitly programmed, using training examples to achieve a desired output corresponding to the input by updating the weights between the neurons in the network. The method used to update the weights and biases between neurons is called backpropagation, which is explained

further in section 2.5.1. An ANN with a sufficient amount of neurons and layers can be a exceptional tool for feature detection and classification. An example of an ANN can be seen in 2.2.

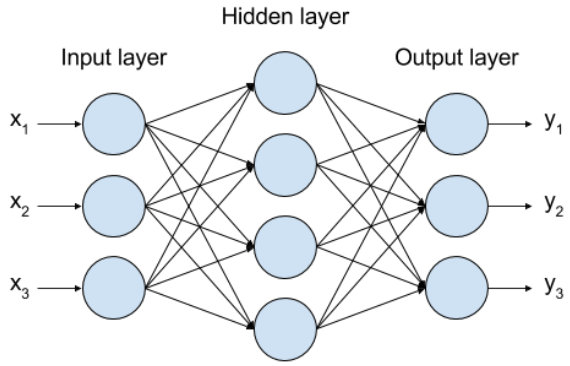


Figure 2.2: An example of a feed forward neural network. The circles represent neurons and the lines between the neurons are the weighted connections. This network consists of three input neurons, one hidden layer with four neurons and the output layer with three neurons.

There are different types of ANNs, e.g. feed forward neural networks and recurrent neural networks. Another type of ANN is the convolutional neural network, which will be described further in the next section.

2.2.1 Supervised Learning

Training a neural network requires selection of the input and output format. One of these training schemes is called supervised learning. Training examples with a corresponding label are required in order to train a neural network using a supervised approach. The goal is to achieve an inferred function from the training set, which produces accurate labels for unfamiliar data. This is the same as training the network to generalize.
Supervised learning can in turn be grouped into further categories, classification and regression. In the classification approach, each training example is assigned a class, and the neural network is trained to output an integer corresponding to a class. In a regression network, each training example is instead labeled with a real value, and the output is also a real value.

2.3 Convolutional Neural Networks

Conventional artificial neural networks have several downsides. Each of the neurons of a layer in a fully connected neural network, is connected to all of the neurons in the next layer, making the amount of neurons required for the large data infeasible. CNNs handle large input by reducing the spatial size of the input. CNNs have performed exceptionally well in complex classification tasks, particularly in image classification [10, 11]. This section will describe the most commonly used components of a CNN, convolutional layers and pooling layers.

2.3.1 Convolutional Layer

The primary component in a CNN is the convolutional layer. This layer consists of a set of filters. The filters in each layer is moved across the input in order to produce an activation map. This process is known as the convolutional step in the network. CNNs utilize sparse connectivity, meaning that only a portion of the input is given as an input to a neuron, as seen in Fig. 2.3. The spatial connectivity between the filter and the neuron is known as the receptive field. The stride of the filters decides how much a filter is moved in each dimension across the data. A stride of 1 results in a filter moving one data point in each step. The stride determines the dimensions of the output, since a smaller filter size leads to less overlapping of the receptive fields. Sometimes it is beneficial to obtain the same input and output dimensions, which can be achieved using padding. Padding is done by adding data, commonly zeroes, to the edges of the input. In order to adjust the amount of parameters, a filter in a convolutional layer uses the same weights and biases across the entire input space. The weight sharing makes the filters in the convolutional layer behave as feature maps. These feature maps achieve spatial translation. Since features can be located everywhere on the input, weight sharing is done. If spatial translation was to be removed, the training set would have to be significantly larger and generalization to new data would be worse.

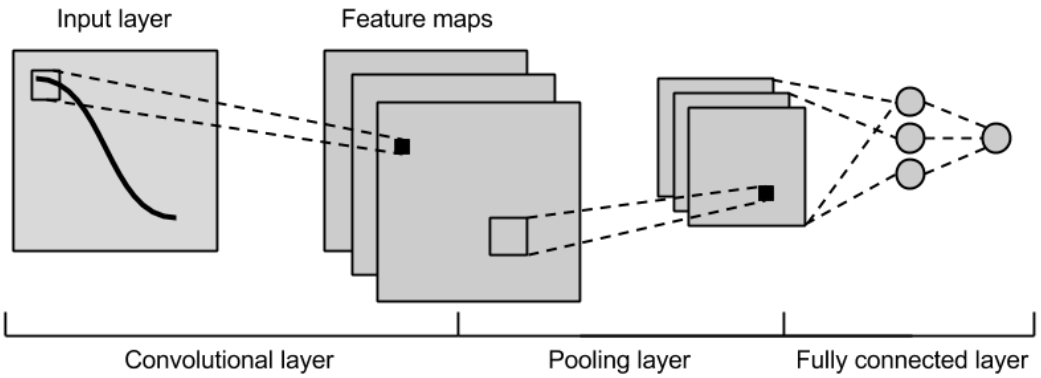


Figure 2.3: A segment, defined by the filter size, is extracted and given as an input to a neuron in the feature map. Subsampling using pooling is then performed and followed by inputting the results of the feature maps to a fully connected layer.

After the convolution has been performed, an activation function is applied. Rectified linear unit (ReLU) (2.2) is one of the most commonly used activation functions. Compared to other activation functions, such as the hyperbolic tangent or the sigmoid function, ReLU achieves shorter training time [10].

$$f(x) = \max(0, x) \tag{2.2}$$

2.3.2 Pooling Layer

The next major component in a convolutional network is the pooling layer. Pooling is performed in order to reduce the overall complexity, and is usually applied after a convolutional layer. In order to perform pooling, a filter is moved across the data, downsampling over each filter by discarding data. One of the most common pooling methods is max pooling. In each region, the maximum value is extracted, as shown in Fig. 2.4. Although pooling reduces the complexity of the data, [12] suggests that simply applying pooling after each convolutional layer reduces performance due to the spatial reduction. In [13], it is proposed to remove the pooling layer in favor of either increasing the stride of the convolutional layers, or replacing the pooling layer with a convolutional layer where the stride is larger than one.

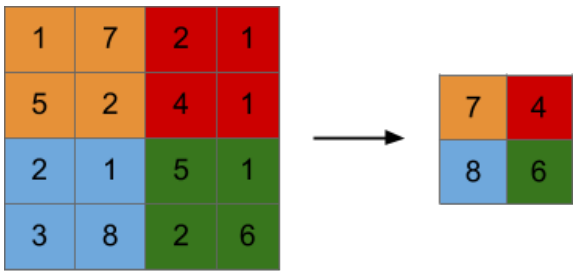


Figure 2.4: An example of maximum pooling.

2.4 Preprocessing

Data used for machine learning purposes can be noisy and not always entirely reliable. In order to achieve valuable results, some degree of data preprocessing is usually needed. The data used for this thesis is time series data.

2.4.1 Time Series

A time series is a set of data points, each indexed at a point in time. The data points in a time series are collected at a fixed interval, where the frequency in which the data points are collected determines the resolution of the time series. Time series data can be represented as a function of time, $f(t)$, where t is a point in time and $f(t)$ is the corresponding value.

2.4.2 Normalization

Normalization of the input data for the ANNs has been shown to make the network perform better generalization and achieve faster network training convergence [14]. Normalizing the input from one layer to another within the network during training enables the use of higher learning rates, and has the potential to eliminate the need for dropout [15]. There are different methods used for normalization, one being zero-mean and unit-variance, shown in (2.3). The normalized data is calculated by subtracting the mean and dividing by the standard deviation.

$$x' = \frac{x - \bar{x}}{\sigma} \tag{2.3}$$

2.5 Training

In terms of neural networks, training is the process of iteratively updating the weights in the network to minimize the cost function. Common cost functions include 0-1 loss, mean squared error and cross-entropy, but the cost function can be defined in many different ways. During training of a neural network, the task is to fit the network to a training set. One of the problems which can arise while training a neural network is overfitting. Overfitting commonly occurs when the parameter size exceeds the number of training examples, or due to excessive training. This can result in the neural network learning features which are not found outside of the training data. Overfitting leads to noise in the data becoming more prominent while training, and the underlying pattern which the neural network should learn is instead obscured by the noise. Various methods exist to avoid overfitting, one example of such a method is dropout, which is described further in section 2.5.4.

2.5.1 Backpropagation

A loss function is used to determine how well a model is performing. In the case of an ANN, the loss function is a value of how accurate the predictions are. In backpropagation the derivatives of the network weights are calculated with respect to the loss function. Based on the derivatives, the weights are then updated according to the chosen optimization method.

Stochastic Optimization

Gradient descent is a method used in conjunction with backpropagation to minimize an objective function, usually also referred to as the loss function, cost function or error function. Minimization is done stepping in the steepest direction using the derivative of the objective function, updated by iteration over each value in the data set. The new gradient is calculated according to (2.4). The goal is to find the value of θ which minimizes the objective function $J(\theta)$. η represents the step size, determining how much each iteration should affect the value of θ.

$$\theta := \theta - \eta \frac{d}{d\theta} J(\theta). \tag{2.4}$$

Stochastic gradient descent (SGD) is an extension of gradient descent which successfully handles large data sets [16]. In order to handle large data sets, the training data is randomly arranged, and each training sample is used to update the value of θ. (2.5) shows the update step when using SGD. Training item x and corresponding label y is a randomly chosen pair from the training set. Computing the gradient based on a high number of data points can lead to the incorrect steps, resulting in a higher value of $J(\theta)$. However, setting a low value for the step size achieves convergence by computing the gradient a large number of times. The fluctuations

of the SGD algorithm enables it to potentially jump and find a better local minima. SGD in combination with backpropagation is the standard algorithm used for the ANNs.

$$\theta := \theta - \eta \frac{d}{d\theta} J(\theta; x(i), y(i)). \tag{2.5}$$

Adam

Adaptive moment estimation (Adam), is an adaptation of stochastic gradient descent [17]. The previous updates contain information about the direction of the minimum. Adam utilizes the previous updates, adding momentum when calculating the next update. The direction of a minimum builds momentum, increasing the rate in which a local minima is reached. Adding a momentum component prevents heavy fluctuations due to constantly updating the direction. Another feature of Adam is the adaptive parameter updates. Parameters benefit differently for a set learning rate. Therefore, using separate learning rates for each individual parameter, the learning process becomes faster. Compared to the other optimization methods, Adam performs well in practice due to being computationally efficient and having low memory requirements.

2.5.2 Learning Rate

The learning rate of a network is a measure of how aggressively it adapts according to the value of the loss function. A higher learning rate means that the network is quicker to adapt, but less capable of fine-tuning. If a network has too low learning rate, it will take longer to converge and increases the risk of converging to a local optima.
Introducing a decay rate, causing the learning rate to be gradually lowered, can potentially allow the network to converge quickly and be tuned finely in the later stages of training.

2.5.3 Cross Entropy

The squared loss function can be used in cases where a large deviation from the target should be punished more heavily than several smaller ones. This is due to the fact that a very large deviation will cause the learning rate to slow down. The cross-entropy cost function targets this weakness by making the learning rate be controlled by the error in the output [18].

2.5.4 Dropout

Dropout is the process of randomly discarding a portion of the hidden and input units. This is done to punish the network for over-reliance on a few units. In each training step, a neuron and its connections are dropped with a probability of $1 - p$ and retained with probability of p. After training, the resulting neural network can be seen as a combination of several smaller neural networks with the shared

weights. The resulting neural network has less risk of overfitting and better ability to generalize [19].

2.6 Validation

Even with steps taken to avoid overfitting, there are no guarantees that it will not happen. Therefore, the accuracy and loss metrics from training can be misleading. It is therefore common to divide the data into training and validation sets. The validation set is not fed to the network during the training phase, but tested on after to give more reliable performance metrics. Feeding the validation set to the network outputs performance metrics about accuracy and loss.

3
Methods

This chapter presents the methodology used in this thesis. The classifications used and the structure for the neural network is described.

3.1 Data

The data used in the thesis consists of the received power from around 700 links in the area surrounding Gothenburg. The data has been logged every 10 seconds, beginning in 2015-04-28. The data was split into 24 hour segments, consisting of 8640 samples.

3.1.1 Expected Behaviour

Most of the time the received power is expected to remain level. Variations tend to lie within 2dB of the mean, however exceptions to this exist for links in certain environments such as over water. An example of a link with received power in the expected range can be seen in Fig. 3.1.

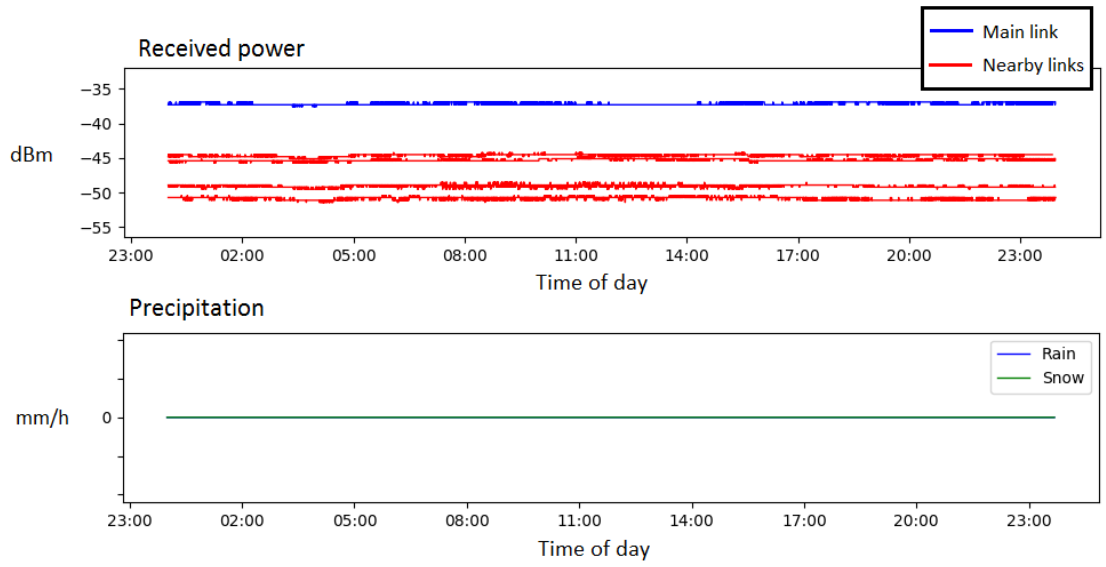


Figure 3.1: The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day.

The received power is impacted by environmental effects such as rain and snow. While this can cause significant drops in performance, the effects can only be mitigated, not avoided completely.
Regardless of whether it is raining or not, the links are expected to behave similarly to those nearby. Fig. 3.2 shows an example of a link affected by rain. The rain occurred at the same time as the drop in the received power, and the nearby links showed the same pattern.

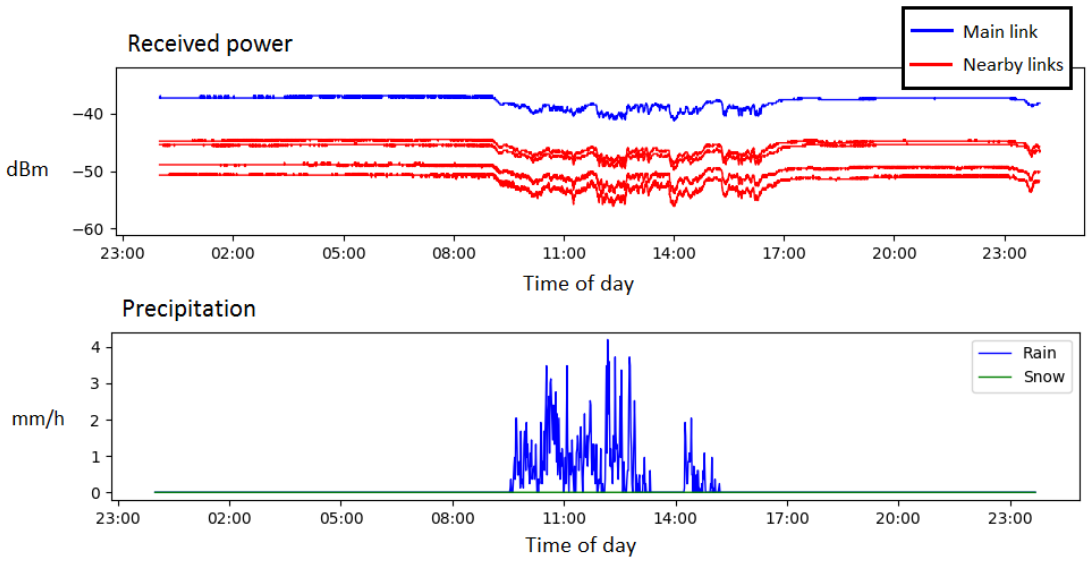


Figure 3.2: An example of a weather disturbance. The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day.

3.1.2 Common Disturbances

In urban environments, such as Gothenburg, construction sites and in particular the cranes used in the construction can interfere with the transmissions. These disturbances cause a very distinct pattern in the received power as the cranes normally only move during the standard working hours. In fact, during the working hours the received power can fluctuate heavily in very short time spans. Fig. 3.3 shows a link with a crane disturbance.

3.1.3 Normalization

The mean received power can vary greatly, but the deviations from the mean appear similarly for the same situations. As such, normalizing the data to a zero-mean makes a lot of sense. It is common to also normalize to unit-variance, however it leads to the information loss about the severity of the power fluctuations.

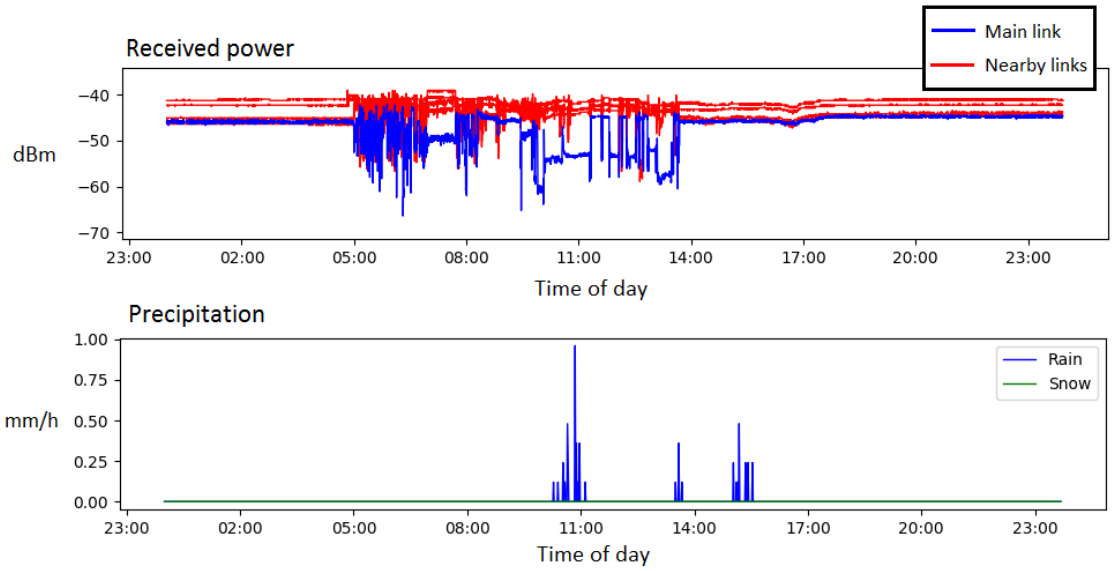


Figure 3.3: An example of a crane disturbances. The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day.

3.1.4 Labelling

To use a supervised approach, the data should be labeled. While automatic labelling to some extent is possible, high number of the underlying parameters make it complex and unreliable. Therefore, human evaluation was used, where an expert user was given rain and snow data and received power measurements for both the link in question and links near it. The data was then classified into one of 4 different categories, normal behaviour, weather disturbance, crane disturbance and unknown disturbance. Fig. 3.4 shows the received power for a main link and the 4 closest links. Rain does not seem to affect the main link similar to the nearby links, and the pattern is very distinct, thus this example was labelled as an unknown disturbance.

3.2 Machine Learning Approach

This section describes the machine learning library that was used to construct the neural network. The design of the neural network is presented, including the various parameters used.

3.2.1 TensorFlow

The neural networks was designed using TensorFlow. TensorFlow is an open-source interface which allows for implementation and experimentation of machine learning algorithms [20]. The usage of both Central Processing Units (CPU) and Graphics Processing Units (GPU) is supported by TensorFlow, which is a necessity when handling large data sets. In addition to the TensorFlow interface, the high level Ap-

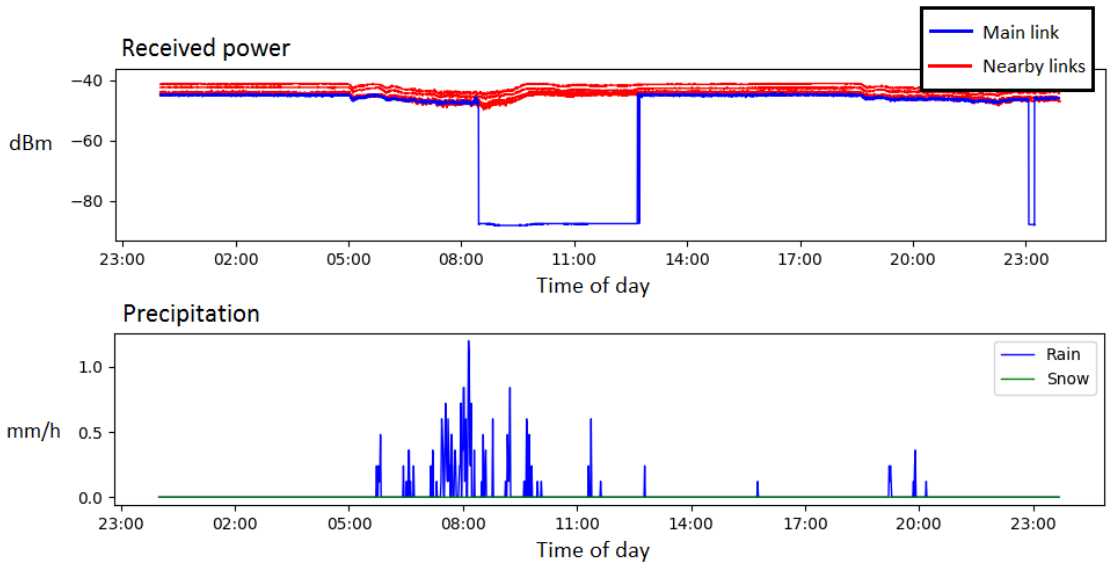


Figure 3.4: The blue line in the top graph depicts the data for the main link over one day, and the red lines are the nearby links. The bottom graph shows the rain for the same day.

plication Programming Interface (API) called TFLearn was used, which builds upon the basic functionality of TensorFlow and allows for more agile experimentation.

3.2.2 Neural Network Design

Selecting a model which perform well, both in classification performance and time consumption, was crucial. A convolutional neural network (CNN) approach was chosen, since CNNs perform well in classification tasks with large input. The input size was structured to provide sufficient data, in order to improve feature extraction. The initial approach was a neural network structure which used a single link as the input. An advantage of this structure is the time consumption compared to a structure using a larger input size.
When constructing the neural network, several data characteristics were taken into consideration. Since a link is expected to behave similarly to its nearby links, having the nearby links as the input data could potentially lead to better feature extraction. However, due to the input size of this neural network structure, the potential increase in classification performance comes at the cost of time consumption.
The depth and the number of hidden layers in the neural network, affect the performance. Several hidden layers provide additional convolution, which in turn leads to improve feature extraction and classification performance. However, a neural network can be unnecessarily large, leading to increased time consumption. Between one and five convolutional layers were tested.
There does not exist one correct architecture for convolutional neural networks in general. However, there are existing widely tested architectures for some specific data sets. While these architectures might be applicable to similar data sets, the data used for this thesis likely contains behaviour only found in microwave data. Thus,

an architecture that achieves good results for microwave data had to be created. Several combinations of parameters were tested by trial and error. A high number of tests were performed with combinations of the parameters listed in the next section. Apart from the trial and error approach, existing CNN architectures were taken into consideration.

3.2.3 Parameter Settings

In order to achieve a satisfactory result, a range of parameters for the neural network had to be considered. Several neural network structures with different combinations of learning rate, filters and pooling layers were tested to find out how to achieve the highest accuracy.

Learning Rate

The learning rate was set to 0.005 with an exponential decay of 0.99 for the first moment estimates and 0.999 for the second moment estimates.

Filters

A higher amount of filters increases training time, but does not necessarily improve the accuracy of the neural network. An arbitrary number of filters was chosen for each layer, ranging between 20 and 50 filters. Different filter sizes were used for each layer. Too small filter sizes can lead to an excessive amount of parameters within the network, and possibly also overfitting. Filter sizes between 5 and 100 were chosen as a baseline.

Pooling

Pooling can be useful due to reducing the complexity of the input data between layers. However, this is not always the case since pooling has a negative effect on classification performance due to the spatial reduction. Different neural network structures were tested, with and without pooling, in order to find out whether pooling improved the accuracy in conjunction with other parameter values.

4
Results

This chapter presents the results from the convolutional neural network structures from chapter 3. A comparison of neural network structures with regards to accuracy and loss, is presented.

4.1 Classification Performance

The results were gathered by running the neural networks for a set amount of epochs. The single link structure typically converged rather fast, and was thereby only run for 100-250 epochs. Training the structure using nearby links was more time consuming due to a larger input size and slower convergence. We found that adjusting the learning rate for different network structures did not seem to have a significant effect on the classification. The best obtained results can be seen in Table 4.1.

Results		
CNN structure	Validation Accuracy (%)	Validation Loss
Single link	100	0.02
Nearby link with single link parameters	98.2	0.1
Nearby link	98.2	0.07

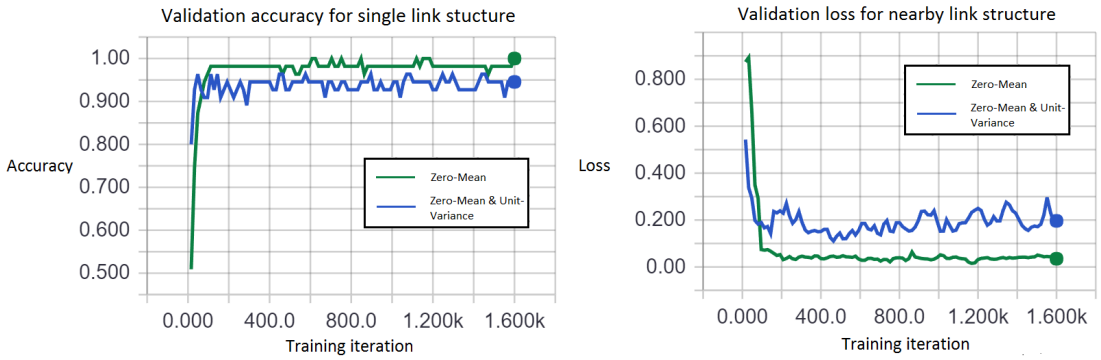
Table 4.1: The results obtained from the best convolutional neural network structures.

4.1.1 Normalization

In Fig. 4.1, the comparison between the results obtained by normalizing the data to the zero-mean and unit-variance data and normalizing to only zero-mean data is shown. As can be seen, the results based on variance normalization yielded both a higher loss and lower accuracy. Similar results were obtained when comparing the same normalization methods in the neural network structure by addition of the nearby links as an input, in Fig. 4.2.

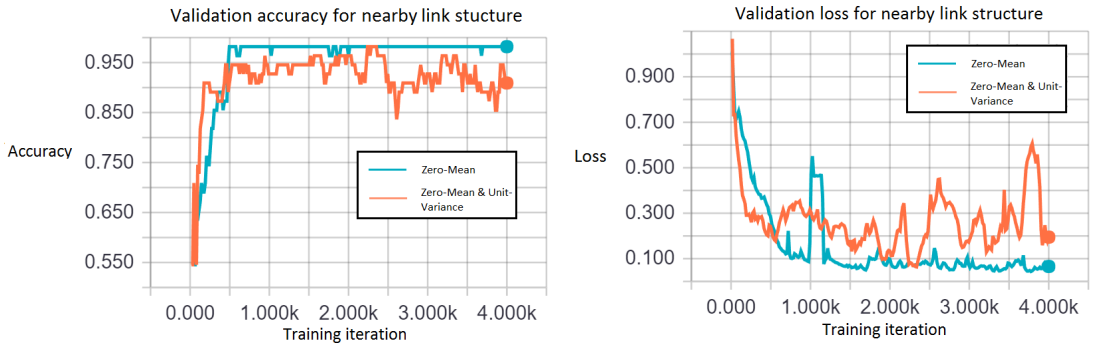
4.1.2 Network Comparison

Two neural network structures were considered in this thesis. The first one is the convolutional networks with a single time series for a link as an input, while the



(a) Validation accuracy **(b)** Validation loss

Figure 4.1: Comparison of zero-mean normalization (green line), with the zero-mean unit-variance normalization (blue line) for the input data of the single link structure.



(a) Validation accuracy **(b)** Validation loss

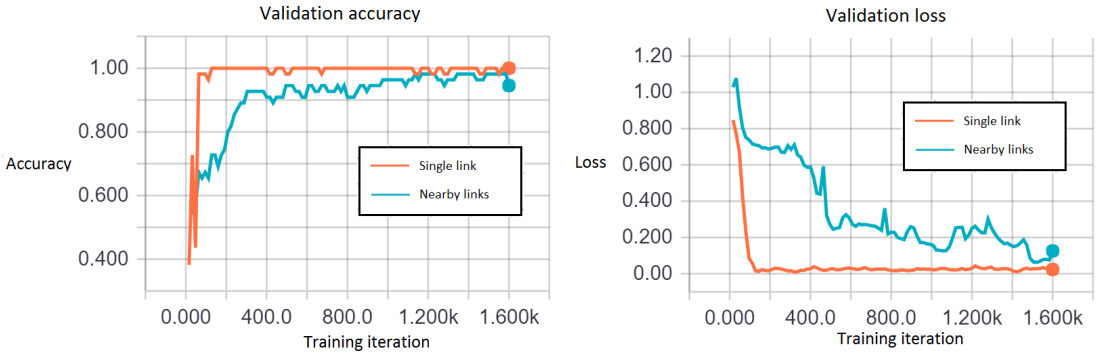
Figure 4.2: Comparison of the zero-mean normalization (green line), with the zero-mean unit-variance normalization (turquoise line) for the nearby link structure.

second one used a main link's time series data combined with the four nearest links' time series data, requiring a two-dimensional convolutional neural network. The structure of the neural network using nearby links can be seen in Fig. 4.5.
The networks consisted of three sequential convolutional layers. All convolutional layers used a ReLU activation function. After these layers, dropout was performed and ultimately there was a fully connected layer with four outputs, corresponding to the four different classes. Cross entropy was the loss function used for each of the neural network structures. We remark that Adam, see section 2.5.1, was selected as the optimizer.
The best result produced by the neural network with a single link as an input, was obtained with the parameters listed in Table 4.2. Pooling layers were not utilized in conjunction with the listed parameters, since it resulted in higher loss and lower accuracy. We also found that training without pooling did not have a significant impact on the time consumption.
The results obtained by adding nearby links as an input, compared to feeding the

Single Link Structure			
	Number of filters	Filter size	Stride
Convolutional layer 1	20	5	5
Convolutional layer 2	20	25	5
Convolutional layer 3	20	25	5

Table 4.2: The parameters for the best performing neural network using a single link as an input.

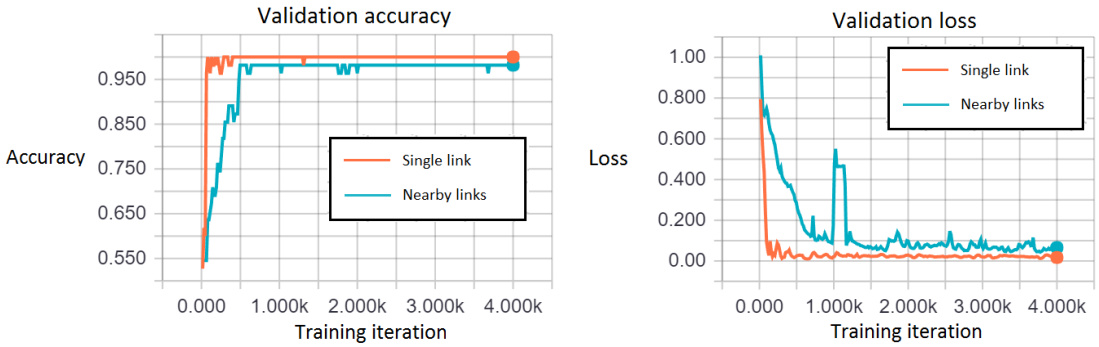
main link can be seen in Fig. 4.3. The structure for the network without nearby links was the best one obtained through our experimentation. Using the same network structure, the network using nearby nodes takes significantly longer time to converge and results in a slightly lower accuracy and loss.



(a) Validation accuracy **(b)** Validation loss

Figure 4.3: The orange lines show the best obtained accuracy and loss when using the single link structure. The turquoise lines show the result of using the same structure, but with the addition of nearby links.

However, the network using nearby nodes was able to perform significantly better using a different structure and setup parameters. The comparison of the two approaches using the best tested structure can be seen in Fig. 4.4. This shows a much more similar performance to the neural network using a single time series as an input.
Even with the improved structure, the network using nearby links takes longer time to converge. Furthermore, the increased input size causes training to be significantly slower.
The best classification accuracy and loss for the network structure using the nearby links was achieved when applying larger filter sizes, compared to the structure using a single link. Pooling was also used in the neural network structure with the nearby links, as it showed improvements in both time consumption and classification performance. Furthermore, small filter strides for each convolutional layer produced the best result. The parameters used can be seen in Table 4.3.



(a) Validation accuracy **(b)** Validation loss

Figure 4.4: The neural network structures with the highest obtained accuracy and lowest loss. The turquoise line represents the structure using the nearby links, and the orange line represents the structure using a single line link as an input.

Nearby Links Structure			
	Number of filters	Filter size	Stride
Convolutional layer 1	20	100	5
Convolutional layer 2	20	100	5
Convolutional layer 3	20	100	5

Table 4.3: The parameters for the best performing neural network by addition of the nearby links as an input.

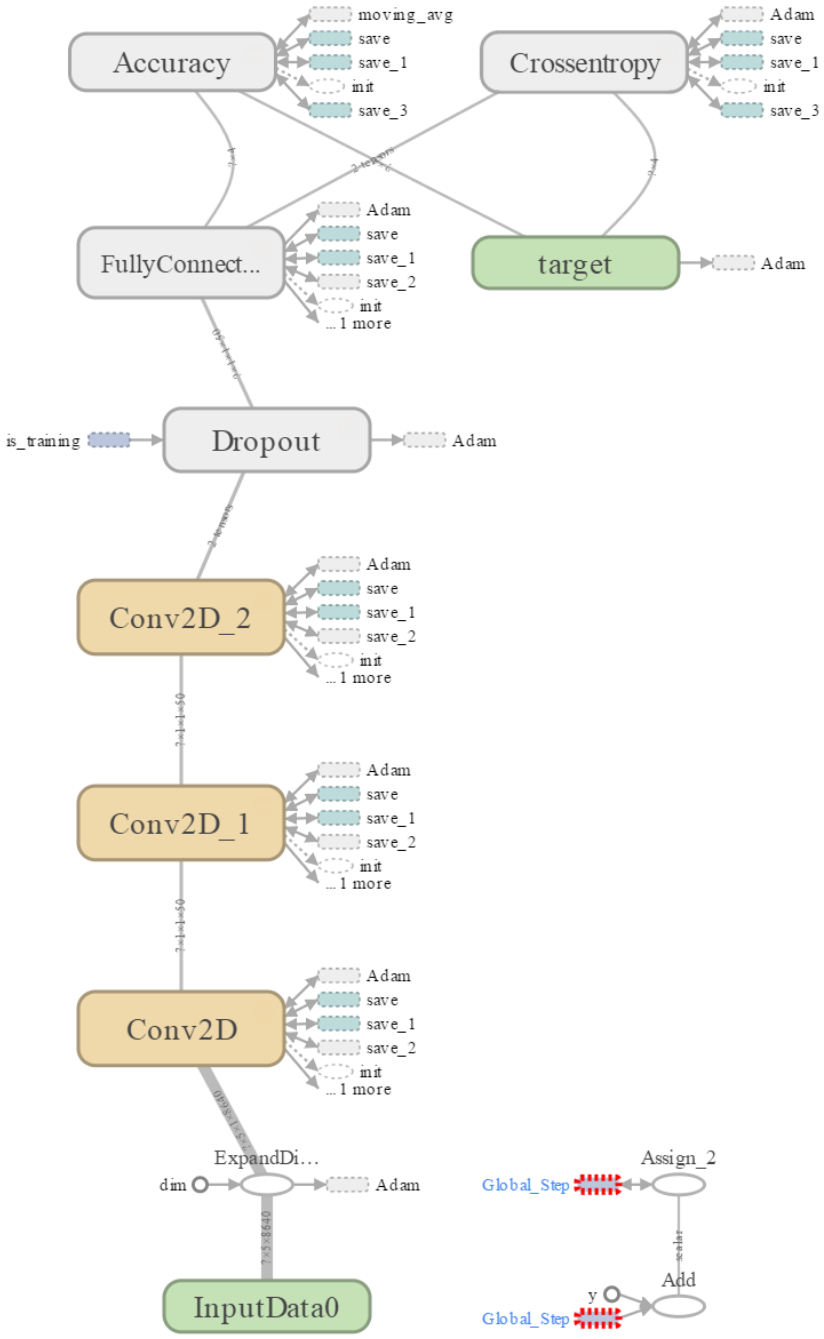


Figure 4.5: This figure shows the structure of a convolutional neural network using time series data of the main link and its nearby links as an input data. The neural network consists of three convolutional layers, followed by dropout, and then supplied to a fully connected layer which returns a classification.

5
Conclusion

Immense amounts of data is gathered by companies such as Ericsson. This data might be useful for other purposes, potentially providing a monetary benefit. However, due to the sheer size of the available data, efficient and precise methods have to be devised. Machine learning provides the capability to manage, and find patterns in large amounts of data.
In this thesis, we have evaluated large amounts of data which Ericsson has gathered from a microwave network in Gothenburg. A model, using a convolutional neural network, which detects and classifies disturbances in a microwave network was developed.
The neural network structure, achieving the highest classification performance was trained on 24-hour segments of the received power, each segment containing the data for one link. This neural network achieved a very high classification performance. A second structure was developed, which was trained based on the data of a main link and the data of the nearby links, for the same 24-hour segment. Although this neural network did not perform as we had hoped for our classifications, it could potentially perform better generalization for other disturbances.
The obtained results are promising, and it is apparent that machine learning can be used to detect and classify disturbances in a microwave network.

6
Future Work

In this chapter, possible future work is presented. Additional implementation and improvements for this project are discussed.

6.1 Worldwide Deployment

Since the current results are promising, there is a possibility of deploying the program across the rest of world. With this, there will come many new possibilities and challenges.

6.1.1 Scalability

One effect of including more areas will be a vast increase in the amount of the stored data. Therefore, scalability is an important issue.

Training Time

For every classified data sample, there will be one iteration required for each epoch in the training. As such, the training time for the network scales linearly with the amount of training data. This is only true excluding hardware limitations, however, and for example some changes may need to be made with respect to what data is held in random-access memory (RAM).
The amount of labeled data required, however, will not be expected to increase linearly with the number of microwave links. Urban areas similar to Gothenburg will require few or no additional labels. Even a rural environment in a similar climate is expected to at least share the behaviour during precipitation. Overall, the training time would be expected to scale significantly better than linear scaling with respect to the number of microwave links, assuming no other types of labels are needed.

Classification Time

While training only need to be done when a significant amount of new labels have been added, classification need to be done every day for continuous updates. Therefore, the efficiency of the updates is an important issue. Each segment consists of one days measurements for one link, and each of these segments will have to be classified independently of the rest. Therefore, the time required for classifying the segments for one day can, at best, scale linearly with the number of links.

6.1.2 Location Dependence

The data used was only from Gothenburg. Since this is a homogeneous urban area, it might not be possible to simply deploy the same solution across the world.
Other regions may have new types of disturbances that are not present elsewhere. One might, for example, have to take sand storms into account in desert areas, in which another classification is required to be added to the problem. In the areas where sand storms are not present, however, this means a potential classification which can only be wrong, never correct, has been added. This could reduce the classification performance.
Another possible scenario is that the same type of disturbance could have different characteristics in different environments.
A solution might be decentralizing the classifications. This means that we split the world into similar regions and consider independent classifications for each of them. This would require more classified examples but could potentially improve the performance.
Another approach is to include metadata to the neural network, i.e., include data about the type of the environment, the link is located in.

6.2 Network Modifications

Performing an exhaustive search for all possible network structures and parameters is not possible, so there is no guarantee that the currently highest performing structure is optimal. Tweaking parameters may allow for slight performance improvement but changing the type of neural network could unlock entirely new possibilities.

6.2.1 Semi-supervised

One can change the approach to the semi-supervised one. This allows the network to make use of the unlabelled data, in addition to what has already been labelled.

Self-classifying

One of the simplest semi-supervised approaches is to let the network be self classified. This starts in the usual way by training the network and classifying all data points. However, afterwards, the most confident classifications are converted into real labels. The process is then repeated many times, each time giving the network more labelled data to train on.
It has been shown that self-classifying can produce better performance [21], but these results may not be applied to our algorithm and dataset.

Ladder Networks

The semi-supervised ladder network structure has shown a lot of promise. It has achieved state of the art performance for classification in both semi-supervised Modified National Institute of Standards and Technology (MNIST) and Canadian Insti-

tute for Advanced Research 10 (CIFAR-10) [22], two datasets for object recognition in images.
Any conventional neural network can be converted to a ladder network. The initial structure is considered the clean encoder path. Another path, a corrupted encoder, is then added in which also adds noise to the signal in every layer. A denoising function is then used to reconstruct the original output of the activation function. The difference between the reconstructed output and the clean one is the denoising cost for each layer.
The supervised cost is based on the difference in output between the corrupted encoder and the actual target. The unsupervised cost is simply the sum of the denoising costs for each layer. For the supervised data the final cost is the sum of the unsupervised and supervised costs.
The ladder network can also be trained on unlabeled data, using only the unsupervised cost. While the ability to incorporate unlabeled data is a key feature, the ladder network architecture achieves good performance even in a fully supervised scenario [22, 23].

6.2.2 Additional Data

In the neural network structures that were evaluated in this thesis, only data of the received power was used as an input. The dataset that was used contained both transmitted and received power, but it is possible to store additional variables with configuration of the data collection. In [6], an approach which uses multi-variate time series is proposed. Such a method could potentially be used, since additional variables create new time series. This variables from the microwave network might provide useful features for the neural network. Furthermore, metadata can possibly be given as an input to a separate fully connected layer in the neural network. Metadata such as transmission frequency or the hop length of the links, could provide additional information concerning the behaviour of a link.
The number of classes used for the neural networks were limited to four, but it is possible to add additional categories. Overwater links are located across a pool of water such as a lake or the ocean. These links show a pattern of high fluctuations due to surface reflection and movement of the water. Multipath, which is due to receiving the transmitted signal from several paths, causing interference, could be possibly added. Additionally, during maintenance, a link's received power drops significantly, showing a clear pattern since maintenance is usually scheduled for evenings and nights. It would also potentially be possible for maintenance workers to report a link which is undergoing maintenance, in order to be classified and used as the training data. These are a few examples of behaviours in the microwave network which can be categorized and classified.

Bibliography

[1] Ericsson, "Ericsson microwave outlook, trends and needs in the microwave industry," [Online] Available: https://www.ericsson.com/assets/local/microwave-outlook/documents/ericsson-microwave-outlook-report-2016.pdf, 2016.

[2] M. Längkvist, *Modeling time-series with deep networks*. PhD thesis, Örebro university, 2014.

[3] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, pp. 1533–1545, Jul. 2014.

[4] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Proc. Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Jan. 2015.

[5] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *CoRR*, vol. abs/1611.06455, Dec. 2016.

[6] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *International Conference on Web-Age Information Management*, pp. 298–310, Springer, Jun. 2014.

[7] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[8] D. M. Pozar, *Microwave Engineering*, vol. 4. Wiley, April 2012.

[9] S. Das, A. Maitra, and A. K. Shukla, "Rain attenuation modeling in the 10-100 ghz frequency using drop size distributions for different climatic zones in tropical india," *Progress In Electromagnetics Research B*, vol. 25, pp. 211–224, Sep. 2010.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., Dec. 2012.

[11] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 3642–3649, Feb. 2012.

[12] B. Graham, "Fractional max-pooling," *CoRR*, vol. abs/1412.6071, Dec. 2014.

[13] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, Mar. 2014.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[15] S. Wiesler, A. Richard, R. Schluter, and H. Ney, "Mean-normalized stochastic gradient for large-scale deep learning," in *Proc. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 180–184, May 2014.

[16] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, Jun. 2010.

[17] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," [Online] Available: https://arxiv.org/abs/1412.6980, 2014.

[18] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[21] N. Fazakis, S. Karlos, S. Kotsiantis, and K. Sgarbas, "Self-trained lmt for semisupervised learning," *Computational intelligence and neuroscience*, vol. 2016, p. 10, Nov. 2016.

[22] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder network," *CoRR*, vol. abs/1507.02672, Nov. 2015.

[23] M. Pezeshki, L. Fan, P. Brakel, A. Courville, and Y. Bengio, "Deconstructing the ladder network architecture," in *International Conference on Machine Learning*, pp. 2368–2376, May 2016.