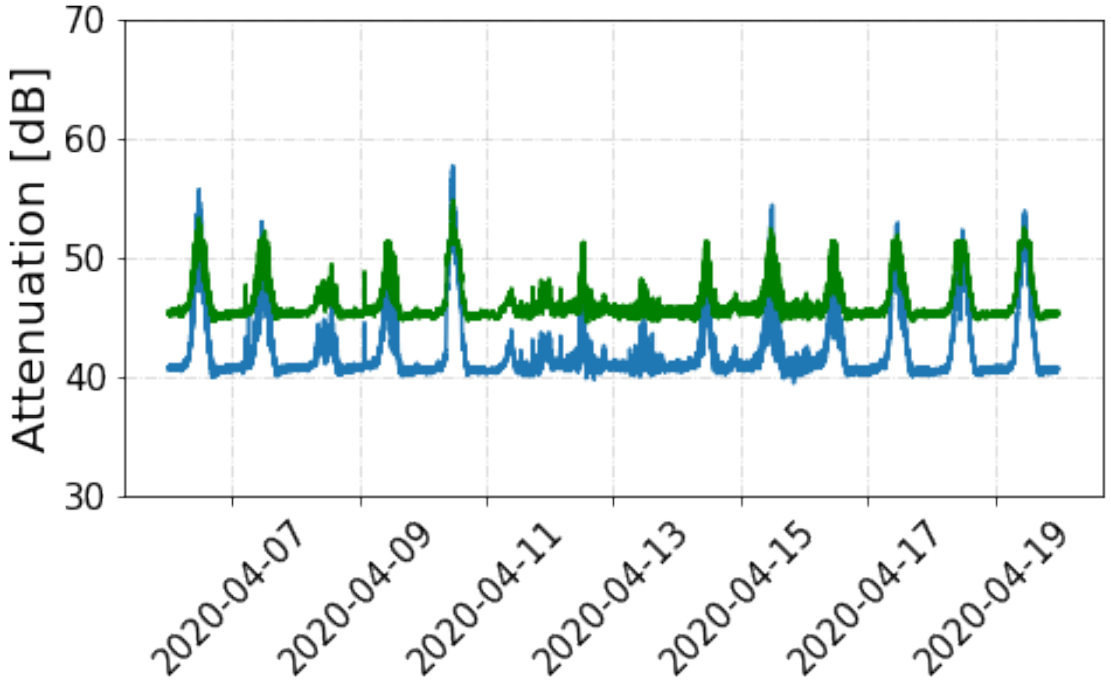

Machine learning algorithm for detecting periodic disturbances of microwave signals

Master's thesis in Mathematical engineering and computational science

Julius Abelson & Oliver Lundmark

Department of Mathematical Sciences

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Machine learning algorithm for detecting periodic disturbances of microwave signals

Julius Abelson
Oliver Lundmark

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2022

Machine learning algorithm for detecting periodic disturbances of microwave signals
Julius Abelson & Oliver Lundmark

Supervisors: Martin Sjödin, Ericsson; Shirin Tavara, Department of Computer Science and Engineering
Examiner: Adam Andersson, Department of Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A periodic disturbance of a microwave signal, indicated by attenuation variation

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Machine learning algorithm for detecting periodic disturbances of microwave signals
Julius Abelson & Oliver Lundmark
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

As digitalisation is expanding into new fields the demand for secure and stable connections between devices are ever-increasing. This can be done via microwave link networks, that comes at a fairly cheap price in comparison with fiber optic cables, but with the con of being more exposed to external disturbances. These disturbances could be caused by several different phenomena, including rain, wind and construction cranes. The scope of this thesis was to expand an already existing tool for detecting disturbances in Ericsson's customer's microwave link network, adding the possibility of detecting and classifying one more disturbance. This disturbance is caused by sunrays, which leads to a thermal expansion on one side of the mast where the node of a link is attached to, causing a miss-alignment between the antennas. The disturbance is called periodic sway, due to its characteristic 24 hours periodicity, correlating to the sun's periodicity.

The tool uses convolutional neural networks (CNN) to detect and classify disturbances. The CNN model needs features to train on to properly classify the disturbances. Today Ericsson's tool uses the features like received signal power and attenuation. When expanding the tool and adding the periodic sway disturbance, further features had to be added, to capture the periodic nature of this particular disturbance. This resulted in adding historical data as a feature.

The conclusions drawn from this thesis are that adding three days of historical data is sufficient for detecting and classifying this disturbance. Furthermore, the results imply that a sparse sampling period of 30-60 minutes is enough for the CNN to detect the periodicity.

Keywords: microwaves, microwave links, microwave disturbances, machine learning, neural networks.

Acknowledgements

We would like to thank our supervisors Martin Sjödin (Ericsson) and Shirin Tavara (Chalmers' Department of Computer Science and Engineering) for their useful help and insights during this project. We also want to show our gratitude to our friends and family, not just for the support during this thesis, but during our whole five years stay at Chalmers.

All the best,
Julius Abelson & Oliver Lundmark, Gothenburg, June 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ELU	Exponential Linear Unit
FFT	Fast Fourier Transformation
ML	Machine Learning
MSE	Mean Squared Error
QAM	Quadrature Amplitude Modulation
ReLU	Rectifying Linear Unit

Contents

1
Introduction

As digitalisation is expanding into new fields the demand for secure and stable connections between devices are ever-increasing. To be able to expand network range and reliability several methods can be used, all with their special traits, pros and cons. Common techniques are, amongst others, fiber cables and directed microwave links. Since signals between microwave links travel through open air they can be disturbed by several phenomena such as rain, snow and refractive index variation. Since the links are bidirectional these disturbances often cause problems at both ends. Furthermore, since antennas are left exposed to the wrath of nature, wind and snow can affect the antennas or even the masts themselves, which in turn affect the signal.

1.1 Background

Ericsson have developed a tool for monitoring networks of microwave links, that with help of a machine learning (ML) model can detect a set of disturbances with good precision. The model identify operating conditions such as Normal, Precipitation (i.e. Rain, Snow, Fog), Signal Obstruction (Construction cranes etc.) and Wind, see Figure 1.1. As can be seen the disturbances affect both directions of the link quite similarly.

These conditions are mainly classified via changes in the signal attenuation, i.e. the ratio of the transmitted power P_{Tx} and the received power P_{Rx}. As the signal travels through a medium a difference between P_{Tx} and P_{Rx} is expected. This difference is called expected attenuation and is usually in a range of approximately 30 to 70 dB, individual to each link and direction. A deviation from this expected attenuation gives a hint of some sort of additional disturbance, see Figure 1.1 and Figure 1.2.

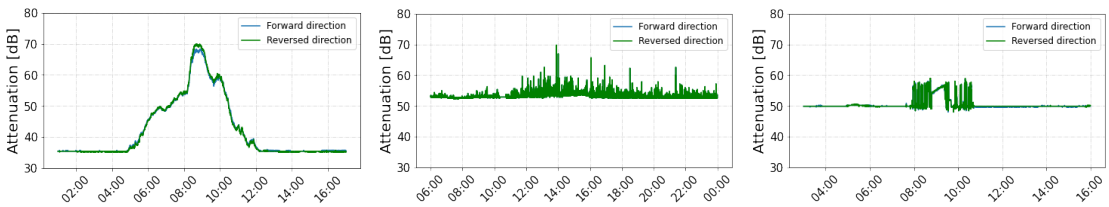

Figure 1.1: Plots showing different disturbances of microwave link signals. The disturbances are caused by, from left to right rain, wind and obstructing objects.

However, most of the classified disturbances as of now are relatively short term and the current model fails to recognise more long term and periodic disturbances. In particular, researchers at Ericsson have found that sunlight variation during the day can cause the sunlit side of the mast, where the antennas are mounted, to expand. This results in a misdirection of the antenna, making the signal attenuation deviate from the expected.

This phenomenon is strongly correlated with the sun's periodicity with a period of roughly 24 hours. Thus, the disturbance will henceforth be referred to as Periodic sway. An example of the disturbance can be seen in Figure 1.2, plotted over a time period of two weeks. The periodicity can clearly be seen, yet many disturbances does not come alone and some rain or wind disturbances can also be seen in the middle of the two week period.

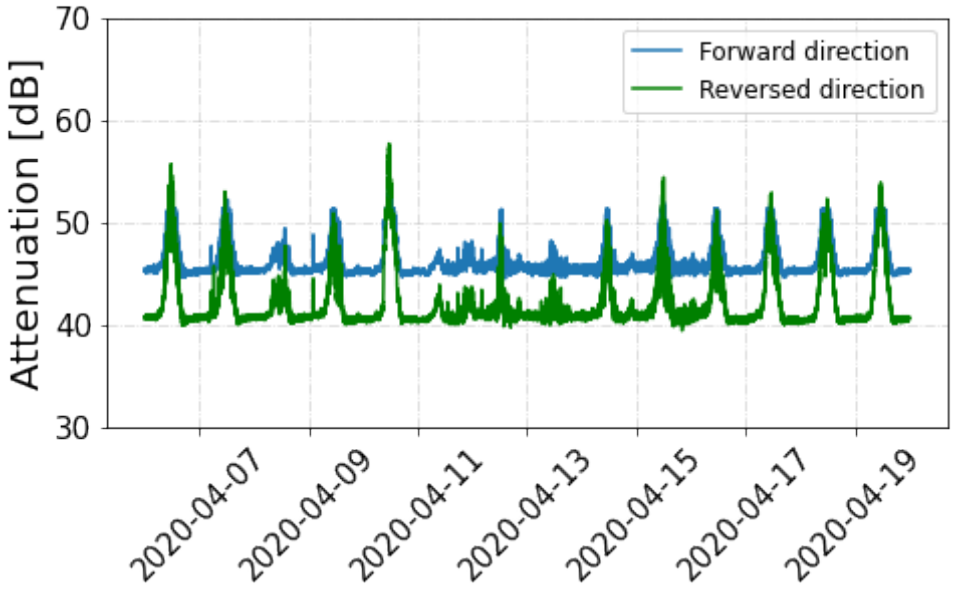

Figure 1.2: Plot showing the attenuation of the Periodic sway disturbance caused by sunlight variation during the day over a two week time period. The attenuation difference between the directions are explained by their different carrier frequency.

The reason why Ericsson needs to monitor the networks is quite simple - the disturbances can cause performance issues. Keeping an eye on temporary deteriorations is a quick way of troubleshooting problems that can occur in the microwave link networks. Before the monitoring system was in place several customers sent back functioning equipment to Ericsson believing that the decline in performance was caused by malfunctioning hardware and not external factors.

Since Ericsson have delivered millions of microwave links, of which a subset is collecting and saving data to a data base, every 10th second, it would not be possible for a human to do the monitoring and operating condition classification by hand. The reliability of the classification is important for the customers, since misclassification can cause the customers to disbelief Ericsson's services. The monitoring of the networks has to be autonomous and expanding the number of detectable operating conditions for quick diagnostics is important.

1.2 Purpose

The goal of the thesis is to answer the following research questions:

- Is it possible to expand the existing models to also include the disturbance caused by sunlight variation without worsening the current performance?
- In order to reduce complexity, how much is it possible to reduce the sampling period of the time series data input or find workarounds and still achieve satisfactory results?

To achieve these goals it is needed to expand the existing training data base.

1.3 Related work

In the specific field of microwave disturbance classification, most of the open research to be found comes from theses at Ericsson. For example Olausson and Sandell [1] investigated how problems in microwave networks can be detected using received power and neural networks. The conclusions from their thesis were that a convolutional neural network (CNN) can be highly useful for microwave network monitoring and with high precision detect and classify disturbances caused by rain and obstructing objects.

The output and solution to the surveillance problem can be applied to more than monitoring microwave link networks. For instance, as shown by Andersson et al. [2], the output for the operating conditions can be used to monitor rainfalls with higher spatial and temporal resolution in comparison with weather radar.

2
Theory

In order to motivate choices of methods, a review of the underlying theoretical framework is necessary. This chapter first contains a short overview of microwave communications and how electromagnetic signals can be used to transmit and receive data. After this the theory and concepts of neural networks will be discussed, with all current techniques and data pre-processing steps that are of use to fully utilise the potential of neural networks.

2.1 Microwave communications

For electromagnetic communication with frequencies smaller than approximately 30 MHz, the environment can be effectively used for signal propagation without free line of sight, as the signals will bounce and refract on the earth and in the atmosphere. However, microwaves are well above that threshold, a free line of sight between receiver and transmitter must be present. This is due to the fact that these waves propagate mostly unhindered through the atmosphere, and hence can't refract or bounce around the curvature of the earth or obstacles like buildings or mountains [3].

2.1.1 Quadrature Amplitude Modulation

To be able to transmit and receive data via microwave links, the data has to be encoded and decoded via a specified schema. Quadrature Amplitude Modulation (QAM) is a method that utilises the cosine- and sinusoidal nature of an electromagnetic wave $s(t)$, i.e.

$$s(t) = A \cdot \sin(\omega t) + B \cdot \cos(\omega t), \tag{2.1}$$

where ω is some angular velocity and A and B are amplitudes of each corresponding wave [4]. Since an electromagnetic wave is continuous in its nature, there is, theoretically, an infinite number of combinations of A and B, which can be used to encode and decode bits. Visualised in Figure 2.1 is a 16 bit example. Note that it is possible with more modulation states than 16.

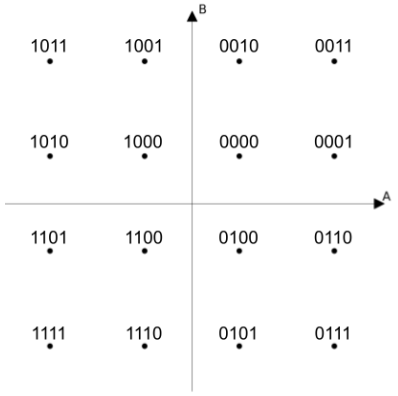

Figure 2.1: 16-QAM constellation de-/encoding four bits per symbol.

2.1.2 Microwave propagation

As discussed in Section 1.1, the attenuation is defined as the ratio of the transmitted power P_{Tx} and the received power P_{Rx}. Even though microwaves travel with little hindrance through open air, an attenuation always occurs and P_{Rx} is normalised in order to get appropriate decoding from the QAM-schema.

2.2 Data pre-processing

Data pre-processing is not only important to obtain the correct type and size of input that the model should process. In this process, it is important to eliminate unwanted features of the data and also eliminate outliers that can affect the performance of the model [5]. For a model to work properly it is also important to have a balanced data set. Otherwise problems might occur, like a over-represented class gets classified more often, only because the data set contain more data of that class [6].

2.2.1 Imbalanced data set and data weighting

A common problem for real world data sets is the problem of imbalanced data sets. A data set is imbalanced if the classes have unequal representation, i.e. ten classes should result in each class taking up roughly 10 % of the total data set. If that is not the case, it can cause a problem with bias if it is not handled appropriately.

There are a few different solutions when working with imbalanced data sets. One solution is to simply add more data. If this can't be done easily, data augmentation can be used in multiple ways, depending on how the data set and the underrepresented classes look. For example, if the data represent images, colors could change, the images could be tilted, flipped etc.

In some cases, augmentation or adding more data to balance the data set is not possible. Then there are other approaches to solve this problem, using techniques like cost-sensitive learning [7]. Cost-sensitive learning works in a way that the model, instead of trying to optimise the accuracy, tries to minimise the total misclassification cost. To use cost-sensitive learning, weights for each class are calculated

depending on how well represented the classes are in a data set. This means that the underrepresented classes will get a higher weight and therefore the model will prioritise to get a higher accuracy for these classes. This because the model want to minimise the misclassification costs [8].

2.2.2 Train/test split and overfitting

A common practice in ML is to split the data set in three subsets - train, validation and test. The training set is used to train the model to learn the patterns of the data set. When training, the validation data set is used to check if the training is moving in the right direction. After the training, the model's performance is validated on the unseen test data set. Since all of these data sets are used together for the model, it is important that the same data does not appear in more than one set.

The reason why the data is divided into three sets is to avoid overfitting. A problem that could occur when training a model is that it learns the patters very well from the training data set and then have a hard time correctly classifying and generalising to other data that is not part of the training data set.

Usually the training data set is a lot larger than the validation- and testing data set. Common practice for the training-, validation- and testing data set sizes could be 70%, 15% and 15% of the total data, respectively.

2.2.3 Normalising

In order for machine learning algorithms to succeed in their goal of finding generalisable models, the data should not contain too many or too large outliers. To be able to get predictable inputs to a model and stabilise learning optimisation, a common ML practice is to normalise or standardise the input data, oftentimes between two numbers like 0 and 1 or scale the input to a normal standard distribution.

In order to scale inputs of a data set between 0 and 1 the formula

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{2.2}$$

is commonly used. x_{norm} and x denote a scaled data point and original data point respectively, whilst x_{max} is the maximum value that x can take and x_{min} is the minimum value that x can take.

2.3 Artificial neural networks

Artificial neural networks (ANN), also called neural networks, are networks inspired by biological nervous systems, how the brain processes information [9]. The way a common architecture of an ANN work is that it uses multiple layers of nodes connected to each other. Together these layers can train to learn different tasks by processing data, similar to a brain. It could be classification tasks for example. An architecture of a basic arbitrary ANN, using fully connected layers, can be illustrated as in Figure 2.2,

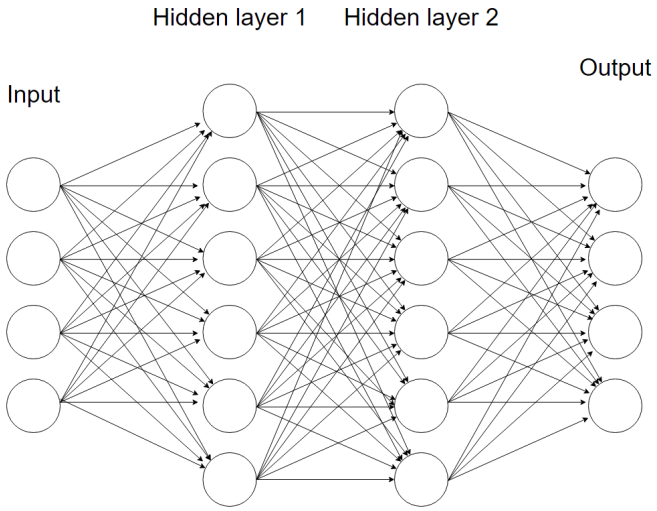

Figure 2.2: Architecture of an arbitrary simple ANN, using fully connected layers, with four feature nodes and four output nodes, showing that it can classify four different classes. The arrows shows the connections between the nodes.

The figure shows multiple layers of nodes, each fully connected to the layer before and after it, this is called fully connected layers. These are commonly used for ANN:s. The four layers shown in the figure are; an input, two hidden layers and an output. The input is the data that the model should process, to learn and classify from. Next comes the hidden layers, those are all the layers in between the input and the output.

Between all of these connected nodes there are more operations happening that are not visualised in this figure. There are weights between the connected layers that the model tries to optimize in able to get a better result. The neurons in e.g. the Hidden layer 1 receive a weighted sum of the input layer and using an activation function passes it on to the next layer, Hidden layer 2, in this case. Then eventually it reaches the output layer, where the predictions are made. Depending on how many classes are used this size will vary. In this example, four classes are used, as Figure 2.2 shows, the output layer have four nodes. All of these layers, parameters and operations will be described one by one in the following sections.

2.3.1 Input layer

The input layer is just the pre-processed data that is fed to the model. Using this data the model should learn to classify the classes of the data, given the labels. This

is the first layer of the ANN and the data have not yet been processed by the model.

2.3.2 Fully connected layers

Hidden layers are the layers between the input and output layer. In Figure 2.2, the hidden layers are of type fully connected layers. Fully connected layers mean that all of the nodes in the previous layer are connected to all nodes in the current layer. Using fully connected layers potentially gives a model a high complexity, depending on the input and network architecture e.g. number of nodes and fully connected layers. The operations made by a fully connected layer can be presented as,

$$Z = W^T X + b. \tag{2.3}$$

Here W are the weights, that the model should optimise to get a desired result. This is usually done by trying to minimise a function and correcting the values of the weights using backpropagation, which will both be described further later in the thesis. X are the input values and b is a bias. Biases can be used to affect the activation function in different ways and are optimised in the same way as W. For the example shown in Figure 2.2 the parameters are,

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & w_{16} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & w_{26} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & w_{36} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} & w_{46} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}. \tag{2.4}$$

2.3.3 Activation function

As mentioned previously, activation functions are used in the neurons of the hidden layers to pass on data to the nodes in the following layer. An activation function takes the values $z_i \in Z$ for $i \in [1, n]$, where n is such that $Z \in \mathbb{R}^n$, where Z is calculated from Equation (2.3). The activation function processes Z to then pass it on to the next layer. Depending on where in the architecture the activation function is located, it can serve different purposes. Between the hidden layers a common activation function is the Rectifying Linear Unit (ReLU). This activation function is formulated as,

$$g(z_i) = \max(0, z_i), \quad z_i \in Z. \tag{2.5}$$

This activation function has commonly been used for a long time, because of its advantages with handling some problems like vanishing gradient [10].

There are more variations of activation functions used in the hidden layers. Another activation function that is not linear is the Exponential Linear Unit (ELU). This activation function have proven to give better classifications results than the ReLU in some cases [11]. The ELU activation function is computed by the following equation,

$$g(z_i) = \begin{cases} z_i, & \text{if } z_i > 0 \\ \alpha(e^{z_i} - 1), & \text{otherwise} \end{cases}, \quad z_i \in Z, \tag{2.6}$$

for some $\alpha > 0$. Here it is clear that this is not a linear activation function, except locally for positive z_i. For values where $z_i < 0$ the ELU activation function returns negative values. This allows the activation function to get a mean activation unit closer to zero. This is like batch normalisation but with lower complexity. Hence the complexity of the model can be smaller using ELU instead of ReLU [11].

As mentioned earlier, different activation functions can be used for different purposes. The ones described above, ReLU and ELU are mostly used in the hidden layers, but not for the output layer and classification purposes. As seen in Figure 2.2 the output layer have four nodes, this means that it has four classes it can classify. In the output layer another kind of activation function is needed. In this case, with more than two classes, a binary classification is not preferred and something like the softmax activation function can be used. The softmax function returns a probability distribution between the different possible classes and from that probability the class with the highest probability is the model's prediction. Softmax can be written as,

$$g_i(Z) = \frac{e^{z_i}}{\sum_j^n e^{z_j}}, \quad \forall\, i \in n \tag{2.7}$$

where Z is the input vector and n is such that $Z \in \mathbb{R}^n$, and represents the number of output classes.

2.3.4 Objective function

An objective function in ML is used for measuring the quality of a solution, the output that is. This function is often called the loss function. This functions is in place to minimise the loss, which is a measurement of the prediction. If the prediction is very off, the error is large and then the loss is high. These predictions occur when the model is learning by training on the training data set. By using an optimisation algorithm and minimising a cost function the model can tune its parameters to make better predictions.

There are multiple different cost functions. Depending on what the task at hand is, the choice of cost functions can differ. A common cost function is the Mean Squared Error (MSE) cost function. The MSE cost function is calculated with the equation,

$$J(Y, \hat{Y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{2.8}$$

where N is the number of samples tested against, $y_i \in Y$, where Y is the vector of true labels with $N - 1$ zeros and a one, representing the true class. And $\hat{y}_i \in \hat{Y}$, where $\hat{Y}$ is the a vector of size N with the sum one, but divided according to the prediction of the model.

For other tasks, like classification, multi class cross-entropy loss might be used. This loss is calculated using the equation

$$J(Y, \hat{Y}) = -\sum_{i=1}^{N_c} y_i \cdot \log(\hat{y}_i), \tag{2.9}$$

where N_c is the number of classes to predict, $y_i \in Y$, where Y is the vector of true labels with $N-1$ zeros and a one, representing the true class. And $\hat{y}_i \in \hat{Y}$, where $\hat{Y}$ is the a vector of size N_c with the sum one, but divided according to the prediction of the model. This means that the model will try to reach the value negative one in Equation (2.9).

2.3.5 Optimisation

In Section 2.3, optimisation of weights was mentioned, in order for the model to work. This is done via optimisation of a loss function. There are different optimisation algorithms that can be used for this, but a common one is to use gradient descent. This works by minimising a function's parameters by updating them in the opposite direction of the gradient for the function it is minimising. In the case of learning for a model it is the weights that are the parameters, denoted θ. For a cost function denoted $J(\theta)$ the gradient descent update function can be written as,

$$\theta_{i+1} = \theta_i - \eta \cdot \nabla_{\theta_i} J(\theta_i), \quad i = 1, 2, 3... \tag{2.10}$$

where η is a scalar learning rate. For ANNs the objective function is usually called a loss function. To update the weights, backpropagation is used. This is done by calculating the gradients of the loss function w.r.t. the weights, $\nabla_\theta J$, updating the weights according to Equation (2.10), and then applying the chain rule to iterate backwards until all weights are updated.

Like the activation function there are also many algorithms for optimisation of the weights and the learning rate. Besides the gradient descent another common function is adaptive moment estimation, also called Adam. The Adam optimiser algorithm is a type of stochastic gradient descent algorithm. The Adam optimiser tries to minimise the expected value of the loss function, that is, $\mathbb{E}[J(\theta)]$. As the name adaptive moment estimation suggest, Adam works in a way that it uses estimates of the first moment (mean) and the second moment (uncentered variance) of the gradients. Adam uses the following updating algorithm [12],

Algorithm 1 Adam, algorithm for stochastic optimisation. g_t^2 indicates the element-wise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$. $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise, where β_1^t and β_2^t are β_1 and β_2 to the power of t.

Require: α: step size
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $J(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialise 1st moment vector)
  $v_0 \leftarrow 0$ (Initialise 2nd moment vector)
  $t \leftarrow 0$ (Initialise time step)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta J_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at time step t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

2.4 Convolutional neural network

Convolutional neural networks (CNN) are a class of ANNs. One of the characteristics of a CNN are the convolutional layers. These layers allow the CNN to potentially solve more complex tasks than that of an ANN with lower computational power [13], by extracting specific key features. This is because ANN:s makes the input one dimensional by flattening the input, if needed, and therefore it takes a lot more computations for it to process the input, assuming it uses fully connected layers.

The architecture for CNNs can vary depending on the task at hand. If a CNN is used for complex data, it has to learn complex patterns, then more filter layers could be used. If the data contains multiple inputs (features) then it needs another kind of architecture, that does several feature extractions, one for each feature input. Figure 2.3 shows a simple CNN architecture with two hidden layers.

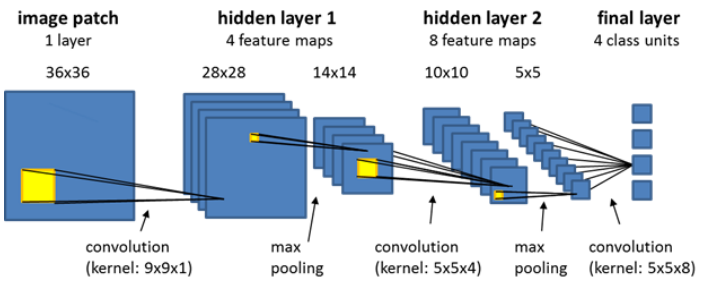

Figure 2.3: Architecture of an arbitrary convolutional neural network. Picture from `https://docs.ecognition.com/`.

This example displays an image as the input and shows how the convolutions work. The yellow square on the left most side represent a filter, that is a matrix with set weights that is used for finding different patterns. Each filter convolves over the entire image input. For each step over the image that the filter takes each cell in the filter matrix is multiplied with the cell in the image input and added up to a sum. Each filter creates a feature map of all values calculated from the filters. This process is visualised in Figure 2.4.

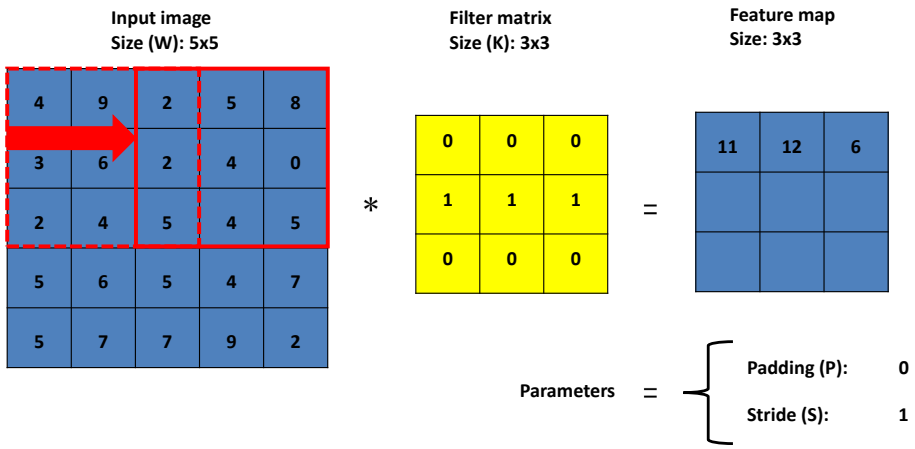

Figure 2.4: Schematic figure of how a filter-operation work. Here the filter represents a horizontal pattern. Stride is the step size of the filter and padding is a method for adding zeros to the input in order to manipulate the size of the input.

As seen in Figure 2.3 the filters after the first convolution can be three dimensional. This is because even thought the input is two dimensional, the feature maps together creates a higher dimension i.e. three dimensions. Three dimensional filters work the same way as two dimensional, but it also uses summation over one more dimension.

The features maps are displayed in the middle section of Figure 2.3. As shown in Figure 2.3 one image gives multiple feature maps. This is because each filter results in a feature map. The size of these feature maps depends on the size of the input, filter size and stride. The size of the stride determines the step size of the filters over the input, that is how many cells the filter move by each step. Also an input could be padded with zeros around the edges. This can be done so that the output feature map does not shrink too much. The result is a feature map of size,

$$\text{feature map}_{\text{size}} = \frac{W - K + 2P}{S} + 1. \tag{2.11}$$

where W is the size of the input, K is the filter size, P is the padding of the image and S is the size of the stride.

The last part of the hidden layer is a pooling layer. These layers can be used to reduce the complexity even further. A common pooling layer function is called

max pooling and works similar to the filters. Max pooling function returns the maximum value for patches, in this case patches of the feature maps, with a specific size of the pooling matrix and also with a stride. Using max pooling the pooling matrix only returns the largest value from each position the pooling matrix takes over the feature maps. Then the output of the pooling layer can be calculated in the same way as the output of the filters, according to Equation (2.11).

In Figure 2.3, after the first hidden layer, another hidden layer is added. These layers can be added to find more complex patters. But adding too many of these could make the model overfit to the specific data [14].

A common way of handling overfitting when training a model is dropout. Using dropout means that there is a chance that some units in the network is dropped. This means that the units are temporarily removed from the network. The chance for a unit to dropout in the model is decided at random, with a set chance for dropout, p. This is a typical way to prevent overfitting in networks [14].

The output from the hidden layers are flattened and then the final predictions are made using fully connected layers. These fully connected layers work as explained in Section 2.3.2. Solely using fully connected layers can quickly result in a lot of computations. When the input is large, maybe an image, then the computation needed to make classifications easily become very large, since as X in Equation (2.4) grows, W and b does as well. Using convolutional layers as feature extractors and using their output as input fully connected layers is therefore favourable, due to the lower amount of parameters needed in the convolutional layer.

2.5 Performance measurements

To asses which models and features yields the best results, several measurements of the outputs can be done to ensure reliability and speed. In general, no one of these measurements are more important than the other and a subjective decision has to be taken in order to evaluate which models and features to move on with, depending on what the aim and usage of the model would be.

Essentially, there are four different ways to label the classifications made of a model in comparison with the true values,

- **True positives (TP)** - Correct classifications as belonging to a specific class,
- **True negatives (TN)** - Correct classifications as not belonging to a specific class,
- **False positives (FP)** - Incorrect classifications as belonging to a specific class,
- **False negatives (FN)** - Incorrect classifications as not belonging to a specific class.

With this in mind it is possible to build up metrics to evaluate a model's performance based on different attributes.

2.5.1 Accuracy

Perhaps the most obvious and straight forward measurement is at what percentage rate the model asses the correct label to an input. This could be done for a data set as a whole, but also individually for each class. The score is obtained via the equation

$$\text{Accuracy} = \frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i},$$

where i could be representing a class or the data set as a whole.

2.5.2 Precision

In order to gain a deeper understanding of a model's trustworthiness and reliability the precision can be taken as a good metric. It calculates the fraction between the true positives and all positives, i.e. if a model predicts a certain class, how sure can one be that it is correctly classified. The precision is calculated with

$$\text{Precision} = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i},$$

where i could be representing a class or the data set as a whole.

2.5.3 Recall

While the precision metric focuses on the events actually classified as a certain class, recall puts weight on how good the model actually is a finding all events in a class. Thus, these two metrics complements each other in a good way. The metric is calculated using the fraction

$$\text{Recall} = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

where i could be representing a class or the data set as a whole.

2.5.4 F1 score

With the previously discussed measurements precision and recall obtained, it is possible to combine those and calculate a F1 score. The formula is given by the equation

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}},$$

which gives a range as $F_1 \in [0, 1]$, where 1 is the desired value.

2.5.5 Time consumption

As many models operate in a live production environment, demands will oftentimes be set for the time consumption of the models. How this time consumption is prioritised, defined and investigated depends on the area of use. For example, if a model needs to be updated frequently with new data the speed of the training process could be of highest importance, while if the model doesn't need to be updated too

often, this could be neglected. In some other case, the model might be expected to take rapid decisions, which then means that it has to be quick on processing the input and give a output.

3
Data

In machine learning projects, appropriate data and data handling is a key factor for success. No model is better than the data it has been trained and evaluated on. Hence this section is used to described the process of data gathering, pre-processing and feature selection.

3.1 Data set expansion

Before any training of models or even pre-processing of the data could start, the data set obtained by Ericsson had to be extended with data representing the new class, Periodic sway. Furthermore, the data set had to be expanded with the historical attenuation of the disturbances, since it was expected that the models would need some sort of input capturing the periodicity of the new disturbance. Ericsson have gathered data from thousands of microwave links over a few years period. This means that Ericsson possess data containing thousands of days for thousands of links, each with data collected every 10th second. Whilst this data is stored in Influxdb, a subset of this data have been extracted to Mongodb, in order to train the models. Influxdb and Mongodb are two data bases used for storing data by Ericsson.

To find data for the Periodic sway class, saved data in Influxdb were processed. First a broad filter, given by Ericsson, using Fast Fourier Transformation (FFT) was used to search for disturbances with a period of 24 hours. This narrowed it down to approximately a few hundred thousand days of data, also known as events, see Section 3.3. Since it was still too many events to look through manually, it was narrowed down even further by only taking the links containing over 150 interesting events from the FFT-filter, to manually inspect. These contained approximately 8,000 events. After all of these events were reviewed only 356 events were selected from 28 links and added as training, validation and test data in Mongodb. For information about the collected data, see Section 3.6.

The time series data sent to Mongodb had a sampling period of ten seconds, and additional features were stored as single values. The most important features were

1. Link name
2. Transmitted signal power (24 h)
3. Received signal power (24 h)
4. Attenuation (24 h)
5. Expected attenuation, i.e. the attenuation with no disturbance
6. Attenuation history, i.e. the attenuation two weeks prior to the event
7. Start and end time of event
8. Carrier frequency of the link

For some links the data collection is flawed and therefore collecting continuous attenuation data was not feasible. Thus these links do not have the data for two weeks of attenuation history available. For these events artificial attenuation history had to be constructed. How this was handled is described in Section 3.3.

Out of the things listed above, received signal power, attenuation, historical attenuation, a date feature, and the operating frequency was used as input features to the models, see Section 3.7.

3.2 Amplifying weak disturbances

For some events of the class periodic sway the difference between the peak value of the attenuation and the expected attenuation was small. In order to not make the model too sensitive, by mixing up normal operation with Periodic sway, a transform to these weak events was applied. The transform's purpose was to amplify the attenuation peaks of the periodic sway disturbances.

Thus, in order to not waste any of the rare data collected, the transform

$$\text{attenuation}_{\text{new}} = 2 \cdot \text{attenuation}_{\text{old}} - \text{expected\_attenuation},$$

was used on these events to make sure that the events with too low deviance from the expected attenuation could still be used in the training process. As described in Section 1.1, the expected attenuation is the attenuation when there is no disturbance of the signal, i.e. operating condition Normal.

The transformation kept attenuation samples near the expected attenuation close to their original values while still amplifying the peaks, which kept the natural traits (noise, other disturbances etc.) of the time series. An example can be seen in Figure 3.1.

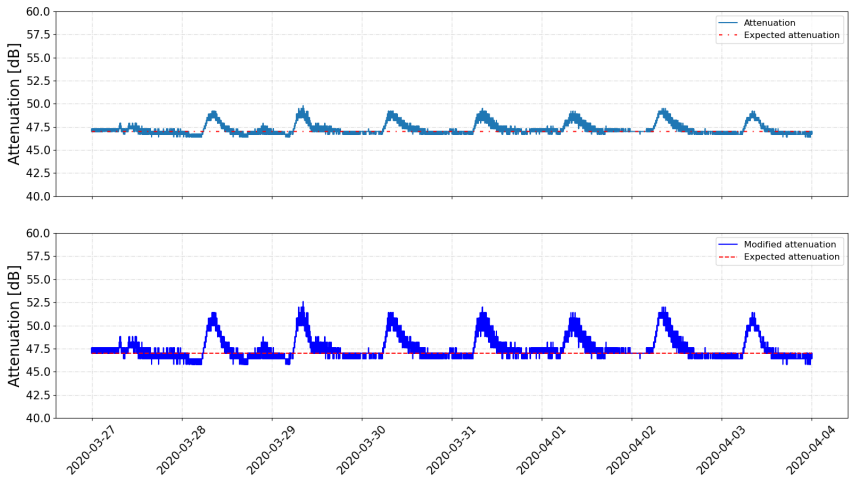

Figure 3.1: Difference between the original attenuation and the transformed attenuation, where the bottom plot shows the transformed state.

3.3 Input window generation

Before the data in the data set were sent to the model for training and evaluation, pre-processing of the data was performed on top of what's previously described in Section 3.2. Some of the data for the historical values of the attenuation could not be found. To still be able to use that data in the training of the models, the missing data had to be artificially generated. If the historical attenuation was missing in just one direction, the historical attenuation from the other direction was mirrored and used for the missing direction, with compensation for a (possible) deviance in the expected attenuation between the directions. This compensation was done by subtracting/adding the difference in expected attenuation between the directions. For the case where attenuation history was missing in both directions, data representing the most common operating condition, i.e. normal and no perpetuation, was generated. This was done by setting all the samples of the historical attenuation to the expected attenuation.

To further expand the training data beyond the events added to the data base, each event was utilised to generate several windows, as Figure 3.2 shows. In the figure a window size of six hours is used and the stride, i.e. the spacing between the start time of two following windows is also set to six hours. The stride is set to such a high number only for the visual presentation, for the live generation of windows a stride between 5 and 60 minutes, depending on class, was used. Furthermore, the figure only shows the procedure for the attenuation of the event, yet the same procedure is done for the received power. Also, the historical attenuation was selected so that its end time coincided with the starting time of each window, as the stride moved forward through the event.

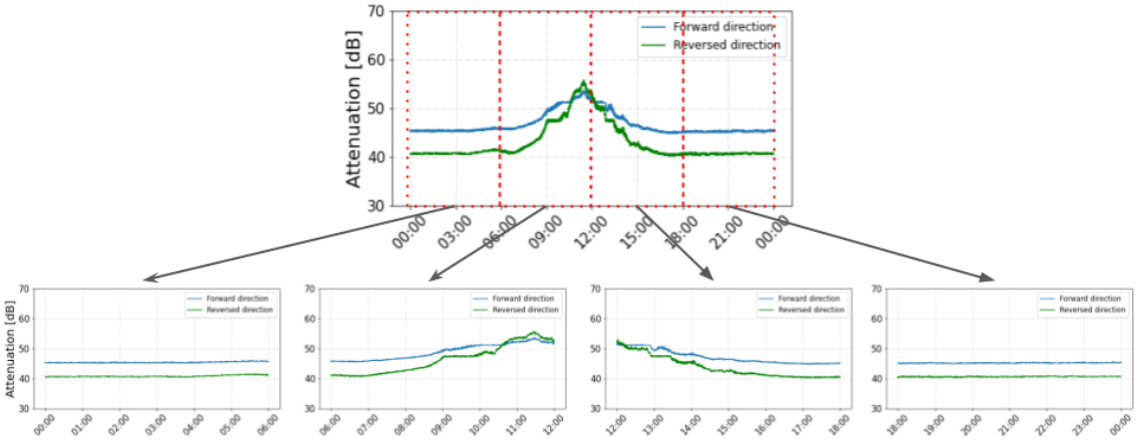

Figure 3.2: The figure shows how an event (top) generates four different windows (bottom), each covering a duration of six hours. In this particular case a stride of six hours between every start time is used. Usually an event generates up to 300 or more windows, depending on the stride.

Though the main event usually contain a disturbance, some of its sub windows may still be without any disturbances. Hence each window was given a classification, either normal or the event's classification. Since it was desirable with a model that was able to classify the disturbances with one hour resolution, the classification of

each window was set to the event classification if a given number of points deviated past a threshold value from the expected attenuation during the last hour of the window. The number of points and the threshold value were being individually set for each class based on previous tests and research at Ericsson.

3.4 Reducing complexity

Since the bulk of the data fed to the models potentially could be derived from the attenuation history, it was of interest to examine ways of reducing the size of the attenuation history feature, without loosing too much information. The tested methods for reducing the number of parameters are described below.

3.4.1 Sample period reduction

As the original data is sampled with a period of ten seconds it was considered that a lot of the historical attenuation data would be redundant, since the periodic disturbances could be seen as a trend over a longer time period rather than over a short time period. Figure 3.3 shows the historical attenuation of an event with different sampling periods - top left a period of 10 seconds, top right of 1 hour (3,600 seconds), bottom left 6 hours (21,600 seconds) and finally bottom right with 12 hours (43,200 seconds). As seen in the figure, the trend is still observable when going from a sampling period of 10 seconds to 1 hour. With a sampling period of 12 hours the trend is somewhat rugged and not as easily interpreted. Note, this is only an illustrative example from one specific event and can not be taken as a truth for the data set as a whole.

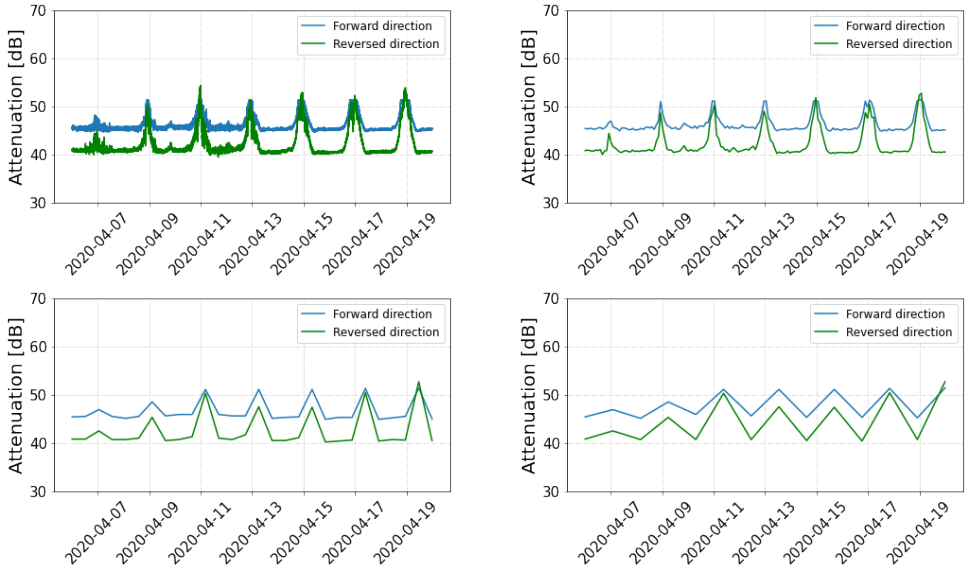

Figure 3.3: Plots of the same historical attenuation for an event with different sampling period. The top left shows a sampling period of 10 seconds, top right of 1 hour, bottom left 6 hours and finally bottom right with 12 hours.

3.4.2 Length reduction

Perhaps the most obvious and straightforward way to reduce the size of the attenuation history feature is to use shorter time spans than two weeks, as described in Section 3.1. If it was possible for the model to classify the periodic sway disturbance with one week of history instead of two, it would imply a huge reduction of the amount of input data.

3.5 Train and test split

As Figure 3.3 shows, the resemblance between disturbances of different days is striking. However, this only holds for events from the same links, and the appearance of the disturbances varies between links, both in length and strength. This could be explained by different surroundings and mountings of the antennas. In order to make sure that the model is able to generalise and not just learn to recognise specific events, it was important to not split up events from the same link when distributing the Periodic sway data among the train, validation and test sets as described in Section 2.2.2.

Since the events of each link were so alike, overfitting wouldn't be easily detected, as similar events could be contained in all data sets. Hence, events from a specific link were all kept within the same data set, i.e. all events from link 1 were contained in the train data set, all events from link 2 in the validation data set etc, while still keeping appropriate ratios between the different data sets, see Section 2.2.2.

3.6 Periodic sway characteristics

To get a better understanding of the Periodic sway data, some characteristics could be observed. It is important to know that the Periodic sway disturbance did not seem to be the most frequent, disturbances caused by for example rain or wind seemed to be more common. Consequences of this were a skew in the number of windows to train on, since more events were collected from more common disturbances. This is not a problem solely for the Periodic sway disturbances but also for other rare classes. This imbalance was solved by weighting the classes individually, as described in Section 2.2.1.

As for the properties of the Periodic sway events, it can be seen in Figure 3.4 that it occurs predominantly in the warmer months of the year, in particular during spring.

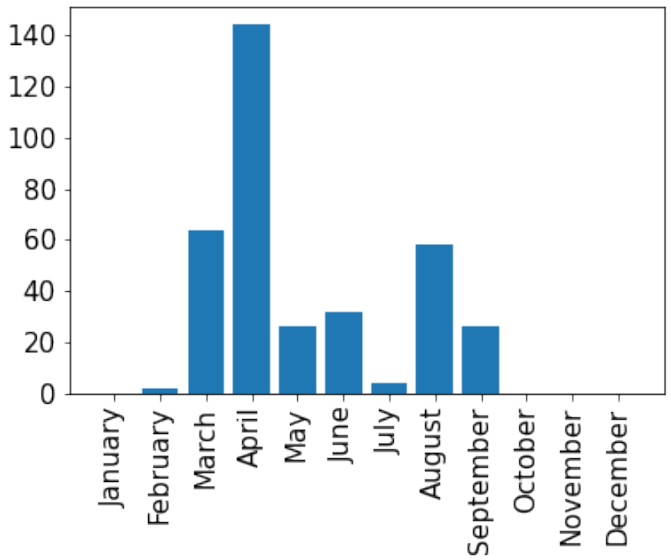

Figure 3.4: Distribution of Periodic sway events over the year.

Yet, one should know that the number of events is too small to draw any far-reaching conclusions. That the events mostly occur during the warmer months is no surprise, however since each single link have such a big impact on the total number of events the tilt to the spring side could be due to the placing and sun exposure of single links.

3.6.1 Distribution after train and test split

Statistics for the splits and the distribution of Periodic sway between the data sets can be seen in Table 3.1. The split fractions were decided by the share of periodic sway windows while still keeping all the windows from the same link in the same set. Notably, this caused some skewness, the training data set is for example made up of events from only four links, but these events had large disturbances over a long time period and thus rendered more windows per event than the events in the validation- and test sets. Furthermore, a large fraction of the events in the validation set had to be modified, as described in Section 3.2.

Table 3.1: Statistics for the class Periodic sway and the different splits, showing the number of links, events, events that were modified and the number of generated windows.

#	Train	Validation	Test	Total
Links	4	16	8	28
Events	150	154	52	356
Modified events	12	151	21	184
Periodic sway windows	3143	345	396	3884
Normal windows	545	72	259	876

3.7 Input feature and batch generation

After the window generation, as described in Section 3.3 and Section 3.4, the received power, attenuation and historical attenuation had to be normalised. The received power was normalised by using the min-max formula stated in Equation (2.2), after adding 100 dB to x, where $x_{\min} = 0$ and $x_{\max} = 80$, relying on domain knowledge from Ericsson. The attenuation and historical attenuation was normalised by first subtracting the expected attenuation to get level out differences between the links and then applying the min-max formula with min $x_{\min} = -50$ and $x_{\max} = 50$, once again using Ericsson's domain expertise.

On top of this, four time features were created, representing

1. Day of the week, i.e. 0 for Monday, 1 for Tuesday etc., normalised by division of six,
2. Hour of the day, normalised via division of 24,
3. A binary variable set to 1 if the hour of the event was between 4 AM and 9 PM,
4. A binary variable set to 1 if the month was among one of the winter months December, January, February or March,

Finally, a binary frequency feature was added, set to 1 if the link operated on a frequency above 71 GHz and 0 otherwise.

3.8 Testing scheme

Improvements of the architecture of the model already in use by Ericsson was tested. An extra hidden convolutional layer was added to the model. This updated model was tested by training the model on the same data set, using the same features, and comparing the results.

In order to evaluate which input that will yield the best results for a live model in production, several different data sets were generated, as can be seen in Table 3.2. Note that only the length and sampling period of the historical attenuation features were varied, and that the attenuation and received power were kept the same, as previous research at Ericsson had shown this to be successful.

Table 3.2: The different settings used for the historical attenuation.

Name	Length	Sampling period
Baseline*	–	–
Baseline	–	–
1 day/5 min.	1 day	5 minutes
3 days/5 min.	3 days	5 minutes
1 week/5 min.	1 week	5 minutes
2 weeks/5 min.	2 weeks	5 minutes
1 day/30 min.	1 day	30 minutes
3 days/30 min.	3 days	30 minutes
1 week/30 min.	1 week	30 minutes
2 weeks/30 min.	2 weeks	30 minutes
1 day/60 min.	1 day	60 minutes
3 days/60 min.	3 days	60 minutes
1 week/60 min.	1 week	60 minutes
2 weeks/60 min.	2 weeks	60 minutes

* With no periodic sway events included

First of all two baseline data sets were generated. The first did not contain the new periodic sway events, in order to be able to evaluate how the newly added class would affect the performance of the other classes. The second baseline data set included Periodic sway events, but not the historical attenuation feature. This was done to obtain a benchmark result and to asses the impact of the historical attenuation feature. After this, several data sets were generated, with different time spans and sampling periods, all visual in Table 3.2.

4
Models

In this chapter the model used by Ericsson today to monitor the microwave link networks and the new updated model will be described. Furthermore, the process of updating the model will also be described.

4.1 Ericsson's model

The model used by Ericsson is presented by Figure 4.1.

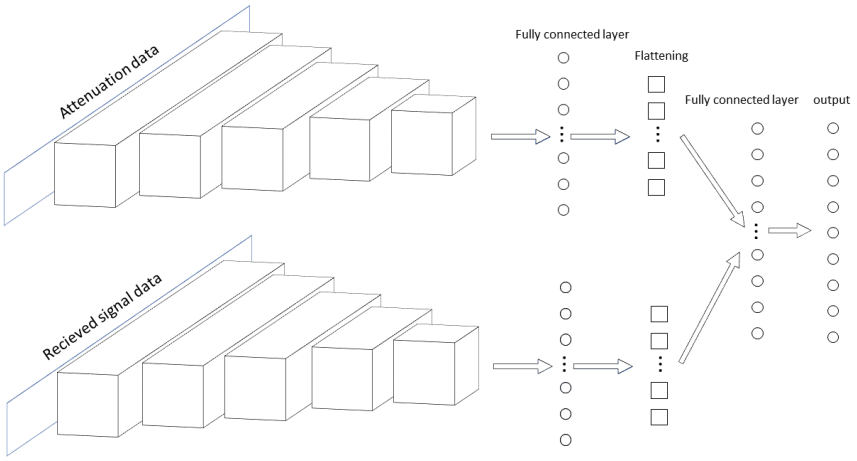

Figure 4.1: The model given by Ericsson.

As the figure shows, the model contains two feature extractors, one for attenuation and one for received signal power. The feature extraction is represented by the boxes stacked after each other and the fully connected layer following the boxes. These boxes are the hidden layers explained in Section 2.4. Additional features are dummy variables as explained in Section 3.7. Other parts of the model that cannot be seen in Figure 4.1 are the parameters used. All of the parameters that can be manually set are shown in Table 4.1.

Table 4.1: The parameters and function used in the model given by Ericsson.

Parameter	Values
First activation function	elu
Second activation function	softmax
Optimiser	adam
Pooling	max pooling
Padding	same*
First fully connected layer	16 nodes
Second fully connected layer	32 nodes
Number of filters	16
Filter size	3x3
Stride	1x1
Pooling size	1x2
Epochs	300
Filter layers	5
First dropout	10 %
Second dropout	30 %

* Function in the Keras library. In this case the function pads the inputs with zeros on the top, bottom, left and right side of the input.

Some parameters there are two of, like the value of the dropout. Then the denotation "first" is used, because it comes first in the model in Figure 4.1.

4.2 Updated model

The model received from Ericsson has been tested and used in action, therefore no big changes had to be made, since the baseline model was already performing well. Not a lot of time was spent on trying to optimise this model. However, some small changes were made.

The first change was adding an extra hidden layer. The impact of this was tested by training one model with and one model without an extra layer, and comparing the results. It was then decided to use the extra layer, since the performance was similar but the number of model parameters was smaller.

After this was done an extra feature extractor was added to the model, so that the model could use historical attenuation as an input as well. This was done by copying one of the other feature extractors and adding this to the model, with one change. The feature of historical attenuation would vary in input size of the data, depending on the sampling size and length of the history of the historical attenuation, as shown in Section 3.8. Since the input size would vary, so would also the output of the feature extraction of the historical attenuation. To manage this, not a set amount of filters were used, instead a value for the size of the output from the feature extraction of historical attenuation was used. It was known that the other features had a fixed input and therefore also a fixed output from their respective feature extraction. For the sake of the complexity of the model, a matching size

of the output would be preferred for feature extraction of historical attenuation. Therefore a loop of these boxes shown in Figure 4.1 was created. The data for the historical attenuation feature would go through this loop until the output of the feature extraction was below this value, set to 50, to match the other feature extraction outputs. This feature extraction was added in order to optimise the learning for the new class added, Periodic sway.

The only additional change was to reduce the training, reducing the number of epochs used for training, which reduced the training time from almost a week in some cases, to a maximum of two days, leaving more time for experimenting with the time span and sampling periods of the historical attenuation.

All these changes resulted in a model with the parameters shown in Table 4.2.

Table 4.2: The parameters and functions used in the model given by Ericsson, after it had been updated.

Parameter	Values
First activation function	elu
Second activation function	softmax
Optimiser	adam
Pooling	max pooling
Padding	same*
First fully connected layer	16 nodes
Second fully connected layer	32 nodes
Number of filters	16
Filter size	3x3
Stride	1x1
Pooling size	1x2
Epochs	100
Filter layers	6
First dropout	10 %
Second dropout	30 %

* Function in the Keras library. In this case the function pads the inputs with zeros on the top, bottom, left and right side of the input.

4.3 Evaluation and selection of models

To evaluate which of the proposed models that yielded the best results, several metrics were used. Generally speaking, the models should be able to correctly classify as many Periodic sway events as possible without degrading the performance for the other classes. Hence, in order to get an overview and examine which models to investigate further, the overall F_1 score was taken as a measurement for the overall performance of the model, since it captures both the recall and precision. To evaluate the performance for the Periodic sway class its recall and precision were calculated. Further, the time consumption described in Section 4.3.1 was also taken into consideration.

After training and validating each model once, some models were selected on the basis of the metrics described above, the process was repeated and the results averaged. These models were more thoroughly inspected, as the recall and precision of each class were calculated, taking the mean of both metrics for the three runs. To further investigate where errors were caused, the confusion matrix was also inspected.

4.3.1 Time consumption

As the final model is supposed to be used in a live production environment and classify thousands of windows every hour, time consumption and computational complexity are key factors for assessing the quality and usability of the models. As the live data is stored in Influxdb, this data have to be fetched, pre-processed, fed through the model and then transmit the classification results back to Influxdb.

For a fair evaluation of the time efficiency, the time required for each model to fetch the same 100 events, pre-process and classify those events, was measured. To get a more fair estimate the procedure was repeated 20 times for each model, with the mean of these 20 runs taken as the final value.

5
Results

All results from the training of the models will be shown in this chapter. First, the results for selecting the baseline model are presented, thereafter the results for the models including the periodic sway class are shown. Then, an overview of all models is given, and following that the results of the most promising models are further investigated and presented.

5.1 Selection of baseline model

The results from testing the model provided by Ericsson can be seen in the normalised confusion matrix shown in Table 5.1. Overall the performance of the model is good, i.e., reaching close to or even hitting the 100% accuracy mark for all classes, apart from disturbances caused by snow. For the classification of snow disturbances, the model provided by Ericsson misclassified 10% of the total windows as rain.

Table 5.1: Results from the tests of the model provided by Ericsson, presented with a normalised confusion matrix.

		Predicted labels							
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
True labels	Normal (1)	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Rain (2)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
	Construction (3)	0.00	0.00	0.98	0.00	0.00	0.00	0.02	0.00
	Multipath (4)	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
	Maintenance (5)	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
	Restart (6)	0.00	0.00	0.00	0.00	0.01	0.99	0.00	0.00
	Wind (7)	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
	Snow (8)	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.90

The results from testing the modified baseline model are provided in Table 5.2. Overall the performance of the model is good, i.e., reaching close to or even hitting the 100 % accuracy mark for all classes. In contrast to the unmodified model, it has no big flaws in the classification of snow disturbances.

Table 5.2: Results from the testing of the modified model, presented via a normalised confusion matrix.

		Predicted labels							
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
True labels	Normal (1)	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Rain (2)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
	Construction (3)	0.00	0.00	0.98	0.00	0.00	0.00	0.02	0.00
	Multipath (4)	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
	Maintenance (5)	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
	Restart (6)	0.00	0.00	0.00	0.00	0.01	0.99	0.00	0.00
	Wind (7)	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
	Snow (8)	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.97

After testing both the unmodified and the modified model it was decided to move on with the modified model as the baseline model, since the modified model's ability to better classify the snow disturbance.

5.2 All models

The results from testing the models using the data sets from Table 3.2 are presented in Table 5.3. Models with promising results were further examined and the results from these tests are presented in Section 5.3.

Table 5.3: The results with the different time spans and sampling periods used for the historical attenuation.

Data set [len./sam. per.]	F_1 score [%]	Precision Periodic Sway [%]	Recall Periodic Sway [%]	Time fetch data [s]	Time classification [s]
Baseline*	99.32	–	–	26.472	6.790
Baseline	95.34	85.00	52.02	26.845	7.603
1 day/5 min.	97.46	82.41	79.29	33.499	5.706
3 days/5 min.	98.36	93.63	89.14	40.261	8.374
1 week/5 min.	98.23	95.08	87.88	53.605	10.332
2 weeks/5 min.	93.64	62.08	55.81	75.113	9.509
1 day/30 min.	97.06	89.12	76.52	32.284	5.065
3 days/30 min.	98.69	95.69	89.84	33.525	7.827
1 week/30 min.	98.07	89.00	91.92	35.951	8.619
2 weeks/30 min.	95.34	98.52	50.51	41.338	8.718
1 day/60 min.	97.29	79.33	83.33	31.415	5.158
3 days/60 min.	98.48	91.41	91.41	32.412	8.325
1 week/60 min.	97.63	90.33	89.64	34.753	7.938
2 weeks/60 min.	96.07	66.87	82.58	38.419	7.836

* With no periodic sway events included

5.3 Promising models

As explained previously, not all models are of interest, since some perform better than others in terms of the chosen metrics and the complexity is better (lower) for some models. Based on the results and using a variation in the length and sampling period of the historical attenuation, shown in Table 5.3, it was decided to move on with four models for further testing. Those models were using 1 week/30 min., 1 week/60 min., 3 days/5 min., and 3 days/60 min. of historical attenuation.

In Table 5.4, the means of precision and recall after three training sessions are shown for each class and each interesting model. Overall the metrics are high and consistent, with the exception of the classes Snow and Periodic sway, where the metrics are worse and the standard deviation σ larger.

Table 5.4: The recall and precision for each class and each interesting model, shown as means after three training sessions. The standard deviations σ are shown within the parentheses.

(a) Model using 1 week/30 min.

Class	Precision [%] (σ)	Recall [%] (σ)
Normal	99.93 (0.04)	99.93 (0.06)
Rain	99.50 (0.11)	99.52 (0.51)
Construction Site	99.85 (0.17)	98.82 (0.66)
Multipath	99.13 (1.24)	99.52 (0.23)
Maintanance	99.95 (0.08)	99.64 (0.31)
Restart	99.53 (0.46)	99.96 (0.07)
Wind	99.52 (0.32)	100.00 (0.00)
Snow	98.13 (0.07)	89.97 (4.01)
Periodic sway	88.79 (6.01)	91.16 (5.97)

(b) Model using 1 week/60 min.

Class	Precision [%] (σ)	Recall [%] (σ)
Normal	99.88 (0.14)	99.93 (0.04)
Rain	99.23 (0.26)	99.61 (0.28)
Construction Site	99.69 (0.47)	98.97 (0.64)
Multipath	99.75 (0.36)	99.24 (0.56)
Maintanance	99.76 (0.21)	99.88 (0.04)
Restart	99.70 (0.20)	99.80 (0.17)
Wind	99.70 (0.06)	99.98 (0.02)
Snow	96.83 (1.20)	89.42 (3.28)
Periodic sway	86.93 (3.77)	90.91 (1.57)

(c) Model using 3 days/5 min.

Class	Precision [%] (σ)	Recall [%] (σ)
Normal	99.93 (0.02)	99.96 (0.01)
Rain	99.47 (0.14)	99.85 (0.03)
Construction Site	99.82 (0.14)	99.82 (0.19)
Multipath	99.95 (0.08)	99.04 (0.26)
Maintanance	99.95 (0.08)	98.68 (0.36)
Restart	98.79 (0.52)	99.96 (0.07)
Wind	99.77 (0.02)	100.00 (0.00)
Snow	98.13 (0.21)	94.32 (1.46)
Periodic sway	95.72 (1.88)	89.92 (0.76)

(d) Model using 3 days/60 min.

Class	Precision [%] (σ)	Recall [%] (σ)
Normal	99.95 (0.02)	99.97 (0.01)
Rain	99.59 (0.05)	99.76 (0.02)
Construction Site	99.71 (0.19)	99.57 (0.17)
Multipath	99.90 (0.00)	99.43 (0.01)
Maintanance	100.00 (0.00)	99.86 (0.25)
Restart	99.55 (0.17)	100.00 (0.00)
Wind	99.85 (0.14)	99.95 (0.04)
Snow	99.84 (0.12)	94.42 (0.95)
Periodic sway	87.74 (3.18)	95.12 (3.21)

In Tables 5.5, 5.6, 5.7 and 5.8, the confusion matrices for each interesting model are presented. These values are the means of the predictions after three training sessions rounded to integers. As seen in the tables, the most frequent misclassification is Snow and Periodic sway, where it is mixed up with Rain.

Table 5.5: Confusion matrix of a model using historical attenuation with a time span of 1 week and sampling period of 30 minutes.

		Predicted labels								
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
True labels	Normal (1)	8612	0	1	1	0	0	0	0	2
	Rain (2)	1	36285	8	79	0	1	15	26	45
	Construction (3)	0	1	7212	1	0	0	83	2	0
	Multipath (4)	1	45	0	13165	0	0	18	0	0
	Maintenance (5)	0	0	0	0	1381	5	0	0	0
	Restart (6)	0	0	0	0	1	1705	0	0	0
	Wind (7)	0	0	0	0	0	0	27564	0	0
	Snow (8)	0	106	36	36	0	2	18	1483	1
	Periodic Sway (9)	5	30	0	0	0	0	0	0	361

Table 5.6: Confusion matrix of a model using historical attenuation with a time span of 1 week and sampling period of 60 min.

		Predicted labels								
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
True labels	Normal (1)	9635	1	1	1	0	0	1	0	3
	Rain (2)	0	36320	8	32	0	3	17	29	51
	Construction (3)	1	6	7190	0	0	0	46	21	0
	Multipath (4)	5	77	0	13161	0	0	17	0	1
	Maintenance (5)	0	0	0	0	1384	2	0	0	0
	Restart (6)	0	0	0	0	3	1703	0	0	0
	Wind (7)	1	4	0	0	0	0	27559	0	0
	Snow (8)	0	163	14	0	0	0	1	1506	0
	Periodic Sway (9)	4	32	0	0	0	0	0	0	360

Table 5.7: Confusion matrix of a model using historical attenuation with a time span of 3 days and sampling period of 5 min.

		Predicted labels								
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
True labels	Normal (1)	9639	0	1	0	0	0	0	0	2
	Rain (2)	0	36405	7	0	0	3	5	27	14
	Construction (3)	0	0	7285	0	0	0	13	1	0
	Multipath (4)	1	81	0	13102	0	0	45	0	0
	Maintenance (5)	0	0	0	0	1368	18	0	0	0
	Restart (6)	0	0	0	0	1	1705	0	0	0
	Wind (7)	0	0	0	0	0	0	27564	0	0
	Snow (8)	0	83	5	0	0	0	0	1554	0
	Periodic Sway (9)	5	31	1	0	0	0	0	2	357

Table 5.8: Confusion matrix of a model using historical attenuation with a time span of 3 days and sampling period of 60 min.

		Predicted labels								
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
True labels	Normal (1)	9639	0	1	0	0	0	1	0	2
	Rain (2)	0	36371	10	0	0	2	6	28	42
	Construction (3)	0	0	7267	3	0	0	20	6	2
	Multipath (4)	1	57	0	13153	0	1	13	0	3
	Maintenance (5)	0	0	0	0	1384	2	0	0	0
	Restart (6)	0	0	0	0	0	1706	0	0	0
	Wind (7)	0	3	10	0	0	0	27551	0	0
	Snow (8)	0	75	0	10	0	3	0	1556	5
	Periodic Sway (9)	4	15	0	0	0	0	0	0	377

6
Discussion

Here the results presented in Chapter 5 are inspected. Firstly the baseline model results are discussed. It is followed by an analysis of the effects of adding the Periodic sway class and followed by a discussion about the results from adding different types of historical attenuation. Finally, some future research is proposed.

6.1 Selection of baseline model

Looking and the results from Section 5.1, as shown in Tables 5.1 and 5.2, the modified model using six filter layers performed better, in terms of accuracy. Due to the lack of time, the model was only tested once. The results might differ for multiple testing of the model. Even if the results would be the same, the updated model with the added filter layer is better when also taking complexity into account. That is why an additional filter layer was added to the model already used by Ericsson.

Since only one test was made for the baseline model, it is hard to draw any firm conclusions based on this test only. But as explained in Section 2.4 the more filters used the more complex patterns the model can learn. This could be the reason why the modified model achieves a better accuracy. Considering the snow class there is a difference between the two models.

The model with six filter layers was used as a benchmark for the other models. The other models then compared their metric scores and time consumption for fetching and classifying events with this benchmark model. It was also used to compare the impact of adding the historical attenuation feature.

6.2 Effects of adding the Periodic sway class

Looking at the two first results of Table 5.3 the difference in training the baseline model with and without the Periodic sway class can be seen. As expected, only adding the class Periodic sway without adding the historical attenuation did not result in a model with a good F_1-score or recall. The precision is still decent, but this might be because the model did only find 52% of the Periodic sway windows.

The recall for the baseline model trained on data with the Periodic sway class included, as seen in Table 5.3, shows that only 52% of the Periodic sway windows was correctly classified. This means that the rest of the Periodic sway windows were mistaken for another class. This might be because over a single event of 24 hours

the Periodic sway disturbance can look similar to the Rain disturbance, as shown in Figure 1.1.

6.3 The historical attenuation feature

Table 5.3 shows that it can be seen that almost all models using historical attenuation performed better in terms of the F_1 score, recall and precision than the model not using this feature. This was expected since the Periodic sway disturbance is periodic and therefore an input feature covering its periodicity helped the models to classify this disturbance more accurately, as explained in Section 3.1.

One surprising result came from using two weeks of historical attenuation. Since this is the longest time period it was expected that the model trained on such data would perform best, but from the Table 5.3 it looks like the opposite is true. There are several plausible explanations for why this is the case. When using historical data of two weeks, a lot of noise and other disturbances could be included as well, causing the periodicity to be more indistinct. Furthermore, more data results in more hidden convolution layers, as explained in Section 4.2, which could lead to a model that overfits. Since our data set is skewed with little data for Periodic sway this explanation seems plausible.

Looking at Table 5.3, there are four models with high recall, high precision and a low time for fetching data and classification compared to the others. This motivated further investigation of the models using historical attenuation with three days and five minutes sampling period, three days and 60 minutes sampling period, one week and 30 minutes sampling period and finally one week and 60 minutes sampling period.

6.4 Promising models

Table 5.4 summarises the results for the most promising models for all classes. This makes it clear that all models perform well on all classes, but with slightly lower performance for Snow and Periodic sway. This was seen already when adding the Periodic sway class to the baseline model, as discussed in Section 6.3. Further investigating the confusion matrices in Tables 5.5, 5.6, 5.7 and 5.8 it is clear that all of the models mixes up both Periodic sway and Snow with Rain. This does not affect the recall or precision for Rain a noticeably, since the class is much larger than the Snow and Periodic Sway.

6.5 Future work

In order to improve the models, more data is necessary, especially for the Periodic sway class. Ericsson has more data available, but the processing and searching of new events is time consuming. One idea would be to look into artificially generating events to mimic the behaviour of the Periodic sway disturbance. Perhaps a more applicable and reliable solution is to use one of the models presented in this thesis, in

order to investigate the unprocessed data and find more events and by that expand the collected number of Periodic sway events.

To further improve the performance of the microwave network monitoring, more disturbances, if there are any, than the now included classes could be added. In order to find new, unknown disturbances, utilising unsupervised learning algorithms to cluster events could be valuable [15].

7
Conclusion

The results from this thesis show that it is beneficial to add a historical attenuation features as an input to convolutional neural network models, in order to detect periodic disturbances in microwave signals. When using historical attenuation the time span is important and the results implies that more historical data might not always be better. Furthermore, it is shown that to detect periodic disturbances in microwave signals, the sampling period of the historical data can be sparse and still contain enough information to greatly improve the performance of the CNN, in comparison with not using any historical input.

For the Periodic sway class, no model was able to achieve metrics scores similar to those of the other disturbances. But, from the results it is deemed possible to expand the existing model to classify the disturbance caused by sunlight variation, without substantially worsening the current performance.

Bibliography

[1] T. Olausson and V. Sandell, "Disturbance detection and classification in large microwave networks," *Electrical Engineering, Chalmers University of Technology*, 2017.

[2] J. Andersson, P. Berg, J. Hansryd, A. Jacobsson, J. Olsson, and J. Wallin, "Microwave links improve operational rainfall monitoring in gothenburg, sweden," *Proc. CEST*, pp. 1–4, 2017.

[3] J. G. Proakis and M. Salehi, "Digital communications," *McGraw-Hill Higher Education*, 2008.

[4] T. S. Rappaport, "Wireless communications: Principles and practice," *Prentice Hall PTR*, 2002.

[5] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, "Data preprocessing for supervised leaning," *International journal of computer science*, 2006.

[6] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS international transactions on computer science and engineering*, 2006.

[7] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010.

[8] B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," *Third IEEE International Conference on Data Mining*, pp. 435–442, 2003.

[9] S. B. Maind and P. Wankar, "Research paper on basic of artificial neural network," *International Journal on Recent and Innovation Trends in Computing and Communication*, pp. 96–100, 2014.

[10] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[11] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv*, 2015.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.

[13] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, 2017.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, pp. 1929–1958, 2014.

[15] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," *CoRR*, 2018.

DEPARTMENT OF MATHEMATICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY