Predicting Loss of Communication Between Radio Enabled Devices Using Deep Recurrent Neural Networks

JOEL KLINT & OSCAR RYDH
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Predicting Loss of Communication Between Radio Enabled Devices Using Deep Recurrent Neural Networks

Joel Klint & Oscar Rydh
dat13jkl@student.lu.se, psy13ory@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Flavius Gruian & Jonas Mortin

Examiner: Erik Larsson

February 20, 2019

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This thesis investigates the effectiveness of applying recurrent neural networks (RNN) to detect communication errors between radio devices, also known as supervision violations, on imbalanced data. The task is to classify whether a supervision violation is to occur within seven days. The available data is in the form of radio packets, which are being re-sampled and pre-processed such that they can be interpreted by RNNs. The RNNs are trained as both classifiers and generators to enable detection of supervision violations. Using RNNs, an extensive evaluation is made into different pre-processing methods, using multiple test sets and network architectures, applying average precision as metric and precision-recall curves as the main evaluation technique. The results show that it is possible to achieve an average precision of 0.75, and that experimentation with pre-processing parameters along with multiple testsets are needed to ensure a generalised model.

Keywords: Recurrent Neural Networks, Long Short-Term Memory, Predictive Maintenance, Imbalanced Data, Pre-processing

Popular Science Summary

The internet of things revolution is rapidly transforming the world we live in. As these gadgets rely on wireless communication, any disturbance in the environment can cause the device to break down. Using AI technology, there is now a promising solution for predicting communication errors.

More and more of our daily lives are being connected to the internet. For example, the traditionally seen furniture company IKEA is starting to push their smart home solutions with appliances such as connected lights and curtains. This is enabling you to control your home using either your app, or the increasingly popular smart home hubs such as Amazons Alexa or Google Home.

Smart homes has a great reliance on wireless communication. When you say "Alexa, please open my curtains", the signal must travel from the smart hub to the curtain. Should this signal fail to reach it, you could scream all you want without letting any sun in. It is therefore important that the connection between the devices in a smart home is working at all times.

In wireless communication, multiple things affect the ability for a signal to reach its destination. For example, the surrounding environment can destroy the travelling signal (Try using the internet next to a running microwave and see what happens). Other things, such as electric cables could disturb the signal through changes in the magnetic field.

By using AI technology, a new promising solution is able to detect if the environment is disturbing a wireless signal. From this information, you could make changes to your devices such that the connection between them flow better, and thereby increase their functionality.

For example, if the AI detects that the communication between your smart curtain and smart hub will break down, you could take preventative actions such as moving the smart hub to a better location. The AI could also be integrated with a suggestions systems, which could help your smart home work as smoothly as possible. For instance, Alexa could tell you to move the smart hub closer to your devices. Thereby, never again would you be irritated on your smart lights not turning off, as you lie in your bed.

Terminology

,

- **Device** – A wireless device designed for a specific purpose
- **Installation** – A set of devices
- **Central Unit** – A special device, communicating with all devices in an installation
- **Supervision Violation Status** – A status representing that a device is unable to communicate
- **RNN** – Recurrent Neural network
- **LSTM** – Long short-term memory
- **GRU** – Gated Recurrent Unit
- **MLP** – Multilayer Perceptron
- **IoT** – Internet of Things

Acknowledgements

We would like to thank Jonas Mortin for always taking the time to answer questions, help understanding the data and giving much appreciated feedback on our work and this report.

We would also like to thank Flavius Gruian for great discussions on combining the neural network literature with the data provided, as well as the feedback on the report.

Finally, we would like to thank Mikael Nilsson for providing great insights in working with neural networks during a critical period of the thesis.

Joel Klint & Oscar Rydh

Lund, Sweden – 27 January 2019

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data availability and machine learning has helped transform the industrial world in a way much comparable to the 19th century industrial revolution [42]. Data is increasingly gathered by information in Internet of Things (IoT) devices [43]. At the same time, data storage capacity has grown exponentially since the 1980s [20]. This combination has led to a rapid increase in data availability, enabling computer based solutions for problems previously thought only solvable by humans. Computers already beat us in complex games [44] and perform image recognition faster with higher precision [46]. This is thanks to the increased use of neural networks. There are more opportunities for neural networks to create business value for companies today [41]. Therefore, this thesis will attempt apply them to the industrial maintenance domain.

1.1 Background

This thesis originates from a proposition suggested by a global IT company. As a part of their business model, they provide installations with a set of wireless devices to their customers whom expects 100% up time. The devices communicate over radio frequencies with a central unit, which coordinates the installation as a whole. The connection between each device and central unit is critical for system operation, as the occasional loss of communication can undermine an installations functionality, thereby hurting both the customer and company reputation.

There are currently protocols in place for handling these situations, but preventing them is always preferable. It is therefore of high interest to investigate the possibility of predicting when there is a high risk of loss in connectivity between the devices. Should this be possible, with enough lead time, appropriate actions could be taken to avoid the communication errors from taking place completely. This would help the business in several ways. For example, it would enable fewer connection errors, as devices could be maintained before breakdown. Instead of using routine checkups, maintenance could be conducted in such way that only necessary devices would be maintained, enabling a reduction in cost and more efficient use of resources.

To ensure the communication channel between a device and the central unit is functioning at all times, it is continuously tested. The device periodically sends a packet, which the central unit listens for constantly. This period of time is called a *heartbeat*. If the central unit fails to receive any packet from the device for an ~ 20 times longer duration than the heartbeat, the device is considered to enter **supervision violation** status. Once the central unit receives a packet from the device, during its supervision violation status, the device is considered to exit that state. A device in supervision violation therefore implies that the device is unable to communicate. Some possible reasons for a device getting a supervision violation status are disturbances in the radio environment and power failure. During this study, the cause of the supervision violation status is considered unknown and out of scope. Still, events which have predictable supervision violations, such as low battery levels and device tampers, are omitted.

All of the communication done between devices is being monitored and stored. This results in databases filled with millions of radio packets sent within the installations. It is not cost efficient to have employees interpret and extract insights from this large data set. Hence, when keeping track of devices with recurrent problems of supervision violations, mainly the history of previous violations are taken into account. With this data at disposal, an opportunity for a computer based solution to accurately predict future supervision violations has emerged, providing better tools for the organisation.

1.2 Research question

This paper will attempt to predict the probability of a supervision violation occurring. By doing so, preventative actions could be taken bringing a significant value to the company's maintenance process through minimised downtime and maximised cost efficiency. The project will utilise neural networks for prediction. The question therefore needs to be formulated such that a neural network can model it. The following defines the research question.

> *Is it possible to detect a supervision violation within seven days of occurrence?*

This thesis will attempt to predict whether a supervision violation will occur during a time period, without regards to at which point in time it could happen. The supervision violation will be considered to be as likely to appear at the start of the time period, as it is to appear in the middle or end of it. In an attempt to do this, a study detailing how different variables affect predictions will be made. Variables in the study will include different data pre-processing methods and neural network architectures.

1.3 Contributions

This thesis ran for 20 weeks. Each of the authors have contributed equally to the totality of the study.

1.4 Related Work

Using machine learning in fields related to maintenance is not unexplored. In the following sections some works are highlighted.

Predicting Remaining Useful Life

Predicting a supervision violation status is closely related to that of predicting Remaining Useful Life (RUL), since if one can estimate the RUL, preventative actions can be taken prior to any functionality disturbances. On that topic, there are some recent studies. Wu et al. attempts RUL prediction on turbofan engines by using neural networks [47]. It compares different neural networks and show how Long Short-Term Memory (LSTM) neural networks provides significant performance improvement over standard Recurrent Neural Networks (RNN) and Gated Recurret Unit (GRU) models.

PHM data challenge 2015

The PHM data challenge is an annual competition for the Prognostics and Health Management (PHM) Society Conferences. The 2015 edition was focused on fault detection and prognostics [35]. A dataset of system signals for time sequences was provided. Based on this data, the task was to solve problems regarding fault detection. From it, there are especially two papers worth highlighting. Xie et al. presents an interesting feature extraction process which abstracts the data sequences to distinct features [49]. This is combined with ensemble decision trees to solve the problem of fault (supervision violation) detection. Wie Xiao also used a similar feature engineering process for his solution [48].

Predictive maintenance with Neural Network classifiers

Another thesis was just recently published which tried to tackle a similar problem as this project [6]. In this case the goal was to classify time series, and predict needed maintenance. This project used many of the available neural network techniques, and drew the conclusion that it was hard to reach satisfying results. They argued that this was mostly due to the fact that the available data had problems with both missing and imbalanced information. As the data in this thesis also suffer from a heavy imbalance between classes, it has to be addressed.

1.5 Outline

This thesis is organised as follows. Chapter 2 will provide theory regarding fundamental machine learning concepts, neural networks and predictive maintenance. It's designed to provide the knowledge needed to understand the rest of the thesis, and any reader already familiar with these concepts should have no problems jumping straight into the contents of the study. Chapter 3 will introduce the proposed solution for predicting the supervision violation statuses. The actual implementation of the solution is then provided in chapter 4. This is followed by chapter

5, detailing the experimentation setup for evaluating the implemented models. Finally, an investigation of the results, together with a discussion is provided in chapter 6.

Chapter 2

Theory

This chapter will provide an introduction to the theory regarding all techniques used throughout this thesis. It will cover everything from predictive maintenance to how to build and evaluate modern machine learning models. It is intended as an introduction to the domain, as well as background necessary for easier understanding of this thesis.

2.1 Predictive maintenance

Any device that has ever been produced is subject to the laws of entropy. As time passes things will slowly deteriorate into chaos. To avoid such a situation everything has to be maintained. For example, wheels on a car need to be replaced on regular basis, batteries recharged on smart-phones and engines oiled.

Maintenance is a well studied field with many areas. This thesis revolves around **predictive maintenance**. Predictive maintenance attempts to predict when upkeep is needed based on the current state [29], for example by keeping track on wear and tear on system critical parts. In predictive maintenance, the maintainer intend to be informed when some part will start to deteriorate, and put necessary measures in place prior to the breakdown.

Predictive solutions are not the only techniques when handling maintenance. A device would traditionally be maintained in a scheduled or reactive manner. This means that regular check-ups is conducted on the system. Though, there are drawbacks with this approach, mainly increased costs due to unnecessary labour and inefficiently utilised resources. With a good predictive maintenance process, it is possible to maintain systems more cost efficiently than with reactive maintenance [29].

2.2 Machine learning

The term *Machine Learning* has been around for some time. It was originally coined by Arthur Samuel as a paraphrasing regarding parts of his paper *Some Studies in Machine Learning Using the Game of Checkers* written at IBM in 1959 [39]. It is defined as *"a computers ability to learn without being explicitly programmed"*. The subject has since then developed immensely with countless algorithms. During the last couple of years it has received increased attention in media and main-stream culture. One example is AlphaGo which was able to beat world leading players in the board game *Go* [44]. Another example is an algorithm which is able to transfer the art style from one picture to another [17]. Machine learning has the ability to solve all sorts of tasks and creative problems. The following sections will provide theory necessary to understand how these results could be achieved.

2.2.1 Traditional Machine Learning

Machine Learning is a wide field with a flurry of tools and techniques. Each technique is typically designed to solve a certain task by learning from a set of examples. The tasks can be formulated to be solved using one of two techniques: **Regression** or **Classification**.

Which formulation to use depends on the problem. A typical classification problem can be the following. Given a picture of either a cat or a dog, which type of animal does the picture contain? On the other hand, a typical regression example is instead, given a series of temperature measurements, what will the temperature be tomorrow? A classification model thereby attempts to predict whether some input belongs to a predetermined class, while a regression model tries to predict a continuous value given some data.

What is learning?

In machine learning, problems can be learned thanks to parameterised functions. By looking at many examples of matching input/output pairs, a machine learning model can store similarities as the function parameters. The function is defined by a human and its complexity may vary wildly. One common task is to fit a function on set of data points (regression). In that case, the machine learning algorithm would set the parameters (also known as weights) for the function in such way that it fits all data points as well as possible. This can generally be described as optimising a function on an unknown data distribution using a set of provided samples from said distribution [30].

The Learning process

The learning process can be done in multiple ways. Traditional algorithms typically make use of mathematics to gradually better describe the data, such as the K nearest neighbours technique [2]. As the complexity of the task becomes harder, this is no longer possible since the accuracy typically decreases. Instead more complex models make use of examples during training. A large quantity of data points is needed when using examples to build a model. These examples are used as basis when optimising the parameters of a model.

Classification

An example of a classification task is: given an input (eg: a picture), decide on which class it belongs (eg: a cat or a dog). For this task the mathematical model of a logistic regression is commonly used. It is formally defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

where x is any real number. The main reason for using this function is due to the property $\sigma(x) \in [0, 1]$, which can be interpreted as a pseudo probability for the input belonging to a class. See Figure 2.1 for a graph of the function. From the cat/dog example, 1 could be mapped to dog and 0 to cat. This would mean that a model outcome of 0.7 describes a 70% probability of dog and 30% probability of cat.

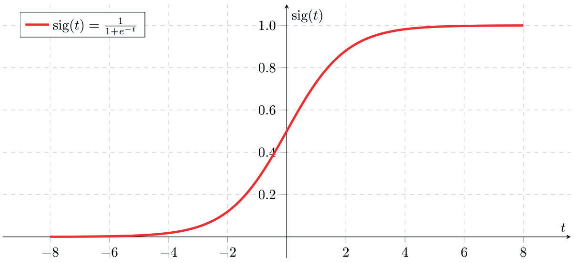


Figure 2.1: The logistic regression function [45]

The formal definition of the logistic regression is not parameterised. Its formulation is extended when applied in machine learning, as seen in Equation 2.2.

$$\sigma(x) = \frac{1}{1 + e^{-(\omega_0 + \omega_1 * x)}} \tag{2.2}$$

In Equation 2.2 ω_0, ω_1 are the variables that are learned in the machine learning process. They are optimised to correctly classify as many examples as possible. The classification model can be generalised to handle multiple input values, such as several pixels in an image, which can be seen in Equation 2.3.

$$\sigma(x_1, x_2, ..., x_n) = \frac{1}{1 + e^{-(\omega_0 + \omega_1 * x_1 + \omega_2 * x_2 + ... + \omega_n * x_n)}} \tag{2.3}$$

The machine learning algorithm would need to learn all the ω values. This thesis uses multiple input values when training machine learning models for classification.

Regression

A regression model maps a set of data points from a distribution, to a function which can be used to generate new data points from the same distribution. In machine learning, it is usually a linear combination of a set of input values x. The most simple example is seen in Equation 2.4.

$$\sigma(x) = \omega_1 * x + \omega_0 \tag{2.4}$$

The machine learning model should find the set of ω values which best describe the provided data points (See Figure 2.2). Comparing this to the temperature example mentioned earlier in this section, each x value could be a temperature at a point in time. A good regression model can be used as a function describing temperature at any time. It is worth noting that a regression model is not bounded by the same properties as the logistic model as $-\infty < \sigma(x) < \infty : \forall \omega, x$

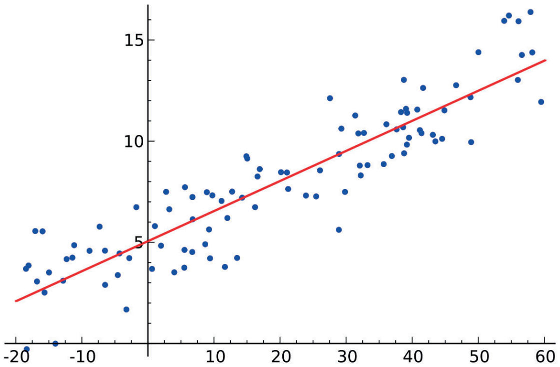


Figure 2.2: Linear Regression [14]

Just as the logistic regression can be extended to a set of input values, so can a regression model. This is called a multivariate logistic regression [5], formally described as

$$\sigma(x_1, x_2, ..., x_n) = \omega_1 * x_1 + \omega_2 * x_2 + ... + \omega_n * x_n + \omega_0 = \sum_{i=1}^{n} \omega_i * x_i + \omega_0 \quad (2.5)$$

Such a model would be able to take into account more than just the temperatures from a set of days. As the temperature today is probably influenced by multiple variables, such as yesterdays temperature, humidity, overcast etc, this could lead to a better prediction. Note that more variables does not necessarily lead to a better prediction. Unrelated variables, such as the average human ear size, would only introduce noise to the model.

2.2.2 Feature Engineering

A machine learning algorithm should only be provided relevant data. Theoretically all data in the world could be relevant, thought this is generally not a good idea as most data will be introduced as irrelevant noise. Distinguishing relevant data from irrelevant is a non-trivial task traditionally, done using **feature engineering**. The goal of feature engineering is to find the variables in data, which provide necessary conditions to train a well performing model.

The process of feature engineering is not standardised as most features are typically domain specific. Identifying animals in a picture requires different data compared to predicting temperature. Traditional statistical methods can be utilised in feature engineering, as well as domain specific experimentation.

2.2.3 Training & testing data

If a model perfectly learned all examples in a dataset, but is bad at predicting on previously unseen data, the model is overfit to the dataset [7]. This means that the model is unable to generalise its learned abilities to a problem as a whole. A comparable example is a student studying for a test by memorising a book word by word. When writing a test, the student is unable to generalise the knowledge when asked to describe the key points.

It is crucial to have a large dataset for a machine learning algorithm to be able to learn and properly adjust its weights. To identify overfitting a model should be evaluated on previously unseen data, a *test set* [36]. Thereby, the data is commonly split into two groups, a *training set* and a *test set* (see Figure 2.3).

The optimal ratio between the two is up for debate. A test set is usually some orders of magnitude smaller than a corresponding training set. A typical ratio is around 80% training data and 20% test data. The training set is used for model training, and the test set for model evaluation. This gives a better indication on model performance in the general case.

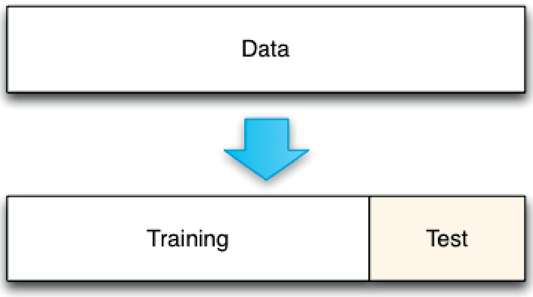


Figure 2.3: Train/test data divisions [26]

2.3 Evaluation Techniques

When searching for an optimal machine learning model, multiple models could be trained [10] and compared. This presents a need for comparison between model performances. This section will cover evaluation techniques in machine learning relevant to this thesis. Techniques for classification and regression models will be presented separately, as they produce results on different forms (see subsection 2.2.1). Evaluation is in essence about comparing model predictions with the corresponding true values. As such, all methods require a set of testing samples and the model predictions for those samples.

2.3.1 Evaluating Classification Models

Predicting image content is a typical classification example. The pixels can be used as input features, and the output formulated as a probability of the image belonging to a specific class (eg: a dog). This probability can be evaluated with several methods.

Precision/Recall/F1-score

This method is based on counting how many samples are both correctly and incorrectly predicted, as belonging or not belonging to a class. Three evaluating scores can be derived based on these counts, *Precision*, *Recall* and *F1-score*. *Precision* describes the true positive prediction rate as seen in Equation 2.6. It can be thought of as the models accuracy in its predictions, as a perfect precision score means that all predictions made were correct. *Recall* on the other hand describe how many of the positive examples the model can find, as seen in Equation 2.7. From these metrics, the *F1-score* is derived as the harmonic mean of precision and recall (see Equation 2.9) [34] [40], describing the models ability to both find and correctly predict positive examples.

$$Precision(model) = \frac{\#True\ positive\ predictions}{\#Positive\ predictions} \tag{2.6}$$

$$Recall(model) = \frac{\#Positive\ predictions}{\#Positive\ samples} \tag{2.7}$$

$$Prediction(x) = \begin{cases} class_0, & \text{if } x \leq 0.5 \\ class_1, & \text{otherwise} \end{cases} ; 0 \leq x \leq 1 \tag{2.8}$$

$$F1(model) = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.9}$$

As the predictions are seen as probabilities (see section 2.2.1), a *decision boundary* has to be introduced. This is a numeric value over which the prediction is regarded as positive, otherwise negative. A typical starting point for the decision boundary is 0,5 as in Equation 2.8. Figure 2.4 provides a visualisation of how each each metric is calculated.

A drawback with this evaluation technique is that it is based on a decision boundary. A decision boundary of $0,1$ will create different scores compared to a decision boundary of $0,9$. The importance of recall vs precision could change in the future, but the evaluation will remain the same as the decision boundary has already been decided.

Precision-Recall Curve and Average Precision

As describe above, one challenge in evaluating classification models is setting the decision boundaries. For this, the precision-recall curve, together with the average precision metric is introduced. Their strength lies in the ability to both show the models performance using multiple decision boundaries, as well as removing the

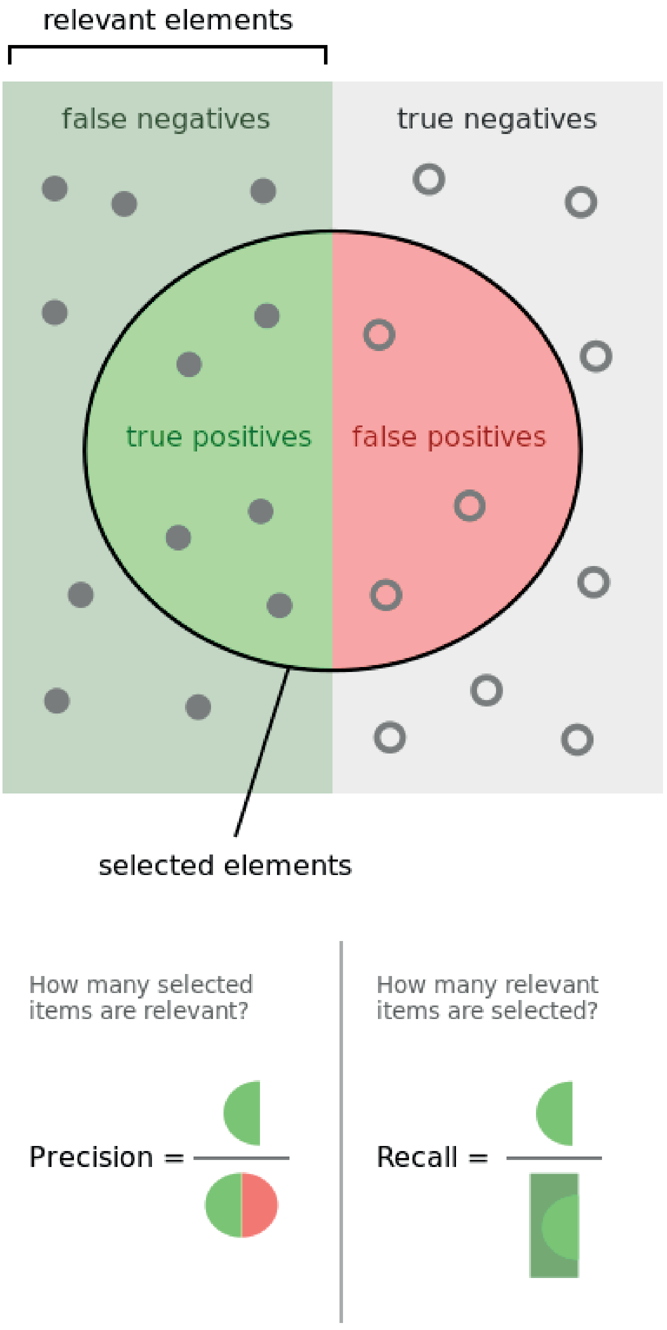


Figure 2.4: Precision and Recall visualisations [13]

fixed decision boundary from the evaluation. This enables more informed decisions when selecting a model.

The precision and recall scores are subject to change as the decision boundary changes. A low threshold can result in higher precision and lower recall, while a high boundary can yield the opposite. It could be the case that no matter the threshold, both precision and recall stays high/low. A way to visualise such a result is using a *precision recall curve* (see Figure 2.5). This curve enables informed decisions regarding the decision boundary, based on the current preferences of recall vs precision. The curve is shown to be an accurate representation of an algorithm's performance when dealing with highly imbalanced datasets [16], a central problem in this thesis.

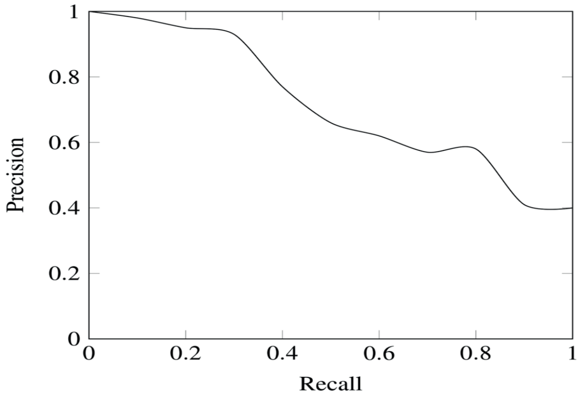


Figure 2.5: A typical precision-recall curve [25].

A precision-recall curve can be summarised as a numeric value, *Average Precision* (AP). This can serve as a comparable metric, similarly to the F1-score. The average precision is calculated as the integral of the precision-recall curve [50], more simply, the area under the curve (see Equation 2.10). Average precision will be used as the main comparable metric in this thesis.

$$AP(model) = \int_0^1 precision(recall) \tag{2.10}$$

2.3.2 Evaluating Regression Models

Regression models are used to predict continuous values compared to binary as with classification models. This requires other techniques when evaluating their performance.

Mean squared error

Mean squared error is calculated as the average of all squared residuals to a linear model [5] (See Figure 2.6). Formally it can be described as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2 \tag{2.11}$$

where n are the number of samples, Y_i the correct value and $\hat{Y_i}$ the predicted. The lower the mean squared error, the better the model fit.

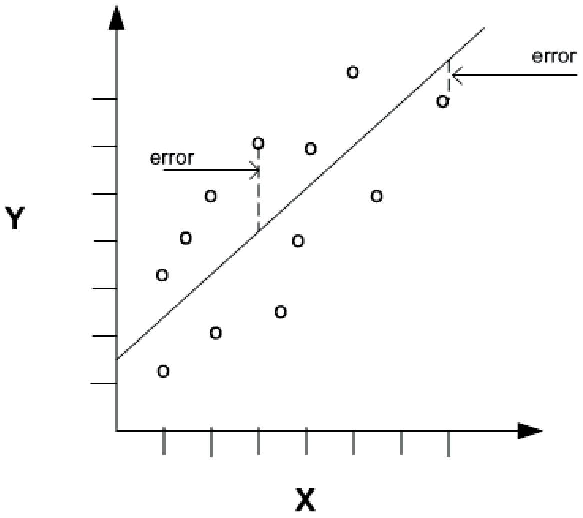


Figure 2.6: Residuals for data points on a trained model [32]

2.4 Neural networks

One of the most active research areas within machine learning deals with neural networks. The concept of neural networks is inspired from the human brain. Neural networks is the main technique used throughout this thesis and this section will introduce the concept.

2.4.1 The Brain Neuron

A human brain is built by billions of neurons. Each neuron is connected to another neuron through synapses. A neuron takes input from connected neurons and sends a signal to other nearby neurons. The input is received through dendrites and the output is sent via the axon. The axon is connected to other neuron's dendrites, hence connecting the brain together as a huge network of neurons communicating with each other, as seen in Figure 2.7 [27] .

The human brain has an ability to learn. This is done through the process of neuroplasticity. Neuroplasticity is the brains ability to change its configuration of

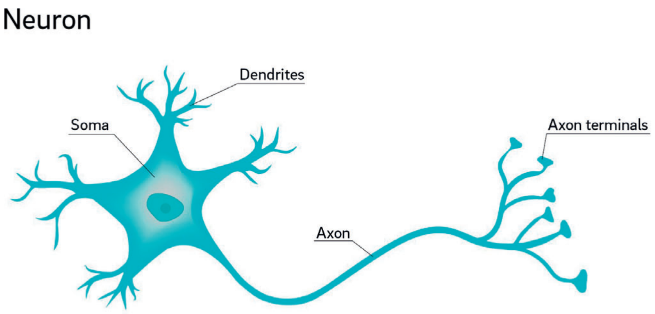


Figure 2.7: A conceptual visualisation of a neuron [3]

neurons and their connections, based on experiences. A human brain constantly changes its neurological connections throughout the day, in order to adapt its owner to the world. One theory which describes how the neurons connect is the Hebbian theory. It states: *"Neurons that firetogether, wire together. Neurons that fire out of sync, fail to link"* [19]. Hence, through the process of neuroplasticity, the brain keeps connections within the brain that are high in activity and rewires areas which do not fire together

2.4.2 The Neural Network Neuron

A neural network in machine learning tries to mimic the learning ability of the human brain by creating a structure similar to it. As there is limited knowledge about the human neuron, a neural network neuron is a simplification of it. A neural network neuron is also known as a perceptron [37].

The classic perceptrion takes, a set of inputs and produces an output. It makes a linear combination on all input, producing a single output. This model was first proposed by McCulloch et al. in the paper *A logical calculus of the ideas immanent in nervous activity* in 1943. [28].

The perceptron does not gain a learning ability by simply aggregating the input as an output. But by associating a weight to each input in the aggregation process, a perceptron can learn how it should prioritise different inputs. The weights are learned in the machine learning process, as described in section 2.2.1. The perceptron is formally described by Equation 2.12 (compare with linear regression in Equation 2.5), where x_i is input i and ω_i is the weight of input i.

$$\sigma(x_1, x_2, ..., x_n) = \omega_1 * x_1 + \omega_2 * x_2 + ... + \omega_n * x_n + \omega_0 = \sum_{i=1}^{n} \omega_i * x_i + \omega_0 \tag{2.12}$$

The perceptron can adjust the weights in the machine learning process, enabling them to be specialised for specific problems.

The perceptron is still missing one part according to the Hebbian theory of *"Neurons that fire together, wire together"*. As the current formulation is just a linear combination of all its inputs, there is no notion of when a neuron should fire. To give the perceptron this ability, the concept of activation functions are introduced.

An activation function is applied on the output of a neuron as a transformation which describes "when" the neuron should fire. Table 2.1 lists some common activation functions.

Putting everything together, Equation 2.13 describes the common formulation of the perceptron, where $f(x)$ is any activation function.

$$f(\sigma(x_1, x_2, ..., x_n)) = f(\sum_{i=1}^{n} \omega_i * x_i + \omega_0) \tag{2.13}$$

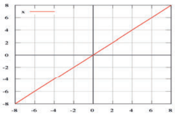
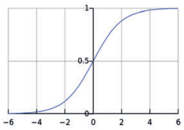
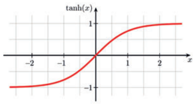
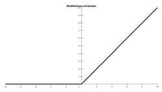
Name	Formulation	Visualisation
Identity	$f(x) = x$	
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	
Tanh	$f(x) = tanh(x)$	
Relu	$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases}$	

Table 2.1: Some common activation functions

2.4.3 Network Types

Throughout the learning process, perceptrons search for ideal weights. After the learning process, the weights are static which means the perceptron will always produce identical output for any set of identical inputs. In order to learn more complex problems, multiple perceptrons are combined in layers. A perceptron in a previous layer is connected to at least one perceptron in the next layer until the final output layer is reached (see figure 2.8). This structure enables more complex non-linear transformations which helps a model learning and solving more difficult tasks. By combining different types of perceptrons, even more complex tasks and problems can be tackled. This thesis will use the following two types of network architectures.

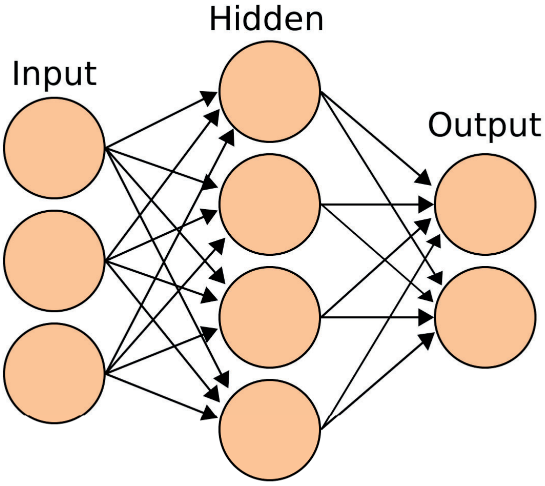


Figure 2.8: A typical neural network [11]

Multilayer Perceptron

The multilayer perceptron (MLP) is the classical neural network. It is also the oldest type, discussed by Frank Rosenblatt in 1961 [38]. An MLP has at least three layers: one input, one hidden and one output layer (see Figure 2.8). Perceptrons in the input layer receive one input each which are propagated throughout the network, until the output is reached. The output of the model is the result from the activation function of each of the output perceptrons. How these should be interpreted is decided by the network architect.

Just as for a traditional machine learning models, the MLP learns from examples. Each perceptron in an MLP has the same formulation as described by Equation 2.13. To give the perceptrons the possibility to cooperate as one model, the input of each node, is the output of the previous layer. An exception is the first layer, which takes features as input. In an MLP, all the perceptrons in a previous layer are connected to the perceptrons in the current layer, also known as fully connected layers.

Each perceptron has a set of weights that it can tweak (see Equation 2.13). As the network sees more examples, each perceptron can update its weights in an attempt to make the entire model capable to solve the task provided better. This leads to a complex network where perceptrons turn on and off (fire or not in Hebbian theory) depending on the input provided and its activation function.

When performing binary classification with an MLP, it would have the same amount of input perceptrons as the amount of features in the input data. One or more hidden layers would follow the input layer, each fully connected to the previous layer. The last layer would consist of a single perceptron with a sigmoid activation function, providing a pseudo probability. For a full visualisation of a common network setup see Figure 2.9

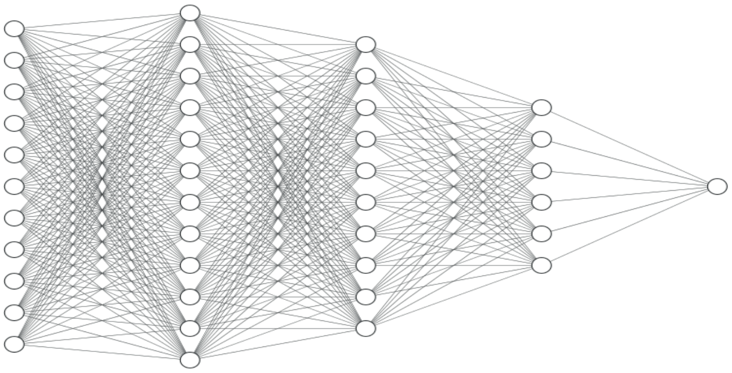


Figure 2.9: A binary classification neural network.

Long Short-Term Memory

The MLP can handle a vast variety of tasks, but it does not understand sequences of data. This means that every set of inputs, is based on a distinct occurrence, and any data provided before that is regarded as independent of the current input. This is problematic when working with time series data. This can be solved by using the Long Short-Term Memory cell, which can handle data sequences.

For some situations, like predictive maintenance, this is problematic as the problem has an inherent dependency on the previous sequence. For handling a sequence of data, the Long Short-Term Memory (LSTM) cell was created.

The LSTM was first presented in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [21]. They presented a specialised cell which has the ability to learn how much of a signal should be passed through a network at different points in time. This gives it a capability to understand how a sequence of data should be interpreted.

The structure of an LSTM cell can be seen in Figure 2.10. Like the perceptron, the cell takes an input x_t and produces an output h_t. The main components of the cell is its memory cell c_t and the three signal gates: the input gate i_t, forget gate f_t and output gate o_t. The memory cell contains the current state of the sequence at a discrete time t. The value of the memory is influenced by the input and forget gate. The input gate decides how much of the input should be added to the current memory, while the forget gate determines how much of the existing memory should be forgotten. The output gate decides how much of the memory and input signal should be sent out at time t.

Note that each gate could be regarded as a perceptron with its own weights. This means that as the LSTM sees more sequences, it can learn how to tweak its gates to let the right amount of signal through to complete the provided task. Also, it is important to know that there are many variations of the LSTM cell which are not discussed here. The interested reader is referred to the comparision by Klaus Greff, et al. [18].

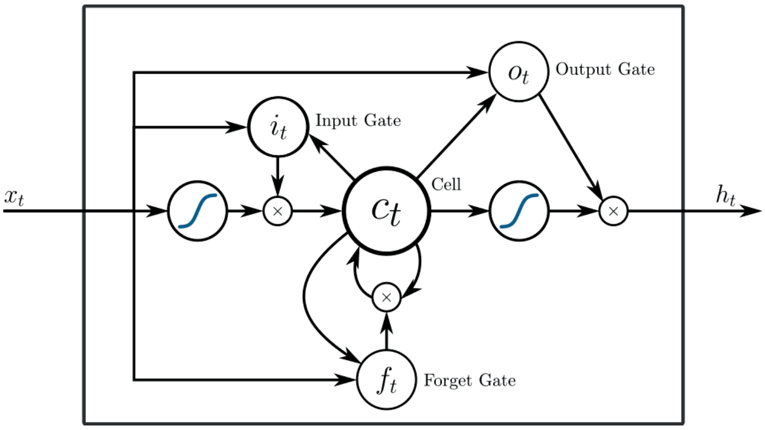


Figure 2.10: An LSTM cell [12]

Chapter 3

Approach

This chapter will cover how the study has been approached. It will cover the conceptual ideas of how neural networks will be utilised to produce a probabilistic prediction of the supervision violation status. It will also cover the method used for the study when deriving model performance.

3.1 Conceptual ideas

Two concepts will be presented, both of which are approaching neural networks differently. These use either direct classification, or generation of features using regression.

3.1.1 Direct classification

A neural network will be trained to classify how likely a supervision violation is to occur within a future time period of seven days. The network will receive a sequence of historic data as input and produce a single probability as its output. This concept is the fastest and most straightforward approach, as one prediction of a sequence with this neural network will immediately produce a probabilistic output. This will be the main concept used throughout this thesis.

3.1.2 Classification through regression

A second neural network will be trained to generate future input data at Δt minutes into the future, which enables a form of "time travelling". A probability representing the likelyhood of the device entering supervision violation status will also be produced. Given input data, this network will be run iteratively to generate data seven days into the future producing multiple supervision violation probabilities at different points in time. By aggregating the probabilities, the classification of a possible supervision violation status will be produced. This concept uses a slower and less straightforward approach, as the neural network must iteratively generate data and probabilities. It does however present a more dynamic solution, as it will be possible to change the desired time period without training a new neural network. This will be the alternative concept throughout this thesis, and not as carefully studied due to time constraints.

3.2 Developing a baseline

The first part aims to develop a baseline. Neural network architecture is the variable in the search for the baseline, the baseline will therefore be developed with static data-preprocessing. The study will start with data investigation to develop a sense regarding sensible default parameters for data-preprocessing. While it is aspired to have a baseline with a high average precision score, the purpose of the baseline is to allow comparisons with various experiments. The baseline will be developed with the direct classification concept (see subsection 3.1.1).

3.3 Deriving model performance

This thesis will also investigate how different data pre-processing parameters and neural network architectures affect average precision. Direct classification (see subsection 3.1.1) will be the primary concept investigated, while the regression approach (see subsection 3.1.2) will also be investigated to a lesser extent due to the time constraint.

The study will be experimentation based. Many models will be trained with varying parameters. The models will then be compared on the average precision metric (see Equation 2.10). In the comparisons a search for patterns will be made, out of which conclusions will be drawn.

As the data is a multivariate time series (see section 4.1), it presents a vast search space which makes an experimentation based approach more challenging. This study will assume independence between all variables to reduce search space and complexity due to time constraints. Parameters will be changed one at a time (see chapter 5 for details regarding experiments). The study will investigate parameters within three categories.

- Neural network architectures
- Data pre-processing
- Probability aggregation

At last, the study will present the top performing models from the experiments.

Chapter 4
Implementation

This chapter will cover technical implementation details of how the investigation has been carried through. It will describe everything from converting raw data, to a model architecture able to make a prediction. More specifically it will focus on the data pre-processing, neural network model architectures and training parameters.

4.1 Data

The dataset enabling this investigation consists of ∼300 million radio packets sent from devices in ∼700 installations. Each installation has up to ∼34 devices, averaging around 7. In an installation there are multiple types of devices, all communicating with the central unit. See Figure 4.1 for a generic installation.

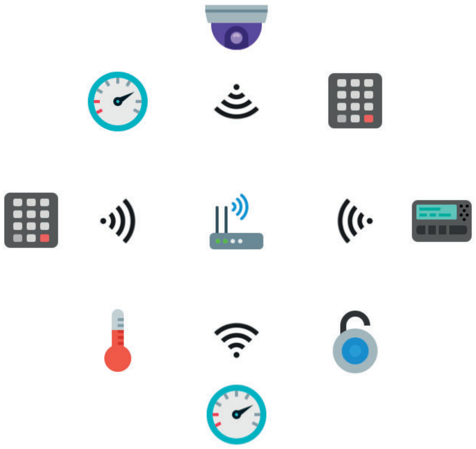


Figure 4.1: The setup for a generic installation. A set of devices (The outer ring) sending radio packets to a central unit. Icons sampled from *http://icons8.com*

Dataset	Spanning time	Packets	Installations	Devices
1	3 months	160 million	296	4115
2	3 + 2 months	292 million	573	7970

Table 4.1: Detailed description of the datasets

All data was divided into two datasets. Dataset 1 contained data from the most problematic installations, in terms of the number of supervision violations. The second dataset extended the first with the purpose of better representing the data distribution (see Figure 4.2). Installations were selected randomly, with the only requirement that any device in the installation, had experienced supervision violation during the last three months. A detailed description of the datasets can be found in Table 4.1. In total, the data was collected over a time period of 5 months.

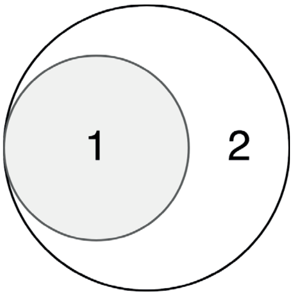


Figure 4.2: Dataset 2 is a superset of dataset 1

4.1.1 Parameters

This section will detail the raw parameters contained in each radio packet of the dataset. The parameters by themselves cannot be used by the neural network models, but are subject to pre-processing, which is described in section 4.2. Due to the radio packets being severely encrypted and time constraints on the thesis, the parameters detailed here were the only ones used when creating the features and input data for the neural networks.

Time

Every radio packet contains the time of which it was sent from a device. The time stamp has a millisecond precision.

Device

The radio packets contain information regarding the device of origin. This study includes the following information about the device:

- Device type

- Device hardware version
- Device software version

The device type is a name for identifying a device, for example "ACCESS POINT 1". Though, in this thesis they will be called either *device type 1* or *device type 2*, as only 2 types were used. Each device also has a specific hardware/software version. The combination of these three attributes describes a single device and their features.

Central unit

Every radio packet has information regarding the central unit which the sending device was communicating with. In an installation there can be several central units, each of which can be of two types. Though, a device will always send its information to a single central unit. The dataset contains information about both the number of central units in an installation, their type and the sending device.

Received signal strength indication

The Received Signal Strength Indication (RSSI) is a common measure in wireless communications. It is defined in the IEEE 802.11 standard as a relative measure of the signal power, at the receiver, on the time of arrival [15]. As this is a measure of the received signal strength, it is self-evident that it will decrease as distance between the device and central unit increases. The RSSI measure is relative, whereby the maximum and minimum value can be configured by the manufacturer. The maximum and minimum RSSI is therefore known beforehand. Also, in the devices provided for this study, there were two antennas. One measured the values horizontally, the other vertically.

The radio packets in this dataset have the devices as source and the central unit as destination. The central units are thereby the receivers. The RSSI value is therefore naturally measured as the signal strength when the radio packet is received by the central unit.

Link quality indicator

The Link Quality Indicator (LQI) is a common measure of the quality of a received packet in wireless communications. It is not related to RSSI, as a packet with low strength can still have high quality. LQI is a relative measure of the link quality, where a lower value indicates a better link quality. The IEEE standard 802.15.4 defines the algorithm to calculate it, though the implementation is left to each vendor [1]. Just as for the RSSI values, the LQI is measured both horizontally and vertically.

Background received signal strength indication

Each central unit in an installation measures its background RSSI values. This is gathered independently of the amount of received radio packets. The value represents the overall signal strength of the surrounding environment. More specifically, the more radio noise in an installation the higher the background RSSI. The available values are the average daily measuring, for each installation.

4.2 Data pre-processing

The raw data had to be pre-processed for several reasons. For example, a neural network only accepts numerical values, while the data contained strings such as the type of a device. Another reason was to enrich the data by e.g. deriving new features through feature engineering. Lastly, using pre-processing the data could be aggregate to shorten the training times of the neural networks.

The time-series aware neural networks used in this paper expects a two-dimensional input. One dimension contains features, while the other represents time. This enables the network to note the changes in feature values over time.

The data pre-processing was done in seven steps, all of which will be explained in greater detail below.

1. Transforming continuous time to discrete time
2. Deriving features
3. Handling missing values
4. Creating samples
5. Labelling samples
6. Normalising features
7. Filtering samples

4.2.1 Transforming continuous time to discrete time

The initial step of pre-processing was to make the time discrete. This is for several reasons. One is to make the problem easier to comprehend, as the time between two packets varies greatly. The other reason is to be able to predict a supervision violation at known times in the future. Without defining a discrete time axis, it would be more difficulty to derive specific future points in time.

To make time discrete, this thesis uses a concept of *time bins*. A time bin T_i starts at time t_i and is stretched d minutes into the future, according to Equation 4.1.

$$T_i = [t_i, t_i + d) \tag{4.1}$$

A time bin T_i holds all the features, derived from the received data (as described in subsection 4.2.2) in the interval defined for the bin. The continuous time series T_s is then made discreet by defining a set of n time bins T_i, such that all data is explicitly in one time bin:

$$\begin{aligned} \cup_{i=0}^{n} T_i &= T_s \\ \cap_{i=0}^{n} T_i &= \emptyset \end{aligned} \tag{4.2}$$

A representation of the time transformation can be found in Figure 4.4. See Figure 4.5 for a visualisation of the time dimension.

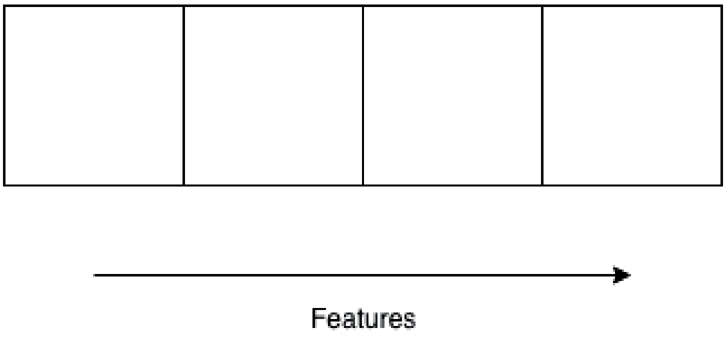


Figure 4.3: The feature dimension of the input

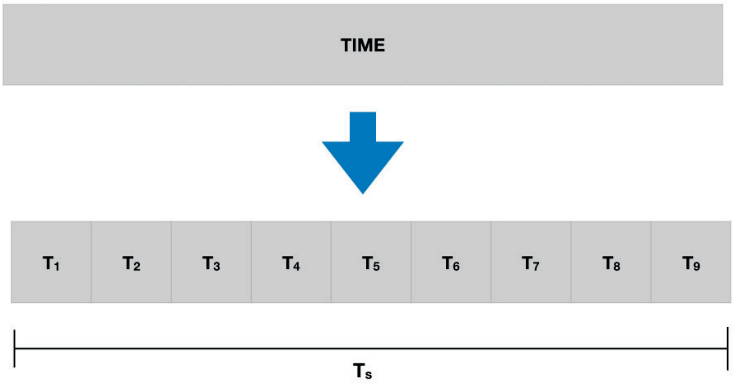


Figure 4.4: A visualisation on how time is made into a set of discrete time bins. Each bin itself contains the features described in subsection 4.2.2

4.2.2 Feature Engineering

Before this step, every time bin consists of zero or more radio packets. But, the intention was to have a time bin to be defined by a set of features. Therefore, a function was created for every feature, which aggregate an arbitrary amount of radio packets within a time bin into a distinct feature value. Several functions exist which enable this aggregation. Some examples are mean, median, combinations of minimum and maximum values, etc. The feasibility of the function differs between features, and several features can be defined for one set of data points, e.g. min, max and mean; each of which could increase the information provided to the neural network. For an overview of the chosen aggregation method for each feature see Table 4.2.

The features are of two classes. The first class are features which change over time. The second class is constant meta data about the installation and/or device type. All features extracted from the time series are the following:

1. Horisontal RSSI
2. Vertical RSSI
3. Horisontal LQI
4. Vertical LQI
5. Background RSSI
6. Number of missing radio packets
7. Number of supervision violations for other devices in the same installation

The features that are constant meta data about the installation and/or device are the following.

1. Central unit type 1
2. Central unit type 2
3. Multiple central units in the installation

RSSI/LQI

The RSSI and LQI features were derived by taking the average of each measurement for all radio packets within a time bin. Formally this can be derived as

$$F_{average}(x_i) = \frac{1}{n}\sum_{j=0}^{n} x_{i,j} \tag{4.3}$$

where i is the time bin and x_i is all the values for a specific feature within that time bin.

Also, it might be that no radio packets were received at a central unit within a given time bin. To mark these events, a flag was added as a feature for each RSSI and LQI reading (Background RSSI excluded). This indicates if a time bin has a true RSSI/LQI value or not.

Number of missing Packets

As noted in the introduction, each device in an installation is pre-programmed to send a signal to the central unit at specific time intervals, also known as a *heartbeat*. This is to let the system know that a device still has contact to the central unit, hence being monitored. If no radio packets are received within the specified interval, the communication link with the device is considered broken. Once a packet is received from the device, the timer is reset.

The duration between packets was used to derive a new feature, the number of missing packets. If no packet had been received within the heartbeat time, it was considered missing. Consecutive missing packets were considered for every additional heartbeat time that passed, until a packet was actually received. The feature added to each time bin was then the accumulated amount of missing packets, based on the last one received.

Installation supervision

As noted in section 4.1, an installation typically contains multiple devices. Just as any other device, these are able to loose contact with the central unit, thereby entering supervision violation status. In the pre-processing, the number of devices in the entire installation entering supervision violation status, within a time bin, was added as a feature.

Central unit information

There are two distinct types of the central units, 1 and 2. Also, some installations have more than one central unit. For this project the communication between multiple central units are disregarded from the models perspective. Instead, three features were derived from the data. These describe if central unit 1 and/or 2 is present, as well as if there are several central units in the installation.

It can be discussed how these constant features should be used in a deep neural network. This paper chose to insert all constant features into the time series data, resulting in every time bin containing the features in Table 4.2.

The features define the second axis of the input data. With these two axes, it is possible to feed the data to the neural network. See 4.5 for data at this stage of the pre-processing.

4.2.3 Impute missing data

In some time bins, there might be no packets at all. This is, for example, the case right before a supervision violation, as a lack of radio data is a precondition for its possibility. Since the neural network does not accept null values, the missing data needs to be substituted with some other numerical value. This operation is known as imputing.

Two different solutions on imputing values are Last Observation Carried Forward (LOCF) and Next Observation Carried Backward (NOCB). These are often used in conjunction with uni-variate time series analysis [31]. Though there are more sophisticated techniques for multivariate time series analysis available [23],

Feature	**Id**	**Type**	**Value**	**Computation**
RSSI Horizontal	RH	Time series	Numeric	Averaged
True RSSI Horizontal	TRH	Time series	Boolean	-
RSSI Vertical	RV	Time series	Numeric	Averaged
True RSSI Vertical	TRV	Time series	Boolean	-
LQI Horizontal	LH	Time series	Numeric	Averaged
True LQI Horizontal	TLH	Time series	Boolean	-
LQI Vertical	LV	Time series	Numeric	Averaged
True LQI Vertical	TLV	Time series	Boolean	-
RSSI Background	RB	Time series	Numeric	Averaged
Missing packets	MP	Time series	Numeric	Summation
Installation supervision	IS	Time series	Numeric	Summation
Central unit type 1	CU1	Constant	Boolean	-
Centra unit type 2	CU2	Constant	Boolean	-
Multiple central units	MCU	Constant	Boolean	-

Table 4.2: A list of all features contained in each time bin.

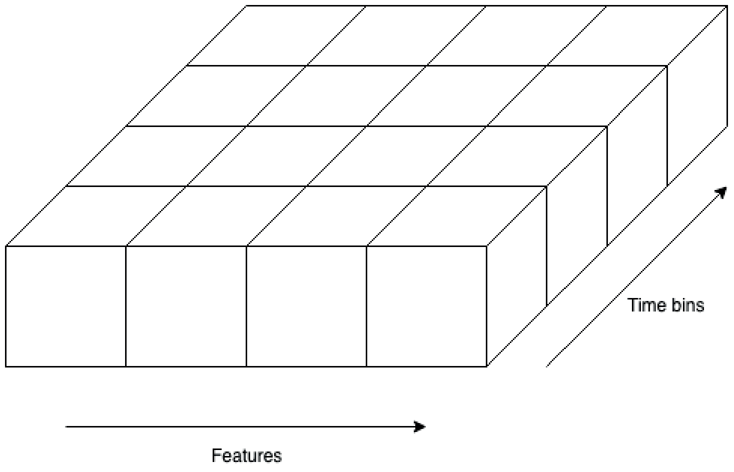


Figure 4.5: The time and features dimensions of the input

this paper chose to utilise the univariate techniques on each numeric feature. Therefore, imputation was done by first applying LOCF, and then a NOCB in case the time series initial value was null. The imputation was done with no consideration of feature correlations, due complexity reduction and the time constraints of the thesis.

4.2.4 Sample data

When all time bins had been defined, along with every feature re-sampled into each time bin, the data was again sampled in order to create learning examples. A sample is defined as a fixed amount of consecutive time bins, which decides the preceding time-frame the model should take into account when predicting a supervision violation. The amount of time bins in a sample, and their size, is variable and was used as a parameter to the model. See Figure 4.6 for a visualisation of the finished pre-processed data.

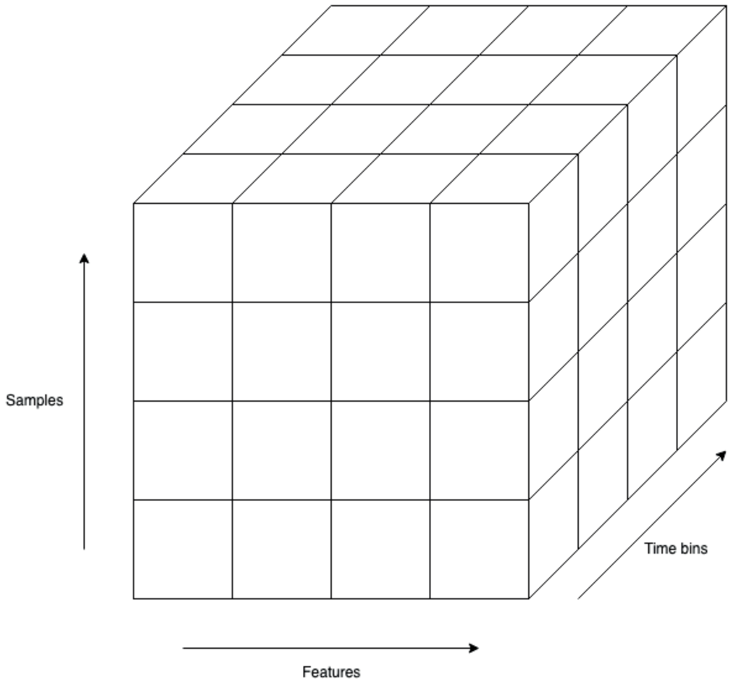


Figure 4.6: The samples, time and features dimensions of the input

4.2.5 Assigning labels

As a part of training a neural network for classification and regression problems, data has to be provided labels to which the model output can be compared. These labels were created as a part of the pre-processing. One type of label was created for each type of model. For a full comparison of sampling and labelling see Table 4.3

In the case of classification, the label was based on how far into the future a supervision violation was to be found. In this work the limit is set to seven days. This enables a model which is accurate on a longer period of time, thereby providing an administrator time to act on a possible supervision violations. The label was set to true/false if **any** supervision violation was reported within 7 days from this moment.

When designing labels for a regression model it gets a bit more complicated, due to the fact that the it is designed to generate new future features based on the trained regression functions. When training the regression functions, the next set of time bins were used as labels, one for each feature. Though, note that the supervision violation status for the specific device analysed is not a feature in Table 4.2. This becomes a bit problematic, as the model has no ability of generating a supervision violation. Therefore a separate feature was added, unique to the regression model, which indicates if a supervision violation happened (true/false) in a **specific** time bin. Thereby, the regression model is able to derive a pseudo probability for a supervision violation status in each time bin.

Model Type	Sample Content	Label
Classification	Features * Time Bins	$\begin{cases} 1, & \text{if S <= 7 days} \\ 0, & \text{otherwise} \end{cases}$
Regression	Features * Time Bins	$\mathrm{F(i)}_{j+x} + \mathrm{S}_{j+x}$

Table 4.3: An overview of the labels for each model, where i is a feature, j is the last time bin in a sample and x is a time bin at a later point in time. F: Feature value, S: Supervision violation.

4.2.6 Normalisation

It is a common practice to normalise data prior to feeding it to a machine learning model. By normalising, the values for all inputs are centered between zero and one. There are several reasons for this. The main reason is that the optimal model converges much faster on normalised data [22]. Normalisation can be conducted using multiple techniques [33]. In this thesis the min-max technique was used individually along each feature. The technique applies the minimum and the maximum value of a feature, creating a function which maps the features between zero and one while still retaining the relative differences between each value.

4.2.7 Filtering

All samples needed to be approved by a filter mechanism. This is to make sure all input data have a certain quality. Remember that a sample is defined by a sequence of time bins, which in turn is defined by a set of features (see Figure 4.3 - Figure 4.6). Suppose that 100% of the time bins in a sample is during a supervision violation status. In that case, there is only synthetic data to base a prediction on, since no radio packets where received, thereby having only imputed features. Having only imputed data is not desirable during training, as all features will have a constant value in all time bins of a sample. This does not only introduce noise, but might also bias the model to look for long flat sequences values, potentially correlating them to a supervision-/lack of supervision violation, depending on the label of the sample.

A threshold was introduced which discards samples having more than a number of time bins with only imputed data. The threshold is a number, and should be less than the amount of bins per sample to have any effect. Depending on the level of data quality desired, the threshold can be adjusted accordingly. This allows for experiments to analyse the effect of the threshold on the accuracy of the predictions. An example of how the number of missing packets are distributed among the samples, which was used when designing experiments, can be found in Figure 4.7.

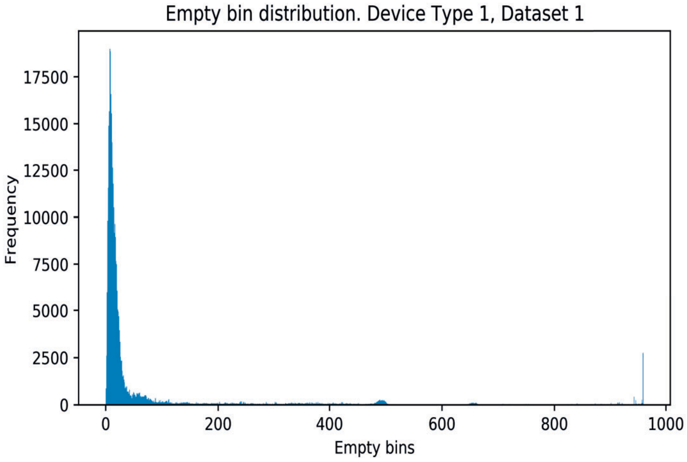


Figure 4.7: The number of missing packets per generated sample for one of the devices in the dataset. Note that most of the data has less than 100 missing packets, and a spike with samples containing almost no packets.

4.3 Dataset imbalance

One of the great challenges of this project was the heavy imbalance of the dataset. Out of all the statuses in a device, only 1.4% are supervision violations. When resampling the data according to section 4.2, the final dataset contained about 15 000 samples with supervisions, compared to 900 000 without, creating a supervision- to no supervision violation ratio of about 1.7%. This makes detecting the violations even more difficult, as imbalanced datasets are notorious in producing suboptimal results [8].

To combat this challenge, this paper opts to the technique of random resampling of the supervision violation samples, more specifically, oversampling [4]. By oversampling the supervision violations, the model is given the same samples of violations multiple times, increasing the balance in the data. Without oversampling, it is likely that the model would only predict no supervision violation statuses, as it would be correct 99% of the time. Such a behaviour was indicated during initial experiments. Further experiments showed that a oversampling ratio 40% produced the best results, as seen in Table 6.1. This means that in the dataset, the supervision violation samples where randomly resampled such that 40% of all the training data where of that type.

4.4 Models

This section will cover the neural network models design and implementation by this paper.

4.4.1 Model designs

Both model architectures specified below are attempts at answering the same question, as mentioned in chapter 3. They expect the same two-dimensional input data, where the two axes are *features* and *time* (as discussed in section 4.2). No preexisting model definitions has been used, as all models are defined by this paper and has been trained with the Keras library [9].

Classification

The classification model is designed as a recurrent neural network in the form an LSTM layer, which in turn is passed to one or more fully connected layers that has a single output node (see Figure 4.8 for a simple overview). The output node has a sigmoid activation function, and produces a prediction of the device receiving a supervision violation status within seven days between 0 and 1; where 1 means 100% probable and 0 means 0%. It uses binary cross entropy as loss function and Adam [24] as optimisation algorithm. The neural network is trained to predict for a predetermined time interval, and once trained, that time interval cannot be changed. To make a prediction for another time interval, for example ten days, the model would have to be retrained on that duration.

Regression

The regression model is the more complex, but also more dynamic approach taken by this paper. It consists of a recurrent neural network based on LSTM and MLP nodes, constructed much like in the classification network. The difference is the output having O_n nodes as described by Equation 4.4, where F_n is generating features and 1 is added for generating probabilities of supervision violations occurring.

$$O_n = (F_n + 1) \tag{4.4}$$

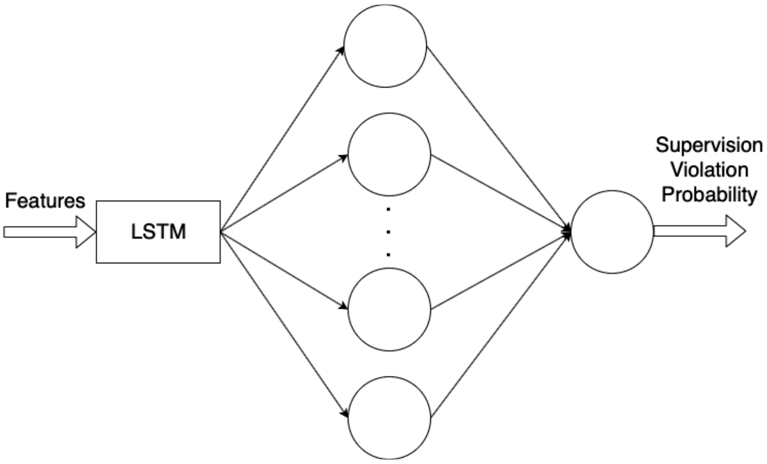


Figure 4.8: A conceptual model of the structure of a direct classification model. The model starts with one or more layers with multiple LSTM nodes, followed one fully connected MLP layer. These are then gathered to a single output node.

The output nodes are setup in groups, such that all features in one time bin are grouped together, and every supervision violation probability is grouped one by one for each time bin. This produces Og_n output groups as described by Equation 4.5, where m is the amount of future time bins to predict (See Figure 4.9). The outputs are grouped so they can have separate loss functions. Separate loss functions are essential as the supervision violation node uses a different activation function compared the ones generating features.

$$Og_n = 2m \tag{4.5}$$

The future feature output nodes uses a linear activation function, while the supervision violation probability output nodes uses a sigmoid activation function. The feature generation nodes uses mean squared error loss functions, while the supervision violation generation uses binary cross entropy loss functions. All losses are summed to create a single unified loss, which is then optimised using the Adam optimisation algorithm.

4.5 Hardware

All neural networks for this thesis were trained on two computers. Each computer's specifications can be found in Table 4.4. One computer trained the classification models while the other trained the regression models.

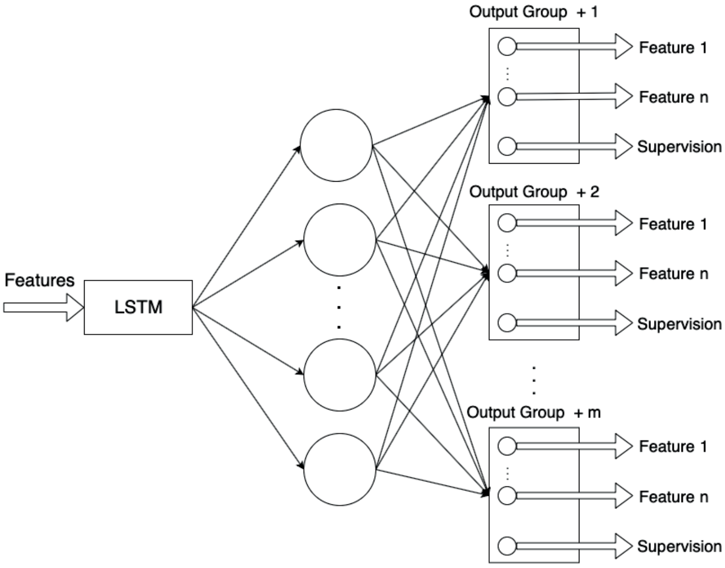


Figure 4.9: A conceptual model of the structure of a regression model. The model starts with one or more layers with multiple LSTM nodes, followed by a fully connected MLP layer. These are then grouped into separate outputs nodes for each future time bin, each node representing either a feature or a supervision status.

GPU	CPU	RAM
Nvidia GeForce GTX 1080	Intel i7, Gen 3, 8 Core, 3.60GHz	64 Gb
Nvidia GeForce GTX 970	Intel i7, Gen 2, 8 Core, 3.40GHz	16 Gb

Table 4.4: Specification of the hardware used during the training of the neural networks

Chapter 5

Experimentation Setup

This chapter will detail the approach taken by the study attempting to derive average precision scores when predicting supervision violation. The goal of the study was to compare different data pre-processing parameters, neural network architectures and feature combinations to study their impact on average precision. A comparison between the two conceptual ideas (see section 3.1) were also initiated during the remaining time of the thesis.

5.1 Available Configurations

An unlimited range of combinations between parameters are available. There are two conceptual ideas, **direct classification** and *classification through regression.* Each conceptual model has unlimited *neural network architectures* with different combinations of layers and node types. Further on, each architecture have multiple *hyper parameters* which can be tweaked. Thereby, the neural networks presents a vast search space.

Data pre-processing further expands the search domain as multiple techniques are used when preparing a dataset (see section 4.2). There are multiple *device types* in the data. The search can be performed for all device types together, or per device type which presents a new dimension in the search domain. *Bin size* affect data granularity and *bins per sample* that decide how many bins defines a sample. Between the two, there are many combinations that varies the data granularity and sample time length. *Imputation* presents a new dimension as it can be performed either on a sample or the full time series. The filter of *allowed empty time bins* affects data quality and has a finite amount of possible values between 0 and total *#bins per sample* (see Figure 4.7). Finally, different combinations of *features* can be used.

All together, the search space has many dimensions with an unlimited amount of configurations, so a structured method is needed. An overview of the search space can be found in Table 5.1.

5.2 Experiments

The study was to explore as broad an area as possible in the search domain. As mentioned in section 3.3, the study assumed independence between all dimensions.

Search dimension	**Domain**
Conceptual idea	Neural network design
Architecture	Neural network design
Hyper parameters	Neural network design
Bins per sample	Data pre-processing
Bin size	Data pre-processing
Imputation scope	Data pre-processing
Allowed empty time bins	Data pre-processing
Feature engineering	Data pre-processing
Device type	Data selection

Table 5.1: Each of the possible configuration variables available for the model, together with their impacted domain

Any experiment locked all dimensions but one which was altered. This lead to a fair comparison on the causal effects of each change. Chapter 6 will deal with the setup for each experiment and the data used.

All experiments were evaluated with average precision as metric (see Equation 2.10). This was because the dataset had a heavy imbalance between the classes. As mentioned in section 2.3.1, average precision is a good metric when facing this problem. By definition (see Equation 2.10), this led to the production of a precision recall curve for every experiment. The precision recall curve enables a later informed decision to be made regarding decision boundaries based on the relative importance between recall and precision.

General Experiment Setup

Each experiment is defined by training six models. This was because there was three test sets for two device types. Multiple test sets were used to identify how general a model was trained as there was a heavy imbalance between the classes in the data. Every test/train split was stratified on installations, the only criteria was that there had to be at least one sample containing a supervision violation in each test set.

Model Architectures

The first set of experiments where designed to look into the different ways of modelling the neural network architectures. A set of different models were created, trained and evaluated. Each model followed the conceptual model described in section 4.4. Still, every setup was distinctively different between the architectures in terms of the number layers of each type (LSTM/MLP), as well as the amount of nodes in each layer. Finally, it was decided that no extensive experiments were to be made regarding the neural network hyper parameters (learning rate, node parameters, etc), since initial experiments did not show any significant performance increases, as well as time constraints did not allow for it.

Data pre-processing parameters

As noted in section 4.2 multiple steps were conducted when transforming the data to the format such that a neural network can interpret it. Many of these are configurable and enumerated in Table 5.1 as *data pre-processing*. For each parameter a set of experiments were designed, tweaking the value significantly and noting the impact on average precision. To allow a fair comparison between these experiments, the same neural network architecture was used throughout them. The chosen model was based on the average precision of the model architecture experiments, where a trade off between model performance and training time was taken into account.

Feature Combinations

Neural networks are often considered as black boxes in terms of explanation of the causal relations between some given features due to the models nonlinearity. This often makes it hard to explain their performance impacts for stakeholders, especially if there is a lack of understanding on neural networks inner technical workings. Therefore, experiments were designed to investigate the average precision for each of the domain specific features. In each experiment, the features of one domain was removed and the average precision was measured. Experiments were also designed to individually investigate the impact of the different radio environment descriptors.

Imputation scope

Even though imputation was part of data pre-processing, it was decided to conduct more extensive experiments on the parameter. The main reason for this was its extensive impact on the formulation of a sample for the network, as an imputation of the full time series would not exclusively result in the same data as when imputing each sample. This is true in the cases where a sample would start with empty bins. If the full time series was imputed it would be forward filled with the previous value, while if each sample was imputed it would be backward filled with the next value. The experiments were therefore designed to investigate the significance of these options.

Datasets

A detailed description of the data can be found in section 4.1. A comparison was done between the datasets by training models with identical neural network architectures and data pre-processing parameters, the only change where in the underlying data.

Chapter 6
Evaluation

This chapter will detail all experiments and their results. It will also draw conclusions regarding how variables affect average precision, based on those results. It will also present the best models based on average precision before a discussion will be held.

6.1 Experiment Results

This thesis performed an experimentation based study, attempting to derive how a set of variables affect average precision scores. Section 5.2 presents an overview of all experiments.

This section will detail each experiment and present it's results together with a brief discussion. All results presented are average precision scores. When combining results for each device type, it is shown as the mean of all average precision scores.

6.1.1 Neural network architectures

An initial search was made with regards to different model architectures, using various random oversampling rates. These experiments differed from the presented evaluation technique in section 5.2, as only device type 1 was used with one testset, due to time constraints. Undersampling was not investigated to a large extent, as early experiments gave low average precision scores. The initial search varied the widths and depths of architectures, with three different oversampling rates. The results can be seen in Table 6.1.

The results showed that shallow architectures perform better than deeper. They also indicate that higher oversampling rates lead to better average precision scores.

The reason why deeper neural networks gives lower average precision scores could be multiple. One indication is that the models start to overfit the training data, leading to a worse generalisation as described in theory in subsection 2.2.3. This means that the data has a complexity which can generalised by a relatively small network.

The results also showed that higher oversampling rates lead to higher average precision scores which acts as evidence for the need of a balanced dataset when

	Oversampling rate			
Architecture	None	20 %	**40 %**	Average
1 x 32 LSTM + 64 MLP	0,256	0,432	**0,426**	0,371
1 x 64 LSTM + 128 MLP	0,227	0,362	**0,319**	0,303
1 x 128 LSTM + 256 MLP	0,345	0,330	**0,388**	0,354
2 x 32 LSTM + 64 MLP	**0,393**	**0,434**	**0,476**	**0,434**
2 x 64 LSTM + 128 MLP	0,314	0,293	**0,458**	0,355
2 x 128 LSTM + 256 MLP	0,113	0,379	**0,193**	0,292
3 x 32 LSTM + 64 MLP	0,147	0,173	**0,396**	0,239
3 x 64 LSTM + 128 MLP	0,334	0,074	**0,385**	0,264
4 x 32 LSTM + 64 MLP	0,131	0,060	**0,080**	0,091
4 x 64 LSTM + 128 MLP	0,078	0,064	**0,050**	0,064
5 x 32 LSTM + 64 MLP	0,162	0,050	**0,050**	0,088
Average	0,227	0,241	**0,293**	

Table 6.1: The average precision of each architecture tested in the initial architecture search, with the top score highlighted. The experiments were based on device type 1. A multiplier indicates amount of layers.

training. As described in section 4.3 the initial supervision violation rate was around 1.7%. This leads to a model seeing the same supervision violation sample about 10-20 times, depending on the chosen oversample rate. As the model sees the same data multiple times, there is an increased risk of overfiting. The results seem to indicate that this could be the case, as the deeper architectures performed better with no oversampling. The results show that when applying random oversampling, it is preferable to stick with more shallow architectures.

Based on the results in Table 6.1, a more detailed investigation into the shallower architectures was made. From this point onward, the experimentation setup presented in section 5.2 was used. As seen in Table 6.2 each device type prefer different network architectures which indicates that the architecture is sensitive to device type. Device type 1 has a session mode in it's communication in which the frequency of packets is much higher, but as time is resampled during data pre-processing, this should not affect the prepared datasets.

Finally, the results in Table 6.2 show a high variance between testsets. For example, the difference between average precision of each individual testset can differ with as much as 0.6 (see the highlighted model for device type 1 in Table 6.2). As it is desired to have models which performs well in the general case, this shows the need of having multiple testsets during evaluation. Without it, average precision scores can not be trusted.

To allow further experiments, a single baseline model was chosen by looking at the mean average precision between both device types (1 x 6 LSTM + 32 MLP as seen in Table 6.2).

Device type 1

Architecture	Test 1	Test 2	Test 3	Average
1 x 6 LSTM + 32 MLP	0,525	0,424	0,056	0,335
1 x 64 LSTM + 128 MLP	0,365	0,405	0,080	0,283
2 x 25 LSTM + 32 MLP	0,358	0,436	0,111	0,301
2 x 32 LSTM + 64 MLP	0,234	0,245	0,036	0,172
2 x 64 LSTM + 128 MLP	**0,652**	**0,382**	**0,027**	**0,354**

Device type 2

Architecture	Test 1	Test 2	Test 3	Average
1 x 6 LSTM + 32 MLP	0,688	0,296	0,362	0,448
1 x 64 LSTM + 128 MLP	**0,775**	**0,327**	**0,245**	**0,449**
2 x 25 LSTM + 32 MLP	0,536	0,328	0,376	0,413
2 x 32 LSTM + 64 MLP	0,417	0,153	0,047	0,206
2 x 64 LSTM + 128 MLP	0,035	0,465	0,161	0,221

Architecture	Mean average
1 x 6 LSTM + 32 MLP	**0,392**
1 x 64 LSTM + 128 MLP	0,366
2 x 25 LSTM + 32 MLP	0,357
2 x 32 LSTM + 64 MLP	0,189
2 x 64 LSTM + 128 MLP	0,287

Table 6.2: The average precision for each neural network architecture per device type, with top scores highlighted.

6.1.2 Datasets

As noted in section 4.1, 2 datasets were created throughout this thesis. The first dataset contained the most problematic installations in terms of supervision violations, while the second extended it by adding installations at random. Table 6.3 shows a comparison of the datasets with the baseline architecture from subsection 6.1.1. From the results, it is clear that more data enables a better model. This is to be expected, as the more relevant information available to a neural network, the better it should be able learn, as seen in subsection 2.2.3.

Device type 1

Dataset	Test 1	Test 2	Test 3	Average
1	0,344	0,291	0,184	0,273
2	**0,162**	**0,316**	**0,437**	**0,305**

Device type 2

Dataset	Test 1	Test 2	Test 3	Average
1	0,587	0,072	0,384	0,348
2	**0,619**	**0,312**	**0,267**	**0,399**

Dataset	Mean average
1	0,310
2	**0,352**

Table 6.3: The average precision for each of the datasets on each device type, with top scores highlighted.

6.1.3 Maximum empty bins

An experiment regarding the maximum amount of empty time bins in each sample was conducted, and any samples containing more empty bins were disregarded. During these experiments, the amount of bins in each sample was 960 (ten days of data). The experiment included three different thresholds of increasing size and the results can be seen in Table 6.4. The average precision scores indicate that the more allowed empty bins in a sample, the better the performance. These results seems to indicate that there is not only information in the available data, but also from the lack of it.

Investigating this further, an analysis was made on the distribution of missing bins in each samples, as seen in Figure 4.7. The data shows that most samples have less than 200 missing packets. In these samples, the time position of the missing packet could vary greatly. For example, they could be 200 consecutive time bins, or they could be distributed evenly within a sample. The evenly distributed case could be an indicator of an upcoming supervision violation while in the consecutive case, the device would already be in supervision violation. This means that a model has the ability to derive information from the lack of data, but it could also be

Device type 1

Max empty bins	Bins/sample	Test 1	Test 2	Test 3	Average
30	960	0,065	0,130	0,041	0,079
100	960	0,162	0,316	0,437	0,305
900	**960**	**0,505**	**0,203**	**0,301**	**0,336**

Device type 2

Max empty bins	Bins/sample	Test 1	Test 2	Test 3	Average
30	960	0,622	0,075	0,229	0,309
100	960	0,619	0,312	0,267	0,399
900	**960**	**0,730**	**0,754**	**0,422**	**0,635**

Max empty bins	Bins/sample	Mean average
30	960	0,194
100	960	0,352
900	**960**	**0,486**

Table 6.4: The average precision using different allowed max empty bins for each device type, with the top scores highlighted.

fooled by it as it could potentially relate long sequences of imputated data with one of the classes.

Still, two things should be noted from this result. A higher threshold gives higher average precision scores, though removing the filter completely does not. Early experiments showed that without this filter, the model lost its' ability to learn anything at all. Therefore, the filtering mechanism was introduced.

Secondly, the implementation of the filtering mechanism was very simple as it just counted the number of empty bins in a sample. By using a finer algorithm, better average precision scores might be possible to achieve. One example of such an algorithm is limiting the number of consecutive missing time bins, instead of just counting the occurrences.

6.1.4 Bin size + bins/sample

As noted in subsection 4.2.4, each sample is a set of consecutive bins. Different combinations of *bin size* and *bins per sample* can make samples represent different time durations with different granularities. Experiments were designed to investigate how this affects average precision.

The results can be seen in Table 6.5. When controlling bin size, the results show that 5/6 times, shorter samples are preferable. This means that the model should have less historical data to base a prediction on (fewer time bins per sample). When controlling bins per sample, there is not as clear a pattern to extract. No pattern can be found for device type 1, but device type 2 seem to favour a longer bin size. This means that device type 2 should use lower data granularity.

It should be noted that when controlling either the bin size or bins/sample, the duration of the time sample differs. For example, locking the bins/sample to 960 and changing the bin size to 15, 30 and 60 would result in three different sequences of increasing length, as the number of bins/sample is constant. Therefore, it is of interest of studying the results when the sequences are always the same.

The results, as seen in Table 6.5, show that increasing the bin size, on the same sequence of data, increases the average precision; i.e decreasing the data resolution increases the model performance. This could indicate multiple things. One explanation is that the information in the data is not enough to enable high resolution models. Another suggestion is that the models are mostly influenced by larger differences in the sequences.

6.1.5 Imputation scope

Two imputation scopes has been defined, *per sample* and *per timeseries*. Experiments were made to investigate their impact of average precision. The results can be seen in Table 6.6. The results are inconclusive as device type 1 significantly better with imputation per sample, while device type 2 is better with imputation on time series level. More investigation is needed in this area, but is left unexplored due to time constraints.

6.1.6 Combinations of features

Seven experiments with different feature combinations were conducted. The results of each experiment is detailed in Table 6.7, with the best results for every device type highlighted. Note that the scores per device type are already averaged over the same test sets as in all other experiments. For more detailed data regarding the individual results, refer to Appendix A.

The data shows that the different device types prefer different sets of features. A clear pattern is hard to extract, but it can be observed that both device types promote a set of features with RSSI values and without LQI values. However, the tables in Appendix A shows that the experiments are highly sensitive to device type and test set.

6.1.7 Classifying with regression

As noted in section 4.4, the regression model generates future time bins with features and supervision violation predictions and was used iteratively to travel seven days into the future. To provide a single supervision violation prediction, all generated supervision violations had to be aggregated.

Eight different aggregation methods were put to test. When looking at the results in Table 6.8, it can be seen that the method which produced the highest average precision score was mean and max. The table also provides a comparison for each method to the direct classification approach. The results show that direct classification is to be preferred. It is important to note that the regression model did not undergo the same thorough investigation of neural network architectures and data pre-processing parameters, due to a combination of long running times

Device type 1

Bin size (min)	Bins/sample	Test 1	Test 2	Test 3	Average
15	480	0,569	0,140	0,214	0,308
15	960	0,162	0,316	0,437	0,305
15	1440	0,011	0,176	0,161	0,116
30	**480**	**0,670**	**0,274**	**0,414**	**0,453**
30	960	0,174	0,168	0,288	0,210
60	240	0,573	0,256	0,476	0,435
60	960	0,345	0,153	0,297	0,265

Device type 2

Bin size (min)	Bins/sample	Test 1	Test 2	Test 3	Average
15	480	0,696	0,617	0,478	0,597
15	960	0,619	0,312	0,267	0,399
15	1440	0,654	0,094	0,340	0,363
30	480	0,546	0,538	0,607	0,564
30	960	0,681	0,539	0,440	0,554
60	240	0,612	0,608	0,503	0,574
60	**960**	**0,662**	**0,708**	**0,704**	**0,691**

Bin size (min)	Bins/sample	Mean average
15	480	0,452
15	960	0,352
15	1440	0,239
30	**480**	**0,508**
30	960	0,382
60	240	0,505
60	960	0,478

Table 6.5: The average precision using different bin size + bins/sample combinations per device type, with top scores highlighted

Device type 1

Imputation scope	Test 1	Test 2	Test 3	Average
Sample	**0,162**	**0,316**	**0,437**	**0,305**
Time series	0,153	0,279	0,110	0,181

Device type 2

Imputation scope	Test 1	Test 2	Test 3	Average
Sample	0,619	0,312	0,267	0,399
Time series	**0,661**	**0,405**	**0,371**	**0,479**

Imputation scope	Mean average
Sample	**0,352**
Time series	0,330

Table 6.6: The average precision using different imputation styles for each device type, with the top scores highlighted

and time constraints. Therefore, the regression model should not be disregarded based solely on these results.

6.2 Top model performance

Even though the focus of the study was not on maximising average precision, there is value in highlighting some of the top performing models. They will be presented with precision-recall curves and average precision scores. In the curves, the *label threshold* shows the different decision boundaries tested, while the *supervision* curve shows the precision recall values for each boundary. The best performing models from experiments not highlighted here, can be found in Appendix B.

To begin with, the graphs for the best architectures are shown in Figure 6.1 (device type 1) and Figure 6.2 (device type 2). The figures show that both models have the ability to retain a high precision for a significant increase in recall, producing a maximum F1-score around 0.75 for both device types.

Similar results can be seen in other experiments. Figure 6.3 shows device type 1 from the bin size and bins per sample experiment and Figure 6.4 shows device type 2 from the max empty bins experiment. In the same way as for the presented architecture models, a high precision is kept as recall increases. It is also seen that the model for device type 2 can achieve an F1-score of ∼0.8. Both of these models were trained on the baseline architecture, with no other variables changed than the noted.

Feature Combination																
RH	TRH	RV	TRV	LH	TLH	LV	TLV	RB	MP	IS	CU1	CU2	MCU	Type 1	Type 2	Average
×	×	×	×	×	×	×	×	×	×	×	×	×	×	0,305	0,399	0,352
×	×	×	×	×	×	×	×		×	×	×	×	×	0,308	0,486	**0,397**
×	×	×	×	×	×	×	×	×	×	×				0,283	0,467	0,375
×	×	×	×											0,155	**0,555**	0,355
×	×	×	×	×	×	×	×	×		×	×	×	×	0,296	0,394	0,345
×	×	×	×	×	×	×	×	×						0,135	0,537	0,336
×	×	×	×					×	×	×	×	×	×	**0,368**	0,300	0,334
×		×		×		×		×	×	×	×	×	×	0,199	0,423	0,311
×		×												0,102	0,417	0,260
×		×		×		×		×						0,118	0,400	0,259
				×	×	×	×		×	×	×	×	×	0,346	0,159	0,252
×	×	×	×	×	×	×	×	×	×		×	×	×	0,141	0,359	0,250
				×	×	×	×	×	×	×	×	×	×	0,158	0,260	0,209
				×	×	×	×							0,120	0,227	0,173
				×		×								0,115	0,202	0,159

Table 6.7: The average precision for different feature combinations. The top scores for each device type, and their combined mean, are highlighted.

Device type 1

Model	Aggregation	Test 1	Test 2	Test 3	Average
Classification	N/A	0,162	0,316	0,437	0,305
Regression	Min	0,047	0,031	0,019	0,032
Regression	Median	0,060	0,035	0,022	0,039
Regression	Mean	0,065	0,035	0,022	0,041
Regression	60 percentile	0,044	0,039	0,021	0,035
Regression	70 percentile	0,040	0,035	0,020	0,032
Regression	80 percentile	0,071	0,037	0,021	0,043
Regression	90 percentile	0,039	0,036	0,021	0,032
Regression	**Max**	**0,115**	**0,050**	**0,030**	**0,065**

Device type 2

Model	Aggregation	Test 1	Test 2	Test 3	Average
Classification	N/A	0,619	0,312	0,267	0,399
Regression	Min	0,331	0,124	0,179	0,211
Regression	Median	0,597	0,301	0,131	0,343
Regression	**Mean**	**0,620**	**0,400**	**0,136**	**0,385**
Regression	60 percentile	0,585	0,174	0,120	0,293
Regression	70 percentile	0,590	0,264	0,115	0,323
Regression	80 percentile	0,596	0,355	0,117	0,356
Regression	90 percentile	0,585	0,386	0,116	0,362
Regression	Max	0,540	0,413	0,148	0,367

Model	Aggregation	Mean average
Classification	N/A	0,352
Regression	Min	0,122
Regression	Median	0,191
Regression	Mean	0,213
Regression	60 percentile	0,164
Regression	70 percentile	0,177
Regression	80 percentile	0,199
Regression	90 percentile	0,197
Regression	**Max**	**0,216**

Table 6.8: The average precision for different aggregation methods in the regression model, with the highest scores highlighted.

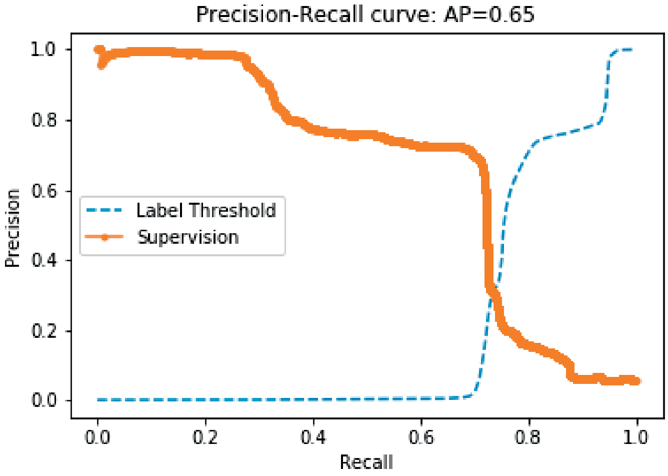


Figure 6.1: Precision Recall curve for the best model architecture of device type 1. (2 x 64 LSTM + 128 MLP)

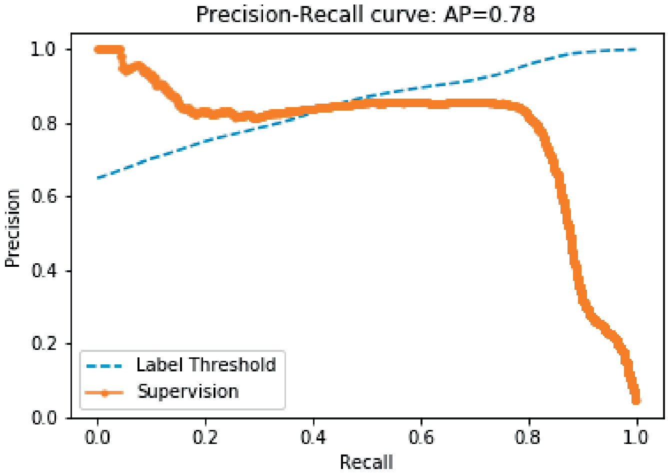


Figure 6.2: Precision Recall curve for the best model architecture of device type 2. (1 x 64 LSTM + 128 MLP)

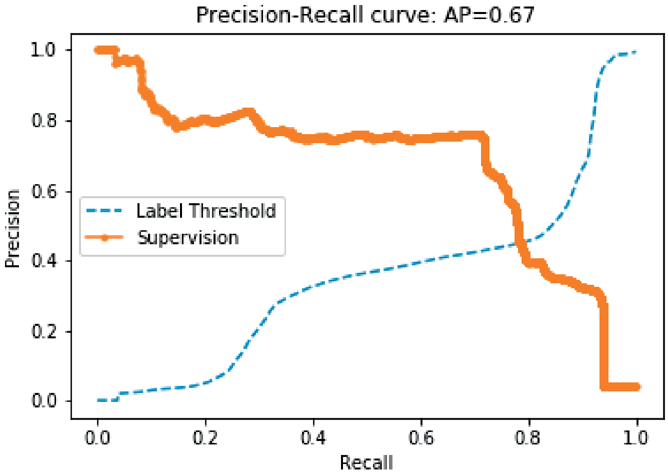


Figure 6.3: Precision Recall curve for the best performing model of device type 1. Achieved with binsize + bins/sample = 30 + 480; baseline model with no other changes

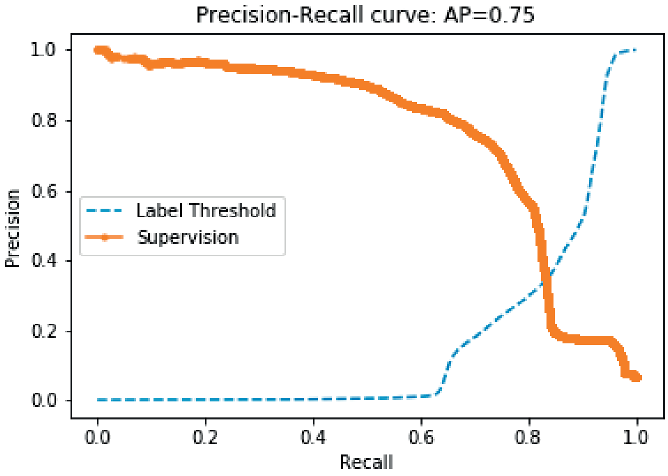


Figure 6.4: Precision Recall curve for the best performing model of device type 2. Achieved with max empty bins = 900; baseline model with no other changes

6.3 Discussion

An extensive set of experiments were made throughout this thesis. In this section, the results will be discussed in general. Also, some mention will be made about considerations before the models can be used in production systems.

6.3.1 Architectures, Features and Data

When predicting supervision violations, this thesis opted to use neural networks as the main model type. Initially, the expectations were that deeper architectures were needed, as the information in the data was thought to have a high complexity. From the experiments, this assumption was shown to be false, as the average precision showed the best results with more shallow architectures. This is favourable from both a development and production perspective, as it enables quicker iterations . When complexity increases, so does computational and memory cost.

An overall trend from most experiments is that scores are highly sensitive to test set with a high variability in average precision. As is often the case in machine learning, this is an indication that the problem and models are highly data dependent.

There are two challenges with the data in this study: 1. Its lack of volume and 2: The imbalance. This thesis ran for 20 weeks and data collection has been a part of it. The intensity of data collection had to be low, as to not place too much load on production systems. This led to a limited dataset size of less than a percentage of the theoretical dataset which would describe all installations. As the results show that more data increases the average precision, it is fair to assume that further expansion in the form of more installations and a longer time period, will increase average precision. How large an increase that can be expected is unknown.

The other problem is the imbalance which lies in the nature of the data. As a supervision violation is a very rare event, modelling this as a classifier creates a heavy imbalance between the classes. Experiments show, that better results are achieved when balancing the classes with the help of random oversampling. While there are methods available for more sophisticated oversampling [4], this thesis leaves those as future work.

One of the more surprising results from experimenting with the pre-processing parameters was the strict increase of average precision, as the number of allowed empty bins increased. In subsection 6.1.3 it was argued that the reason for this was that the model found information in the lack of data. Though, there is also an alternative hypothesis to this: high counts of empty time bins could be consecutive in a sample. This has the potential risk of introducing a bias into the model which conditions it to identify long sequences of imputed data as an indication of a supervision violation. Due to the time constraint of the thesis, no investigation was made into such a potential bias.

An early feature engineering process was made, identifying data which seemed to correlate with a supervision violation. It was thought that all features used in this thesis would help in driving average precision scores up. The results shows that this was not the case, further proving the difficulty of designing features in machine learning. In general, it is hard to draw any broad conclusions on which features to use, as it seems device type dependent. One indication is that the RSSI values are needed for a well performing model, as they are present in the experiments resulting in the highest average precision for both device types, and lacking in 3/4 of the worst. This is also reasonable, as the RSSI values has the most varied data of all of the features, hence containing the most information.

Deriving average precision scores based on neural network architectures, features and data pre-processing parameters is extremely hard. Still, this is not unexpected due to black box nature of neural networks. Instead, this thesis suggests broad experimentation when developing a model and general scepticism when drawing conclusions of their performance, as the results between testsets varies greatly.

6.3.2 Choosing a decision boundary from precision recall curves

Throughout this thesis, the main measurement of results has been through the average precision metric. This is mainly because it is a good metric when working with imbalanced datasets (see subsection 2.3.1), but also because it provides a decision boundary independent property, removing the challenge of labelling a supervision violation prediction as true or false. This removes the authors bias towards what the threshold should be. Thereby, from a research perspective, using average precision as metric seems reasonable.

For a model to be used in a system, a decision boundary has to be decided upon. With the help of the precision-recall curves, the decision boundary can be chosen differently depending on the desired behaviour. For example, if it is deemed more important to be certain of correct supervision violation predictions, a decision boundary with a high precision could be chosen, knowing how much recall would be sacrificed. The same is true the other way around. Hence, the precision-recall curves enable a high flexibility for a business depending on the requirements on predictions.

6.3.3 Regression vs Direct Classification

The regression and classification model tries to solve the same problem using different approaches. From a flexibility standpoint, the regression model is favourable, as it is based on a time scope independent concept.

The results (see Table 6.8) for device type 1 show that the regression model was unable to detect supervision violations. It is hard to determine the cause of this. It could be that a bad set of features and/or experimentation variables were used during training. It could also be that the chosen neural network architecture was not complex enough to handle a regression based solution compared to a classification based. Another explanation could be that the data was not rich enough to make the model generate a viable set of future datapoints.

Still, when studying device type 2, the regression approach performed on par with the classification approach. As the data is device specific, there could be reason to believe that some of the data for devices of type 2, are not present for devices of type 1. This could mean that there exists a regression based model which outperforms the direct classification approach.

It should be noted that the regression approach was not as well studied as the direct classification approach due to the time constraint and limited computing resources of the thesis. Many experiments could not be completed, leaving tracks unexplored which could have lead to a well performing regression model. For example, it might be that the default variables values used when comparing the regression and classification approaches were biased towards the classifier. Hence, if the same amount of experimentation was made with the regression model as direct classification, a similar or even comparably better regression model could be found.

Chapter 7

Conclusion

7.1 Summary

This thesis has explored the possibility of predicting whether radio enabled communication devices will be unable to communicate in a near future, also known as a entering supervision violation status. To solve this problem, a machine learning solution was developed using neural networks. The data enabling this research contained a sequence of radio packets, along with device meta data.

Two concepts based on neural networks were presented to solve the task of binary classifying samles with time series data. The first was a traditional supervised machine learning approach in the form of a classifier. The second was a generative model based on linear regression used for time time travelling, combined with a post-processing solution enabling classification.

The data was put through thorough pre-processing transformations. This included methods such as time re-sampling, imputation of missing data points and aggregation of data unevently distributed in time. The pre-processing solution was made configurable, enabling a large set of experiments for deriving prediction performance.

A large set of experiments were made, evaluated with average precision as metric. Everything from neural network architectures, size of discrete time bins, imputation scopes, device types and dataset sizes were put to test. This lead to a detailed study of how a large set of variables influence neural networks' ability to predict supervision violations. Conclusions were drawn from this, providing direction and support for developing new models.

The study showed that it was possible to predict supervision violations within seven days with average precision scores of up to 0,75, but that the problem is highly sensitive to data. As such, no general purpose model was found that provided respectable scores. Instead the need for experimentation when creating new models for different device types were emphasised.

7.2 Future work

7.2.1 Max allowed empty bins

This thesis implemented the max allowed empty bins filter as early data investigation showed that many samples were completely lacked information. The study concluded that a higher threshold lead to higher average precision scores. This raises the question whether the filter should exist at all. There are reasons to believe that samples with solely imputed data will hurt average precision scores, but only a deeper study will tell.

The study also raised the concern that even though the amount of allowed empty bins is limited, the order of those bins are not considered. There seem to be valuable information in the lack of data, but too many consecutive empty time bins could raise average precision score falsely. 900 consecutive time bins of size 15 minutes represent 225 hours of missing data, which is a more than 100 times the duration before a supervision violation is triggered. It has to be ensured that the model does not learn to correlate supervision violations with samples where a device is already in that state.

7.2.2 Bin size + Bins per sample

The conclusion that models prefer longer time series with lower data granularity was drawn. This study collected limited data due to time constraints. The time series could not become too long while keeping the granularity low as the dataset size was insufficient. The longest continuous duration of a dataset in this thesis was 3 months. Future work could be made by collecting a longer continuous data stream. This thesis recommends the next data collection to be made for at least six months, though the longer the better. This would allow the study regarding longer time series with low data granularity to continue.

7.2.3 Data distribution

This study used data representing less than 1% of all installations which could fail to represent the full distribution fully. It was concluded that a larger dataset lead to higher average precision scores, although this was between two data sets. The conclusion could be backed up better by presenting a third, even larger dataset.

The study also showed that average precision scores varied wildly between test sets. It could be interesting to investigate what makes models perform more equal between test sets, as that could mean a better fit to the true data distribution. With a better fit, more sophisticated methods for data utilisation could be used (such as stratified k-cross on train/validation data). As this would heavily limit the study as a whole due to time constraints, this thesis chose to not use more sophisticated methods.

7.2.4 Imputation and Features

This study performed Forward-Backward-Fill on two different scopes, per sample and the entire time series. It could be interesting to investigate with different techniques to gain insight into how well Forward-Backward-Fill performs to other techniques such as linear interpolation or SMOTE.

The project also found two pattern during the feature combination experiments. It could be interesting to further the feature engineering process. Some examples of new features could be room temperature, device temperature and device CPU load.

7.2.5 Classification through regression

This thesis presented the concept of classification through regression and started investigating its effectiveness. This part of the study was limited due to the time constraint. The best performing architecture from the direct classification concept study was used, and comparison was only made using probability aggregation methods in the regression model. Evaluation was made on the models ability to generate features representing the true data distribution. With a less strict time constraint, the regression concept would be more carefully evaluated by this thesis.

7.2.6 Model interpretation of data

Some of the features used in this thesis were constant and did not change throughout the time series. Though, these features were still provided to the network as features in a time series. This was to enable the use of installation metadata in making predictions (see Table 4.2 for a detailed description of features). It could of interest to investigate whether these features could be fed to the neural network with some other method, more suitable for constant data.

7.2.7 Model optimisation

The aim of this thesis was to research the feasibility of predicting supervision violations, together with studying how different factors affect the performance of the models. Therefore, no systematic attempt to reach the highest possible scores have taken place. This paper leaves as future work to maximise the average precision.

The data used in this thesis was also a multivariate time series. Independence between all variables was assumed to allow a structured search method to be used when defining experiments. More information and better features may be able to be extracted if the variable correlations were to be taken into account.

When developing neural networks for this thesis, no regards were taken to their internal hyper parameters. This means that there might be a combination of them leading to a better model. Also, for the recurrent feature of the network, only LSTM nodes where tested. Hence, it could be of interest to investigate how other types of implementations, such as GRUs, would perform.

References

[1] IEEE 802.15.4-2011 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4*, 2011.

[2] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] David Baillot. Neuron. `https://www.eurekalert.org/multimedia/pub/175358.php` [Accessed 23-Nov-2018], November 2018.

[4] Gustavo Batista et al. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[5] Gunnar Blom. *Sannolikhetsteori och statistikteori med tillämpningar*. Studentlitteratur, 1970.

[6] Anders Buhl and Hugo Hjertrén. Evaluation of artificial neural networks for predictive maintenance. Master's thesis, Lund University, jun 2018.

[7] Kenneth P Burnham and David R Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003.

[8] Nitesh V Chawla et al. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[9] François Chollet et al. Keras. `https://keras.io`, 2015.

[10] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015.

[11] Wikimedia Commons. Artificial neural network. `https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg` [Accessed 30-Nov-2018], November 2018.

[12] Wikimedia Commons. Long-short term memory node. `https://upload.wikimedia.org/wikipedia/commons/thumb/5/53/Peephole_Long_Short-Term_Memory.svg/1000px-Peephole_Long_Short-Term_Memory.svg.png` [Accessed 5-Dec-2018], December 2018.

[13] Wikimedia Commons. Precision and recall. `https://commons.wikimedia.org/wiki/File:Precisionrecall.svg` [Accessed 21-Dec-2018], December 2018.

[14] Wikimedia Commons. Regression analysis. `https://commons.wikimedia.org/wiki/File:Linear_regression.svg` [Accessed 23-Nov-2018], November 2018.

[15] Brian P Crow et al. Ieee 802.11 wireless local area networks. *IEEE Communications magazine*, 35(9):116–126, 1997.

[16] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[17] Leon A Gatys et al. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[18] Klaus Greff et al. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2017.

[19] Donald O Hebb. The first stage of perception: growth of the assembly. In *The Organization of Behavior*, pages 102–120. Psychology Press, 2005.

[20] Martin Hilbert and Priscila López. The world's technological capacity to store, communicate, and compute information. *science*, page 1200970, 2011.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[23] WL Junger and A Ponce De Leon. Imputation of missing data in time series for air pollutants. *Atmospheric Environment*, 102:96–104, 2015.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Will Koehrsen. Beyond accuracy: Precision and recall. `https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c` [Accessed 21-Jan-2019], March 2018.

[26] Lahiru Liyanapathirana. Machine learning workflow on diabetes data: Part 01. `https://towardsdatascience.com/machine-learning-workflow-on-diabetes-data-part-01-573864fcc6b8` [Accessed 23-Nov-2018], Feburary 2018.

[27] Elizabeth A Martin. *Concise medical dictionary.* Oxford University Press, USA, 2015.

[28] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[29] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[30] Mehryar Mohri et al. *Foundations of machine learning*. MIT press, 2018.

[31] Steffen Moritz et al. Comparison of different methods for univariate time series imputation in r. *arXiv preprint arXiv:1510.03924*, 2015.

[32] Oracle. Regression. `https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/regress.htm#DMCON005` [Accessed 23-Nov-2018], November 2018.

[33] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] James W Perry et al. Machine literature searching x. machine language; factors underlying its design and development. *American documentation*, 6(4):242–254, 1955.

[35] Prognostics and Health Management Society. Phm data challenge 2015. `https://www.phmsociety.org/events/conference/phm/15/data-challenge`, 2015.

[36] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[37] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[38] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. 1961.

[39] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, pages 71–105, 1959.

[40] Yutaka Sasaki et al. The truth of the f-measure. *Teach Tutor mater*, 1(5):1–5, 2007.

[41] David Schatsky et al. Cognitive technologies: The real opportunities for business. *Deloitte review*, 16:115–129, 2015.

[42] Klaus Schwab. *The fourth industrial revolution*. Crown Business, 2017.

[43] Toby Segaran and Jeff Hammerbacher. *Beautiful data: the stories behind elegant data solutions*. " O'Reilly Media, Inc.", 2009.

[44] David Silver et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[45] Saishruthi Swaminathan. Logistic regression — detailed overview. `https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc` [Accessed 23-Nov-2018], March 2018.

[46] Ren Wu et al. Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*, 2015.

[47] Yuting Wu et al. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*, 275:167–179, 2018.

[48] Wei Xiao. A probabilistic machine learning approach to detect industrial plant faults. *arXiv preprint arXiv:1603.05770*, 2016.

[49] Cong Xie et al. Feature extraction and ensemble decision tree classifier in plant failure detection. In *Annual Conference of the Prognostics and Health Management Society*, 2015.

[50] Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2:30, 2004.

Appendix A

Detailed results for the features experiments

Used features														Device type 1			
RH	TRH	RV	TRV	LH	TLH	LV	TLV	RB	MP	IS	CU1	CU2	MCU	Test 1	Test 2	Test 3	Average
×	×	×	×	×	×	×	×	×	×	×	×	×	×	0,162	0,316	0,437	0,305
×	×	×	×	×	×	×	×		×	×	×	×	×	0,197	0,358	0,368	0,308
×	×	×	×	×	×	×	×	×	×	×				0,188	0,199	0,461	0,283
×	×	×	×											0,122	0,152	0,192	0,155
×	×	×	×	×	×	×	×	×		×	×	×	×	0,121	0,326	0,441	0,296
×	×	×	×	×	×	×	×	×						0,114	0,235	0,057	0,135
×	×	×	×					×	×	×	×	×	×	0,174	0,369	0,560	0,368
×		×		×		×		×	×	×	×	×	×	0,167	0,246	0,185	0,199
×		×												0,105	0,132	0,070	0,102
×		×		×		×		×						0,122	0,198	0,032	0,118
				×	×	×	×		×	×	×	×	×	0,193	0,324	0,519	0,346
×	×	×	×	×	×	×	×	×	×		×	×	×	0,102	0,232	0,090	0,141
				×	×	×	×	×	×	×	×	×	×	0,191	0,217	0,065	0,158
				×	×	×	×							0,135	0,154	0,070	0,120
				×		×								0,238	0,086	0,021	0,115

Table A.1: Detailed results for the features experiment on device type 1

Used features														Device type 2			
RH	TRH	RV	TRV	LH	TLH	LV	TLV	RB	MP	IS	CU1	CU2	MCU	Test 1	Test 2	Test 3	Average
×	×	×	×	×	×	×	×	×	×	×	×	×	×	0,619	0,312	0,267	0,399
×	×	×	×	×	×	×	×		×	×	×	×	×	0,717	0,362	0,380	0,486
×	×	×	×	×	×	×	×	×	×	×				0,618	0,438	0,345	0,467
×	×	×	×											0,647	0,573	0,443	0,555
×	×	×	×	×	×	×	×	×		×	×	×	×	0,677	0,289	0,217	0,394
×	×	×	×	×	×	×	×	×						0,587	0,554	0,469	0,537
×	×	×	×					×	×	×	×	×	×	0,592	0,198	0,110	0,300
×		×		×		×		×	×	×	×	×	×	0,731	0,157	0,381	0,423
×		×												0,637	0,221	0,394	0,417
×		×		×		×		×						0,276	0,474	0,451	0,400
				×	×	×	×		×	×	×	×	×	0,331	0,082	0,065	0,159
×	×	×	×	×	×	×	×	×	×		×	×	×	0,630	0,220	0,226	0,359
				×	×	×	×	×	×	×	×	×	×	0,442	0,217	0,121	0,260
				×	×	×	×							0,289	0,317	0,074	0,227
				×		×								0,233	0,313	0,061	0,202

Table A.2: Detailed results for the features experiment on device type 2

Appendix B

Precision recall curves for top performing models in each experiment per device type

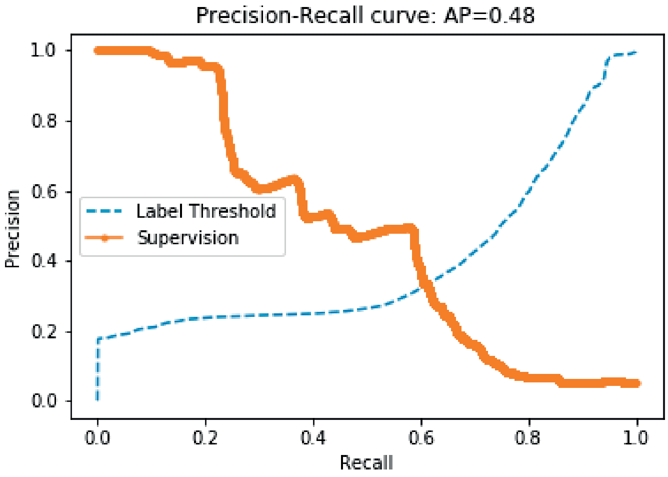


Figure B.1: Precision Recall curve for the top performing model in the architecture grid search experiment. (Device type 1)

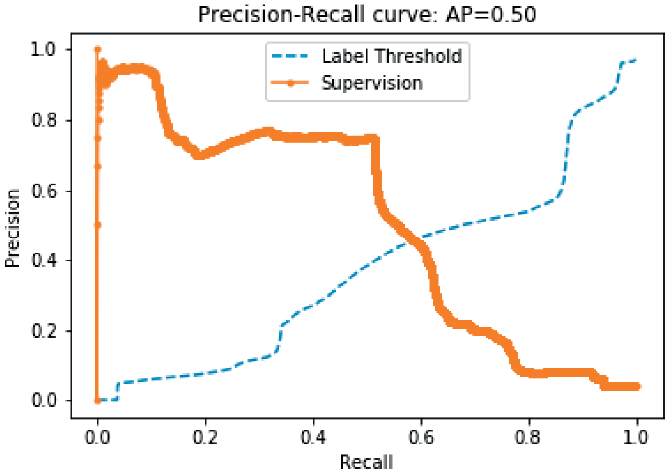


Figure B.2: Precision Recall curve for the top performing model in the max empty bins experiment. (Device type 1)

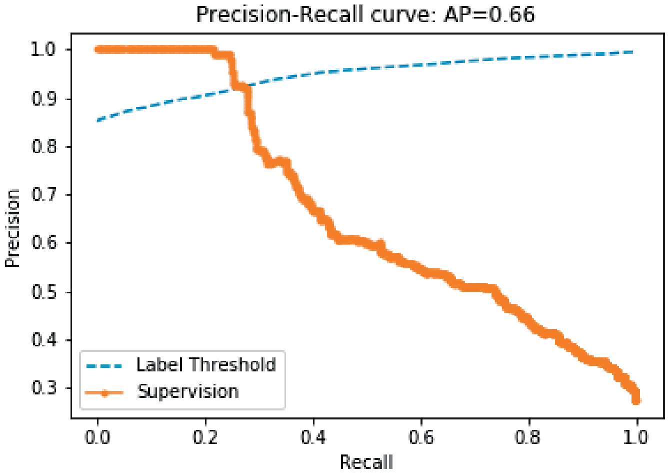


Figure B.3: Precision Recall curve for the top performing model in the binsize + bins/sample experiment. (Device type 2)

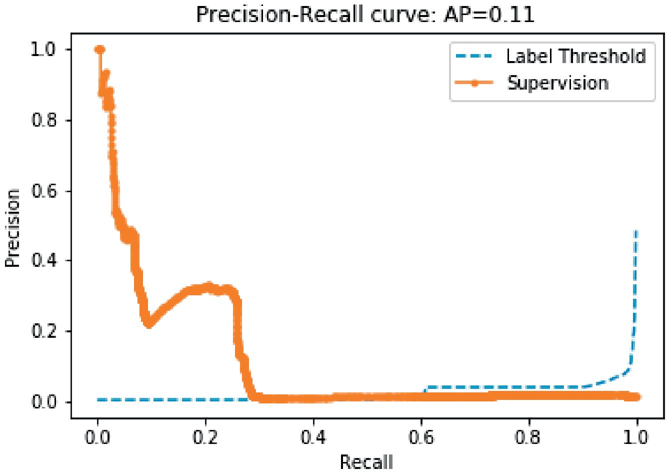


Figure B.4: Precision Recall curve for the top performing regression model. (Device type 1)

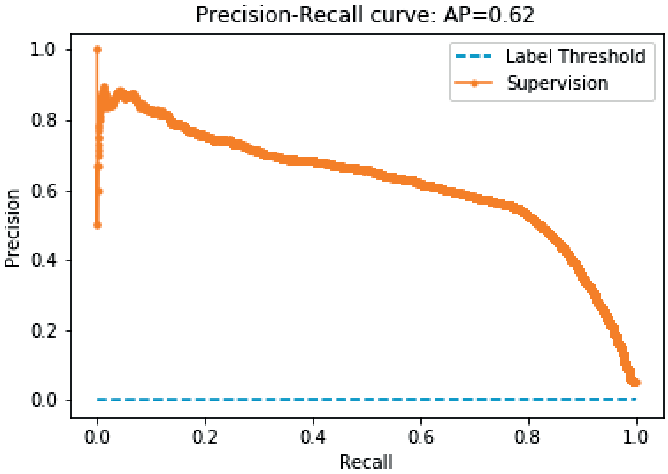


Figure B.5: Precision Recall curve for the top performing regression model. (Device type 2)

Printed by Tryckeriet i E-huset, Lund 2019

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2019-683
http://www.eit.lth.se