Carbon Dioxide Level Prediction for Indoor Air Using Neural networks

University of Oulu
Faculty of Information Technology and Electrical Engineering/ M3S
Master's Thesis
Riku Mäkynen
16.04.2020

Abstract

Indoor air quality is important for our health and well-being. It has been proven that the air quality affects the performance of the workers. One way to achieve better air quality, would be to adjust the air conditioning and heating. By predicting the room conditions one can react faster to changes and ensure that the room conditions stay favourable.

The existing machine learning (ML) models that are used for CO2 prediction are rather basic. This study aims to improve upon the performance of the models compared to earlier studies. The focus of this thesis is to study what type of model would give the best results, what type of training data should be used, and how long history should be fed into the model.

One of the goals of this thesis is to examine whether a deep neural network is better than a wider one. The used data consists of indoor air measurements from nine variables: CO2, Temperature, pressure, illuminance, volatile organic compound (VOC), movement detection, humidity, door state. The data was gathered in VTT's Oulu office (in Finland). Different combinations of input variables are experimented on, to find out, which inputs should be fed into the network. The performance is compared to other models commonly used in prior studies. These models include previous value forward (PPV), line fit, and a Multilayer perceptron with one hidden layer (MLP1). Several hyperparameters are tested to find out which combination of parameters has the lowest error.

Compared to earlier studies, the developed deep Multilayer Perceptron (MLP) model improved root mean squared error (RMSE) by 0,997 ppm. This indicates that deep models perform better for CO2 prediction tasks. The total root mean squared error was 6.07 ppm. This improvement makes it possible to give more accurate readings for the air conditioning control system, which in turn makes it easier to keep CO2 levels low. A history length of seven minutes is used as the input, and the model predicts ten minutes ahead.

Keywords
Deep neural networks, CO2 time series prediction, Convolutional neural network (CNN), artificial neural network, Indoor Air Quality, Multilayer perceptron (MLP), Long short-term memory (LSTM), Gated recurrent unit (GRU), Demand controlled ventilation, CO2, CO2 level prediction

Supervisor
PhD, Postdoctoral Researcher, Umar Farooq

Table of Contents

1. Introduction

We spend a large portion of our time indoors, and indoor air quality affects both our wellbeing and performance. Too high CO2 levels cause performance to decrease and negative effects for the occupants. To this end, indoor air can be controlled so that the CO2 level doesn't get too high. To better control the indoor CO2 concentration, it is beneficial to predict CO2 level, which allows the ventilation system to adjust the ventilation pre-emptively. The focus of this thesis is to build a neural network that can predict the CO2 level in indoor air.

This chapter describes the motivation and research objectives of the study. Research methods and prior contributions from existing literature are also described. Finally, the main contributions of this study are explored, and the outline of the thesis is given.

1.1 Motivations for controlling indoor air

It is very important to predict CO2 level in the future. By predicting CO2 level, we can better control the ventilation and avoid high CO2 concentration in buildings and rooms. There are three main motivation behind this study and to predict CO2 level in future, i.e. saving energy, increasing comfort and improving job performance. Energy savings come from the reduced need to heat (or cool) the house. Also, the fact that ventilation doesn’t have to be running all the time saves energy.

A study Merema, Delwati, Sourbron, and Breesch (2018) found that energy requirements for ventilation could be reduced by 25-55% and heat losses by 25-32% when compared to a constant rate ventilation system. Li, Wall and Platt (2010) observed a 62% reduction in the required airflow when a demand-controlled ventilation system was configured to keep CO2 concentration at a specific level. Several studies found that cognitive performance decreased when CO2 concentration was high (Satish et al. ,2012; Maddalena et al. ,2015; Haverinen-Shaughnessy, Moschandreas, & Shaughnessy, 2011; Haverinen-Shaughnessy & Shaughnessy, 2015). Haverinen-Shaughnessy et al. (2011) observed that students passed standardized tests at a higher rate with higher ventilation rates. Several studies mentioned Sick Building Syndrome (SBS) (Khazaei, Shiehbeigi, & Kani, 2019; Skön, Johansson, Raatikainen, Leiviskä, & Kolehmainen, 2012). According to the World Health Organization (WHO), SBS is a condition where a person has symptoms for apparently no reason while they are in a building (World Health Organization Regional Office for Europe, n.d.). Norhidayah, Chia-Kuang, Azhar, and Nurulwahida (2013) found that ventilation was one factor that can predict sick building syndrome. Chao et al. (2003) found that upper respiratory symptoms were more prevalent with higher CO2 concentrations.

In summary by controlling a ventilation system based on demand, one can improve energy efficiency, job performance, and make the environment more comfortable for the occupants. By predicting the future values of CO2 one can react faster to changes in CO2 concentrations and prevent CO2 concentrations from rising too high.

1.2 Research objectives

The purpose of this study is to find out an effective neural network model for predicting carbon dioxide (CO2) levels in indoor air. By predicting the CO2 levels, the control of the ventilation can be done more intelligently which can improve the sense of satisfaction for the worker and potentially reduce the energy costs.

For regression tasks it has been established, that a model that combines inputs from several different predictors gives better results than any single predictor (Zhang, 2003; Divina, Gilson, Goméz-Vela, García Torres, & Torres, 2018; Evitan, 2019). To limit the scope of the thesis, combined models (combined regressors) are excluded.

1.3 Research questions

In this thesis, two research questions are answered. The first research question is related to the structure of the network. The second research question attempted to find out what kind of training data should be used. Research questions are the following:

RQ1: What kind of network structure would produce the most accurate CO2 prediction results?

This research question aims to find out what the structure of the network should be.

RQ1.1 What network type should be used?

This question aims to find out if one should use a wide or a deep network, and which type of network should be used, and what activation functions should be used. The tested networks are the following: Multilayer Perceptron (MLP), Long short-term memory (LSTM), Gated recurrent unit (GRU), and Convolutional neural network (CNN).

RQ1.2 Should shortcut connections be used?

In image classification, a lot of effort has been spent on developing effective network structures. One of these structures is called the “skip connection”, also known as “shortcut connection”. This structure was also explored in this study in order to determine, whether it also benefits neural networks related to CO2 prediction.

RQ1.3 What to feed to the network?

Another factor that is related to the network structure is the number of input time steps one should feed into the network since this affects the number of parameters the first hidden layer will have. Feature engineering also affects the number of parameters for the first hidden layer and is experimented on as well.

RQ2: What kind of training data gives the most accurate CO2 prediction results?

There are two sub-questions are related to this research question.

RQ2.1 Should low variation sections be excluded?

This question aims to find out if all training data should be used, or if low variation sections should be excluded from the training data. Most of the time office rooms are empty, which was also evident in a study by Candanedo & Feldheim (2016), where it was

found that anywhere from 64% to 79% of the time office rooms are empty. This type of data can easily result in unbalanced data, where low variation sections are overrepresented. My assumption is that network trained with all the data could result in lower prediction accuracy due to the imbalanced dataset.

RQ2.2 Should training data from other rooms be used?

This question aims to find out if training data should only come from the same room one is trying to predict, or should one also include other rooms in the training data.

1.4 Main contribution

The main contribution of this thesis is to propose neural networks which can predict CO2 level in future efficiently. For this purpose, four different neural networks, i.e. MLP, LSTM, GRU and CNN are proposed and compared. In addition, a method for selection and cleaning data is introduced in order to use appropriate data for prediction. Furthermore, feature engineering is conducted in order to select the best set of features to be used by the neural networks.

1.5 Thesis Outline

This thesis consists of six chapters. Chapter 1 gives an overview of the research objectives, motivations, and contributions of the study. Chapter 2 describes how earlier studies have predicted CO2 values in the past. Chapter 3 discusses the research methods that were used in the study. Chapter 4 focuses on the implementation details, and details of each of the used networks. Chapter 5 contains the study, and the findings of the study. The final chapter highlights the findings, explores the limitations, and suggests possible future work regarding CO2 prediction.

2. Prior research

This chapter analysed the existing literature related to the study. In addition, it also explores the issues in existing methods. In order to search related literature, we used a list of relevant keywords. Three search engines, i.e. Google Scholar, Scopus and Finna Search Engine for Oulu University Student are used for finding relevant literature. In total 69 research articles were found using these search engines. The goal was to find literature that were recent, relevant to this study and have high impact. These three criteria are used to select the best articles. Finally, 39 research articles were selected for this study.

The structure of this chapter is the following. Section 2.1 analyses the methods proposed for CO2 measurement. Section 2.2 explains various neural networks that were used in this study. Section 2.3 summarizes the techniques that were used related to this study. Section 2.4 describes common methods that were used in prediction tasks. Section 2.5 describes the limitations of earlier research regarding CO2 prediction.

2.1 CO2 measurement methods

This section describes how CO2 level is measured either directly or indirectly. Two approaches are proposed in the literature for measuring CO2 concentrations: 1. Direct measurement, 2. and measuring the occupancy count in the room. The occupancy counts can be estimated using approaches such as occupancy schedule, occupancy sensors (motion sensors), and people counter on the door (Murphy, 2016). Another alternative to predict the occupancy would be to use cameras and automated person detector algorithms (Dong et al., 2010).

Skön et al. (2012) attempted to predict CO2 by only using temperature and relative humidity as the inputs. The study found that trying to predict CO2 merely from other variables such as temperature and relative humidity is difficult. (Skön et al., 2012.) This view was also expressed by another study by Khazaei et al. (2019). One way to estimate CO2 level would be to estimate it using mass balance equations, occupancy counts and airflow of the room (Calì, Matthes, Huchtemann, Streblow & Müller, 2014). However, this would require detailed information about the airflow and the occupant count. Due to this, perhaps the most straightforward way to measure CO2 is to use a CO2 sensor.

According to Merema et al. (2018) the placement of CO2 sensors is crucial. If this is not done properly, then the measurements will not represent the conditions that the people are sensing indoors. Measurement device near the exit does not work very well instead, the sensor was placed slightly closer to where people were present. (Merema et al., 2018.) The CO2 sensor should be mounted anywhere from 120 cm to 180 cm from the floor, away from air vents (“At What Height Should Sensors be Mounted?”, n.d.; “CO2 Sensor Location: Where to Mount Your CO2 IAQ Monitor”, 2012). Merema et al. (2018) mounted the sensors at 1.5 m height in their study.

2.2 Various neural networks

This section describes various neural network building blocks that can be used for regression tasks. Section 2.2.1 describes the convolutional neural network (CNN). Section 2.2.2 explains the multilayer perceptron (MLP). Section 2.2.3 details the Long short-term memory (LSTM) unit. Section 2.2.4 describes the gated recurrent unit (GRU).

2.2.1 Convolutional neural network (CNN)

This section explains how CNNs are used in literature for regression tasks. Convolution operation was first used by Fukushima in his Neocognitron paper (Fukushima, 1980, as cited in Nan, 2019). A crucial step in the use of convolutional neural networks is the use of backpropagation to train CNNs. The Convolutional network utilizing back backpropagation was first discussed by LeCun (LeCun et al., 1989, as cited in Nan, 2019). Let us first discuss what convolution is. See Figure 1 and Figure 2 below to see the difference between normal convolution and dilated convolution.

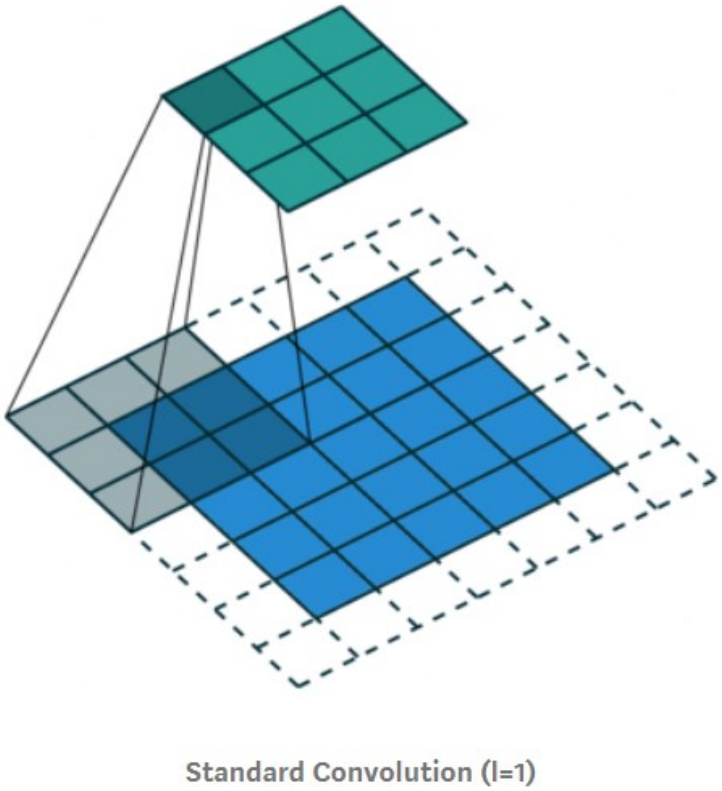

Figure 1. Standard convolution (Tsang, 2018. Used with permission from the author.)

In a convolution, an NxM neighborhood (grey area in Figure 1) and an NxM weight matrix (not depicted in Figure 1) are combined using a dot product. One may need to pad the original image with additional pixels at the edges if one wants to keep the resulting feature map (green area in Figure 1) the same size as the original image. In CNNs the goal is to calculate the values of these weight matrices. There can be several weight matrices, on each layer of the network. The number of weight matrices is dependent on how many convolutional nodes a specific layer has; each convolutional unit produces its own weight matrix.

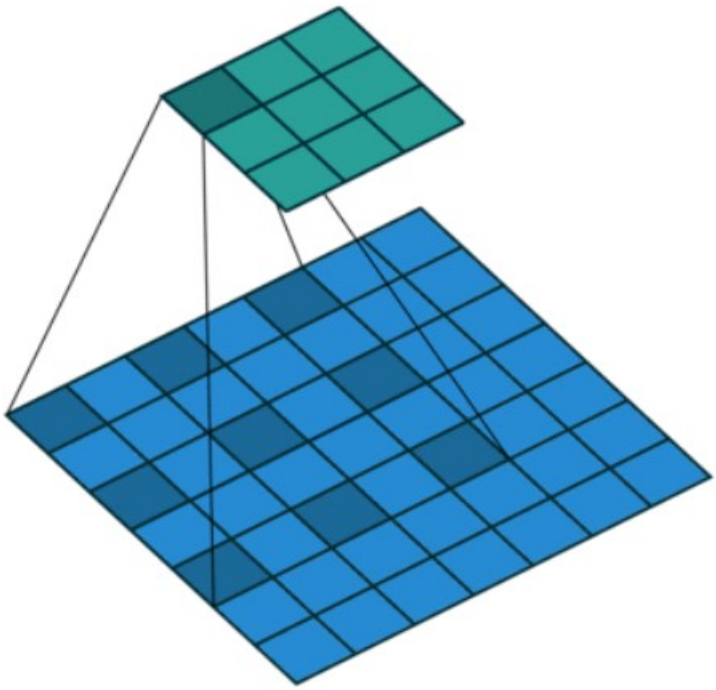

Figure 2. Dilated convolution. (Tsang, 2018. Used with permission from the author.)

The difference between a normal convolution and dilated convolutions is visible in Figure 1 and Figure 2. In the normal version, the neighbouring pixels are multiplied with the weight matrix. In dilated convolution, there is a gap between the pixels that are then multiplied (grey area in the image) with the weight matrix.

Convolutional neural networks have been mainly used in image classification in the past. For instance, Karpathy et al. (2014), and Redmon and Farhadi (2017) used CNNs in their image classification networks. CNNs have been successfully applied in regression tasks as well. For instance, Borovykh, Bohte, and Oosterlee (2018) utilized Google's WaveNet architecture in order to predict future values of a time-series — WaveNet is based on convolutions and dilated convolutions; Oord et al. (2016) used WaveNet to generate audio. Both instances demonstrate convolutional neural network's usability in regression tasks. WaveNet consists of layers of dilated convolutions that are able to predict the next value in a time-series (Borovykh et al., 2018).

2.2.2 Multilayer Perceptron (MLP)

Perceptron was invented in 1958 by Frank Rosenblatt (Rosenblatt, 1958). A perceptron is a binary classifier that utilizes the Heaviside step function as the activation ("Perceptron", n.d.). A multilayer perceptron is a network where several layers of perceptrons are used in successive layers. Other activations than Heaviside step function can also be used in multilayer perceptrons. Simple multilayer perceptron can be seen in Figure 4.

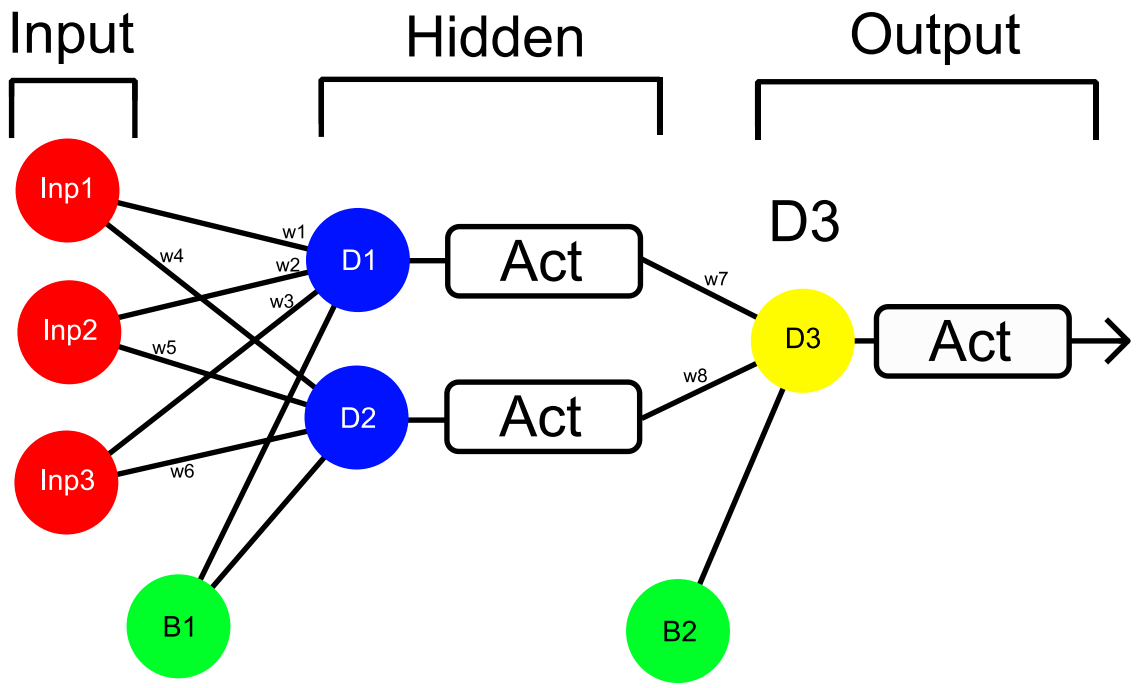

Figure 3. Multilayer Perceptron

Here the input layer contains features that are fed to the network and are always visible to the user. Output is also visible to the user. The output from D1 is activated using an activation function. B1 and B2 are biases. During training, the networks learns the weights and biases. Below are some examples on how to calculate the output from the network. Activation function in the example in 3-4 is hyperbolic tangent (tanh).

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \quad (1)$$

$$D1 = w_1 Inp_1 + w_2 Inp_2 + w_3 Inp_3 + B_1 \quad (2)$$

$$D1_{Act} = \tanh(D1) \quad (3)$$

$$D3_{Act} = \tanh(D1_{Act} w_7 + D2_{Act} w_8 + B2) \quad (4)$$

The general equation for an activated neuron is the following:

$$D_{Act} = Act(\sum_{i=1}^{n} w_i x_i + B) \quad (5)$$

Where B is the bias for that specific layer, n is the number of neurons in the previous layer, and Act is the activation function that was used. Furthermore, w stands for weight, and x is the output from a specific neuron. Each output from the previous layer is multiplied with a corresponding weight, for that specific neuron.

2.2.3 Long-Short-Term Memory (LSTM)

This section describes how LSTMS work. LSTM was invented by Hochreiter & Schmidhuber (1997). LSTM is able to capture long term dependencies over 1000 timesteps to the past (Hochreiter & Schmidhuber, 1997). The structure of an LSTM node is described in Figure 4 and also in the equations 7-12 that follow.

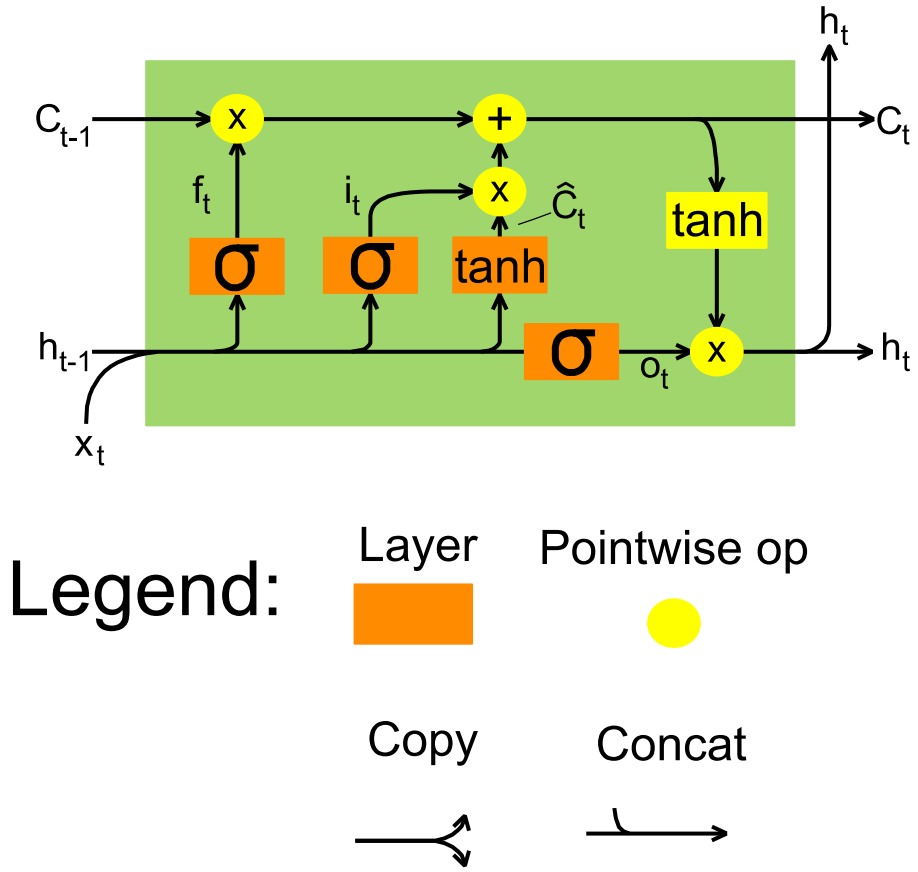

Figure 4. Long short-term memory (”Long short-term memory”, n.d.)

The LSTM equations in matrix form are the following (Olah, 2015)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (7)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (8)$$

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (9)$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \quad (10)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (11)$$

$$h_t = o_t * \tanh(C_t) \quad (12)$$

LSTM can be trained in stateful or stateless mode. In stateful mode, the hidden state h_t from the previous prediction will be used as the initial state for the next one (“tf.keras.layers.LSTM”, 2020).

2.2.4 Gated Recurrent Unit (GRU)

GRU has been motivated by LSTM. Compared to LSTM, the GRU unit is simpler and easier to implement. (Cho et al., 2014.) Figure 5 displays GRU that was used in this thesis, which is the default GRU version that is implemented in TensorFlow ("tf.keras.layers.GRU", 2020).

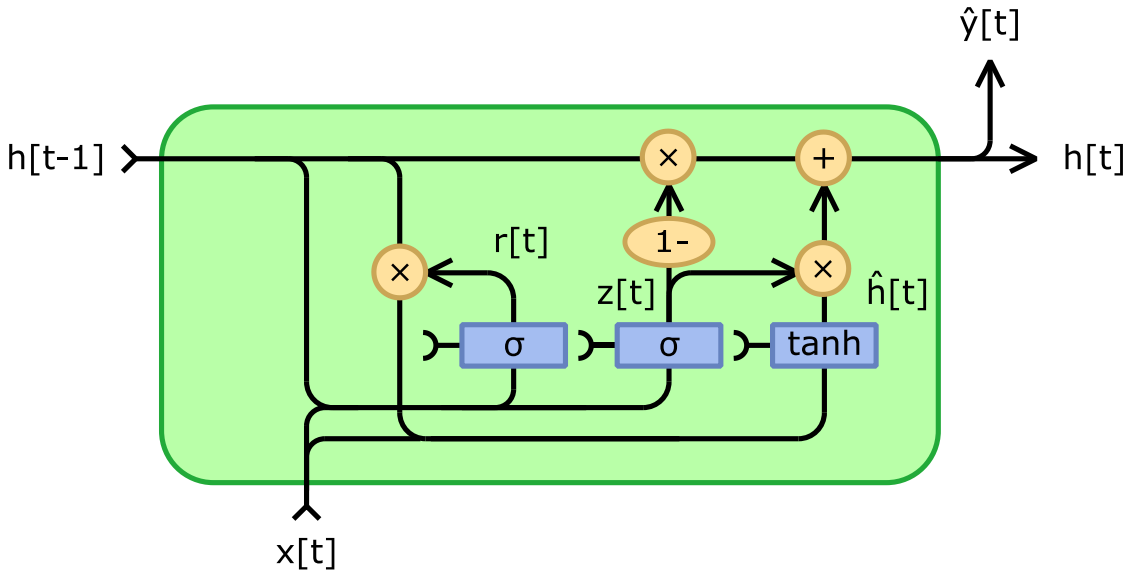

Figure 5. Gated Recurrent Unit ("Gated recurrent unit", n.d.)

In Figure 5 h[t-1] is the hidden state from the previous step, and x[t] is the current input value. The equations for GRU are shown in equations 13-16 below (Cho et al., 2014):

$$r_t = \sigma(W_r x + U_r h_{t-1}) \quad (13)$$

$$z_t = \sigma(W_z x + U_z h_{t-1}) \quad (14)$$

$$\hat{h}_t = \tanh\left(W_{\hat{h}} x + U_{\hat{h}}(r_t * h_{t-1})\right) \quad (15)$$

$$h_t = z_t h_{t-1} + (1 - z_t)\hat{h}_t \quad (16)$$

GRU can be trained in stateful or stateless mode. In stateful, mode the hidden state h_t from the previous prediction will be used as the initial state for the next one ("tf.keras.layers.GRU", 2020).

2.3 Prediction of CO2

This section discusses methods that were used to predict the future values of CO2. Khazaei et al. (2019) studied how multilayer perceptron (MLP) can be used to predict the future value of CO2 in indoor air. The focus of their study was to eliminate the need to count the people in the room in order to control ventilation based on the people count. They focused on predicting the CO2 concentrations in order to control the room conditions. In addition to measuring CO2, they also measured relative humidity and temperature. The data they used was sampled once per minute. They discovered that CO2 could be predicted accurately at least five minutes in the future. The mean squared error (MSE) was 17 ppm, which according to the paper is accurate enough for demand-controlled ventilation. The neural network algorithm was trained using the Levenberg–Marquardt algorithm, early stopping was used to prevent overfitting, and none of the networks were overfitted. In the study, hyperparameters such as neuron count in the hidden layer and the number of input time-steps to use, were found by experimenting with different variations in order to see which one gave the best results. The best artificial neural network (ANN) was one multilayer perceptron with one hidden layer of perceptrons (Dense), which had four neurons. The network took four previous time steps as the input. It should be noted that minimum concentration for CO2 was 485 ppm in the

study. (Khazaei et al., 2019.) According to Rödjegård (2017), the outdoor CO2 concentration is 400 ppm. Due to this, one could assume that CO2 levels would retreat to 400 ppm during the unoccupied period and the resulting CO2 average to be somewhere near 400 ppm. This highlights a potential issue with the comparability of the studies since the data that is used by each author can be very different.

Macarulla et al. (2017) predicted CO2 levels by constructing mass balance equations to predict the CO2 levels. They had data that was sampled once per 15 minutes. Their best model reached root mean squared error (RMSE) of 41.10 ppm. (Macarulla et al., 2017.) The downside of using mass balance equations is that one has to know the occupancy count, the ventilation rate, and the CO2 concentration of the air that is being ventilated. To train an ANN one does not have to know all this information.

Falk (2018) predicted CO2 values ten minutes to the future. He noticed that data from the weekends was quite monotone, due to this only data from weekdays were included in the training process. Falk predicted the CO2 by using the previous five previous time steps. Previous value forward (PPV) was used as the baseline that all methods should beat. He found that the best models were LSTM with RMSE 9.09, Feed Forward Neural Network (FFNN) with RMSE 44.89. The rest of the models performed worse than PPV predictor (RMSE 44.96). (Falk, 2018.)

2.4 Common methods in regression

This section describes methods that are commonly used in regressions tasks. Regression is a set of statistical processes that can be used to estimate an outcome variable based on features. Regression is used in prediction and forecasting. ("Regression analysis", n.d.) A study by Kuremoto, Hirata, Obayashi, Mabu, and Kobayashi (2019) explored the effects of different training methods for artificial neural networks. Backpropagation (BP) was compared to stochastic gradient ascent (SGA) which was shown to improve performance when predicting certain variables. The main difference between SGA and BP is that BP is a supervised method while SGA is a reinforcement learning method. The paper showed a deep belief network (DBN) that can be used for time-series prediction, see Figure 6. (Kuremoto et al., 2019).

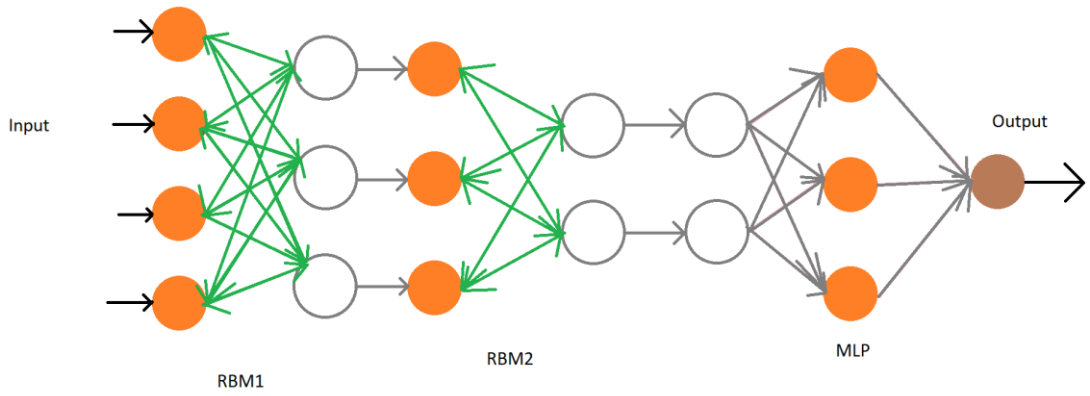

Figure 6. Deep belief network (DBN) for time-series forecasting (Kuremoto et al., 2019).

There are many time-series prediction methods other than neural networks. For instance, Autoregressive integrated moving average (ARIMA) is used in many cases (e.g. Zhang, 2003; Wu, Zhu, Li, & Wu, 2017). It has been shown that ARIMA models are lightweight and can complete time-series regression tasks quite accurately (Wu et al., 2017). Wu et

al. (2017) discovered that ARIMA model trains usually faster than even a simple neural network model. Zhang (2003) discovered that a hybrid model with combined ANN and ARIMA could improve the prediction accuracy when compared to a situation where only one of these methods is being used. Zhang (2003) mentioned that the combined models should ideally be dissimilar to each other, in order to make the prediction error and variance smaller.

Evitan (2019) built eight different regressors to predict housing prices. They used correlation analysis between features to determine which features to feed to the regressors. In addition, the distributions of values between the train and test sets were explored, and some values whose distributions are different in training and test sets may cause the trained model to overfit. Another problem was that certain values were systematically missing from the data. Some sections of the data had outliers that behaved very differently from the rest, so they were left out. The best regressor had RMSE of 0.1079. Meanwhile, the combined regressor produced an RMSE of 0.0640, which is much better than any single regressor. The best results from individual repressors were produced by Elastic net, Lasso, and Ridge methods. (Evitan, 2019.) The study by Evitan (2019) highlights the benefit of using combined regressor to increase the prediction accuracy.

The format in which inputs are fed to the neural network influences the predictive performance. Kumar and Goyal (2013) used Principal component analysis (PCA) to generate inputs to the neural network. This was compared to scaled inputs for each raw variable. It was found that the inputs that were created using PCA gave consistently better prediction results than simply feeding scaled inputs to the network. (Kumar & Goyal, 2013.) This study highlights the importance of properly scaling the inputs and generating proper features.

Aggarwal, Kirchmeyer, Yadav, Keerthi, and Gallinari (2019) attempted to predict housing prices using neural networks. According to the authors: Gaussian Process (GP), Deep neural network (DNN) and Boosted trees are the state-of-the-art models for regressions problems. The study used GP, Boosted trees, DNN (MLP based), and Conditional Generative Adversarial Network (CGAN). They used multiple hidden layers in their MLP network that had up to 100 neurons on each layer. They found that ELU activation function worked better than ReLU and Leaky ReLU on the network when there were three hidden layers in the network. ReLU worked better on the 13-layer network. On the final layer, they used sigmoid activation. Adam optimizer was used in this study, and Batch Size was set to 100. The experiment was run on 2.7 GHz processor. The authors did not notice any significant speed-up when using Tesla M40 GPU to train the network. It was found that Boosted trees gave generally the best results, but they could not find a method that would give the best result across all different datasets. CGAN is competitive with state-of-the-art methods, and it gave the best Negative Log Predictive Density (NLPD) score on two of the used datasets. (Aggarwal et al., 2019.)
The use of ReLU activations on the majority of the layers by Aggarwal et al. (2019) is consistent with my previous experiences regarding neural networks. ReLU doesn't suffer from vanishing gradients nearly as much as sigmoid activation does, which is probably why ReLU was used on the majority of the layers.

2.5 Limitations of prior research

This section describes the limitations of prior research related to regression. There are several limitations in existing methods. Most of the existing methods use basic neural networks. Hyperparameter optimization and network structures, were in some cases not discussed in detail, which makes it difficult to replicate the results from those studies. The sampling frequency in some studies was low, anywhere from 15 minutes to 1 hour. Very few papers in general, discussed CO2 prediction using neural networks.

The comparability of these studies is challenging. The sampling intervals were different. In most cases, the studies predicted a different number of minutes to the future. Studies used various metrics to calculate the errors (discussed later). Also, the lack of a benchmark dataset that everyone could use is also somewhat problematic and makes it difficult to compare results across different studies.

Some papers had trouble getting enough high-quality data. For instance, a study by Zuraimi, Pantazaras, Chaturvedi, Yang, Tham and Lee (2017) had data with one-hour sampling interval. This is not frequent enough for demand-controlled ventilation. Khazaei et al. (2019) predicted the future values of CO2 five minutes to the future. The network structure was rather basic, one hidden layer with four neurons. Early stopping prevented overfitting. (Khazaei et al., 2019). Early stopping is a method for preventing the model from overfitting of the model. Essentially the training is halted once validation error starts to increase. (Prechelt, 1998.) This ensures that the model does not overfit on the training data and performs better on unseen data. Kasche and Nordström (2020) compared several regularizations methods on MNIST and CIFAR-10 dataset and found that dropout regularization is better than early stopping. Dropout regularization works by randomly dropping units (and their connections) from a neural network while it's training (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Another study by Srivastava, Mansimov, and Salakhudinov (2015) found that dropout regularization achieves better results than L2 Regularization. This means that the results of Khazaei et al. (2019) study can potentially be improved by using the dropout regularization method, and by using more than four neurons in the hidden layer.

Very few papers focused on future value prediction for CO2 using neural networks. The papers that were found usually had one or two hidden layers. Sometimes the hyperparameter optimization process was not described in high enough detail to make the study repeatable. In some cases, the exact network structure, and sometimes the neuron counts were described vaguely. This makes it difficult to reproduce the results in the absence of a generally accepted benchmark dataset.

Any of these details could result in improved accuracy. According to Pantazaras, Lee, Santamouris, and Yang (2016) there are still very few papers related to CO2 level prediction. Wei et al. (2019) studied 37 papers related to predictive models regarding indoor air quality. None of the 37 regression models that were studied had more than two hidden layers. Sometimes zero hidden layers were used. (Wei et al., 2019.) A similar observation was made during the prior research phase of this thesis. Most of the network structures related to CO2 level prediction had very few hidden layers in general.

Comparing the results obtained from different articles is rather difficult. The reason for this is the fact that the used data can vary wildly among different papers. For instance, the CO2 concentration in a classroom can reach higher levels than in office rooms since there are usually more occupants there. Candanedo and Feldheim (2016) also found that getting

the data from earlier research is problematic. This rises another potential problem for the comparability of the studies due to the differences in the data.

In prior studies regarding CO2 prediction, there are several sampling intervals that are used. For instance, Zuraimi, (2017) and Pantazaras et al. (2016) sampled once per hour. Meanwhile, Khazaei et al. (2019) used a one-minute sampling interval. The issue exists for the prediction horizon as well. Khazaei et al. (2019) predicted five minutes to the future; Falk (2018) predicted 10 minutes to the future. In some cases, predictions were made up to 60 minutes to the future, for instance Pantazaras et al. (2016). Due to various sampling intervals and prediction horizons, it's difficult to compare the results.

Data cleaning and selection methods can also affect the results. For instance, Falk (2018) noticed that weekends had very little variations in CO2 levels, which is why data from weekends was excluded. Most studies did not discuss the data selection methods, which could mean that all the data was used. The data selection method will also affect the comparability between different studies.

Another problem when attempting to compare results in time-series articles is that they may use different metrics in order to measure accuracy. Aggarwal et al. (2019) used MAE and NLPD; Falk (2018) used RMSE, R-squared (R^2), and F1-Score; Macarulla et al. (2017) used standard error (SE); and finally, Khazaei et al. (2019) used mean squared error (MSE). The following metrics seemed to be the most common MAE, MSE, and RMSE.

A potential issue with time-series prediction papers is, that there isn't an agreed-upon benchmark dataset to test against. For instance, in the field of Image recognition, it is common to test the model against CIFAR10, CIFAR100, or ImageNet datasets. For time-series prediction, a popular dataset like this could not be found. Some less-known datasets were used. For instance, Aggarwal et al. (2019) used many datasets such as CA-housing dataset. Meanwhile, Kuremoto et al. (2019) used CATS dataset. A dataset that would be used by everyone could not be found. So the best way to evaluate the models between different articles would be to consider the best models in each study, instead of trying to cross-compare the results between studies that are not fully comparable with each other due to the reasons stated in this section.

3. Methods and Materials

In this thesis, a neural network predictor is built that can predict CO2 level in indoor air. The Design Science Research (DSR) approach is used to build this network. This chapter describes how the DSR approach is followed in this thesis. This chapter is divided into two sections. Section 3.1 describes the guidelines for conducting DSR. Section 3.2 describes how DSR was applied in this thesis.

3.1 Design science research (DSR)

DSR is a design methodology that aims to find answers to relevant human problems. This process contributes new knowledge to the body of scientific evidence. In DSR, the knowledge and understanding regarding the problem and its solution are found during the creation and use of the artefact. (Hevner & Chatterjee, 2010.) In other words, the researcher does not know the exact structure of the design artefact until after the artefact has been built. The guidelines for conducting DSR research are shown in Table 1.

Table 1. DSR Guidelines (Hevner, March, Park, & Ram, 2004)

#	Guideline	Description
1	Design as an Artefact	Design Science research should generate a feasible artefact. This artefact must be in the form of a construct, method, a model, or an instantiation.
2	Problem Relevance	Provide a technology-based solution for a relevant business problem.
3	Design Evaluation	The usefulness, quality, and efficacy of the artefact must be demonstrated via evaluation methods.
4	Research Contributions	Clear and verifiable contributions should be provided in terms of design artefact, design foundations, and design methodologies.
5	Research Rigour	Rigorous methods should be used during the construction and evaluation of the design artefact.
6	Design as a Search Process	The use of available means to reach the desired outcome, while the laws of the problem environment are satisfied.
7	Communication of Design-science Research	DSR should be presented effectively both for technology-oriented and management-oriented audiences.

The problems that DSR method is trying to solve are characterized by unstable requirements; constraints based on environmental contexts that are not well-defined; complex interaction among subcomponents of the problem and solution; flexibility to change the prior processes and artefacts; reliance on cognitive abilities (e.g. creativity) to create a solution; and teamwork to produce effective solutions. (Hevner et al., 2004.) Documenting the artefact development process is a core part of DSR (Hevner & Chatterjee, 2010).

3.2 Use of DSR in this thesis

The research process for DSR was formed by an earlier paper by Peffers et al. (2007). They identified six activities for producing DSR (Peffers et al., 2007). These activities are visible in Figure 7.

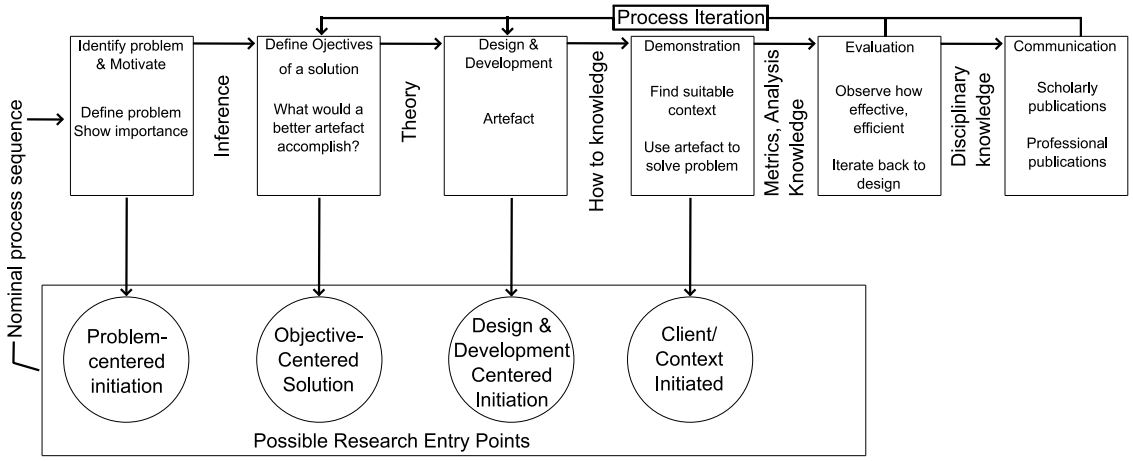

Figure 7. The iterative nature of DSR research (Peffers et al., 2007).

The artefact in this work is a neural network predictor that can predict CO2 values up to 10 minutes to the future. Six activities discovered by Peffers et al. (2007) and their use in this thesis are described next.

Activity 1: Problem identification & motivation

In this stage, the specific research problem is defined. The motivation for controlling indoor air ventilation are reduced energy consumption, wellness, and better job performance. These motivations were discussed in length in section 1.1. The specific problem that is solved in this thesis is to predict CO2 in indoor air using neural networks up to 10 minutes to the future.

Activity 2: Define the objectives for a solution

The objective is to find an effective neural network structure that can predict CO2 levels in indoor air (RQ1). Ideally, the model should outperform existing neural network models. Another objective is to find out what kind of training data one should use (RQ2), and if shortcut connections should be used in the network (RQ1).

The second research question regarding training data selection will be experimented by comparing the same network with different inputs. One network receives all data during training, while the other one only receives those areas, where CO2 value fluctuates significantly. The second experiment related to training data was to find out if data from other rooms helps the prediction accuracy when one is predicting data from just one room or is it better to just use data from that one room.

Activity 3: Design and development.

The design of the artefact was done iteratively. Early iterations were related to training data selection, activation functions, and what type of neural network should be used: MLP, LSTM, GRU, or CNN. The hyperparameter optimization was done using a grid search. Each of these models had similar parameter counts to ensure that the other did not

perform better simply because of having more parameters. Once the best performing model was found, it was then further tuned, by attempting to use more complex network structures such as skip connections (shortcuts). The following iterations then further tuned the hyperparameters (such as features) of the best performing network. MLP network showed good results in the early stages of the study and was chosen to be tuned further. The hyperparameters were then further tuned, across several experiments. Each improvement that were found during the experiment would be used as a starting point for the next experiment. For instance, if N input steps were found to produce the best results, the next experiment would then use N input steps for tuning the next hyperparameter.

Activity 4: Demonstration.

To demonstrate that the neural network model can predict future values of CO2, the network was first trained using a small dataset from a single room. At a later stage, the full dataset from all rooms was used to train the network, to see if including data from other rooms would improve the accuracy of the model.

Various hyperparameters were tested: learning rate, learning rate decay, input time steps, and others. The full list of hyperparameters is visible in Appendices G, H, I. Chapter 5 below discusses the iterations of the hyperparameter tuning process, and also describes the networks structures that were tested.

Activity 5: Evaluation.

Comparison to previous papers regarding CO2 prediction is challenging due to the reasons stated in section 2.5. Due to this, the comparison is done instead by creating baseline models that represent the earlier literature. These baseline models are trained with the same dataset as the proposed new models, which means that baseline models and proposed models can then be compared. Baseline models are neural network with one hidden dense layer (MLP1), line fit (LF), and previous value forward prediction (PPV). Here MLP1 is similar to what was used in most of the earlier research. MLP1 gives us a comparison point when attempting to compare the effectiveness of the networks that were built in this thesis, to those that were used in earlier studies.

To demonstrate that the proposed network produces good predictions, RMSE error was calculated between real value and the predicted value. In addition, also MAE and MSE errors are shown to make it easier to cross-compare results between different publications. However, RMSE score was the determining error criteria based on which all the parameters were chosen. Accuracy measures, data selection, data cleaning, data selection, and feature engineering are discussed further in chapters 4 and 5. The fact that the developed network can predict future values better than earlier models used in the literature demonstrates the effectiveness of the network. The RMSE score is lower compared to all the baseline models: MLP1, Line fit, and PPV. These results are visible in section 5.6.5.

Activity 6: Communication.

The prediction of CO2 levels in indoor air is an important problem because it helps demand-controlled HVAC system to respond faster to changes. This makes it possible to keep CO2 within a reasonable range, which in turn keeps workers more comfortable and makes them more productive while also accruing monetary benefits compared to a ventilation system that is not demand-controlled. Network structures and training parameters were made available to the reader, which I find to be an improvement over

previous research, where networks structures were not always readily available. Training data and codes are not shown since they are owned by VTT.

This study was published in JULTIKA repository, which contains publications that originated from the University of Oulu. The citation of this thesis follows the APA guidelines. The structure of the DSR method closely follows the activities that were suggested by Peffers et al. (2007). General structure of the thesis is based on the faculty of Information Processing Science guidelines in Oulu University.

4. Implementation

This chapter presents the proposed method for predicting CO2 level. To predict CO2 the following four neural networks were used: MLP, CNN, LSTM and GRU. In addition, the method for cleaning and data selection are also discussed. Furthermore, some feature engineering is used in order to find the best performing features.

This chapter is organized in the following manner. Section 4.1 describes the dataset that was used. Section 4.2 explains how the data was cleaned and selected. Section 4.3 presents the network structures that were used. Section 4.4 introduces the hyperparameters that were used. Section 4.5 describes how feature engineering was done. Section 4.6 discusses the criteria based on which the fitness of each model is determined.

4.1 Dataset

This section describes the dataset acquisition process. The dataset was gathered in VTT's Oulu office in 2019 from January until May. The data collection phase was done by other VTT employees as a part of the SCOTT project. The dataset was gathered from thirteen rooms of different sizes. Senseair K30 sensor provided the measurements for CO2 using one-minute sample interval. Other sensors that measured CO2 were a device called "MCF-LW12CO2" and a sensor called "SENSEAIR LP8". These both had a sampling frequency of 15 minutes. Table 2 describes the details of each sensor. Each row in the column is related to either a device or to a sensor. The "measured value" column describes the variable(s) that the device is measuring. The "sensor/device" is the marketed name for the product. Sampling interval describes how often each sensor measures the listed variables. The last column details the source for the product details.

Table 2. Details about sensors and measurement devices.

Measured value	Sensor / Device	Sampling interval	Source for product details
Temperature (°C), relative humidity (%)	Silicon Labs Si7021	1 min	Silicon Labs (n.d.)
Air pressure (Pa)	Bosch BMP180	1 min	Bosch Sensortec GmbH (2020)
PIR (Integer value between 0-12. 0 = no movement)	Panasonic EKMB1301113K	1 min	Panasonic (n.d.)
CO2 (ppm)	SENSEAIR K30	1 min	"K30" (n.d.)
CO2 (ppm), relative humidity (%), Brightness (lux), pressure (Pa), temperature (°C)	MCF-LW12CO2	15 min	mcf88 (n.d.)
CO2 (ppm)	SENSEAIR LP8 (inside MCF-LW12CO2 device)	15 min	"LP8 pin headers" (n.d.)
Noise (dB)	PeakTech PT8005	5 sec. Time weighting: slow (1s).	PeakTech (n.d.)
door state (0=open, 1=closed)	Reed switch (Manufactures by Guard)	1 min	SP-Elektroniikka (n.d.)

CO2 sensors were placed 1.1 meters above the ground level. This sensor placement is similar to an earlier study regarding CO2 level prediction, where sensors were placed 1.25 m above the floor (Macarulla et al., 2017). The sensors are on the same wall as the outflow vent is located, but not directly next to the outflow vent. The goal was to place the sensor in a location where the air is well mixed. Following variables were measured from each room: CO2, relative humidity, pressure, temperature, brightness value (Lux), volatile organic compound (VOC), noise (dB), door state (open vs closed), and the amount of movement in the room (PIR).

4.2 Data cleaning & Selection

This section describes how data was cleaned and selected. The structure of this section is the following. Section 4.2.1 describes how data was cleaned. Section 4.2.2 discusses the way in which data was selected. Section 4.2.3 explains how data was split into training, testing, and validation data.

4.2.1 Data cleaning

This section describes the steps that were taken to clean the data. The data cleaning consists of three steps: 1. Data re-alignment, 2. outlier detection, 3. dealing with outliers.

First, let's discuss the re-alignment of the data. The measurement interval for the devices was not precisely 60 seconds, which means data was misaligned. Due to the misalignment, the data was re-aligned to match the 60-second sampling interval using linear interpolation. Two past time-steps and one new time steps were used to fit a line. Then the line fit equation was used to interpolate the CO2 value to match the 60-second sampling interval. This is also visible in Figure 8, where raw values are used to fit a line, and the re-aligned value is a result, that was found by linear interpolation, based on the raw values. This type of re-alignment was done for humidity, temperature, pressure and CO2. Other variables (Lux, PIR, door state, VOC, noise) were re-aligned simply by rounding them to the nearest full minute, without using linear interpolation. Lux, PIR, door state, and noise can all change from a small value to their maximum value very quickly. Because of this linear interpolation was not used. VOC was sampled only once per 15 minutes, which means that linear interpolation could result in large errors since the measured raw values are so far from the re-aligned values.

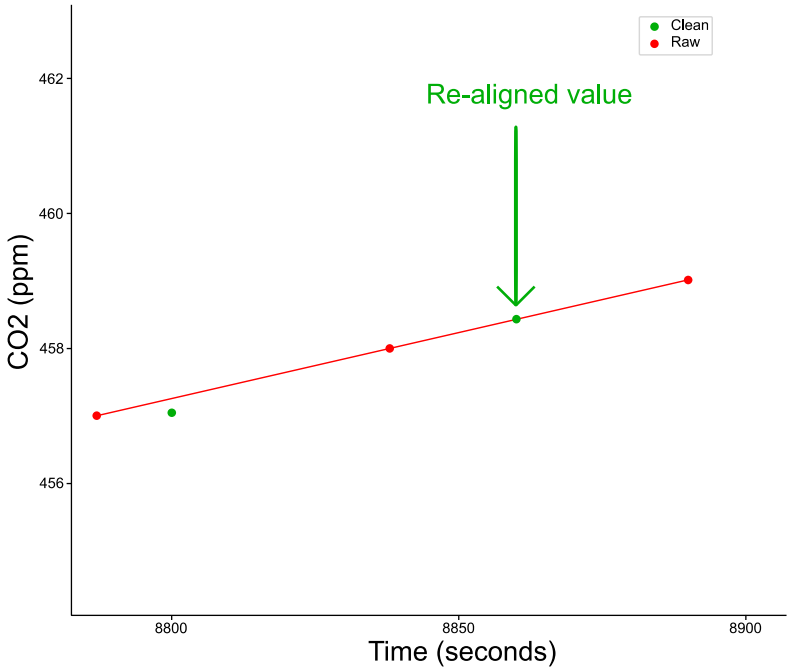

Figure 8. Re-alignment using linear interpolation.

Next, let's discuss the way in which outliers were detected. Most of the changes in CO2 levels are almost zero since rooms are unoccupied for large portions of time. Due to this, it would not make much sense to clean the data with a sliding windows anomaly detection. This is because once someone enters the rooms and CO2 starts to rise, then that risen value would be an outlier compared to rest of the CO2 values that have almost no change in them because there was nobody in the room.

A two-step process was used to detect outliers. First, if the measurement value, for instance, CO2 concentration, was too high or too low, then the value is an outlier. Second, if the rate of change for the measured variable is too high, or too low, then the value is an outlier. The first outlier type is called a Global outlier, while the second type is a Contextual outlier (Cohen, n.d.). In this study, Kernel Density Estimation (KDE) is used to determine the thresholds for contextual outliers. In earlier work by Dai, Song, Sheng, and Jiang (2017), KDE has been used in data cleaning.

Limits for both outlier types were set in order to programmatically clean the data from outliers. The limits for global outlier detection were chosen to exclude the clear outliers that don't match typical office conditions. For instance, 130 dB noise levels would be excluded. The limits (for rate of change) for contextual outliers were determined using KDE.

Table 3. Cleaning settings for global outliers

Variable	Min	Max
Temperature (°C)	0	40
Relative Humidity (%)	0	99
Pressure (mb)	950	1200
CO2 (ppm)	300	5000
Noise (dB)	30	130
Lux (cd)	0	2000
PIR	0	12
VOC (ppm)	0	500
Door State	0	1

KDE was done on the rate of change values. KDE is a non-parametric way to build a variable's probability density function ("Kernel density estimation", n.d.). If the probability density falls below a certain threshold (occurs too rarely), then it's an outlier. See Figure 9 below to get an idea how this was done. The vertical orange lines in the figure determine the rate of change that is considered too high; the values near zero are inliers (x-axis). By looking at the highest probability density of CO2 in the chart, one can notice that it peaks above 0.8, and quickly declines on both sides. This means that most of the time CO2 levels have remained stable or the changes have been small.

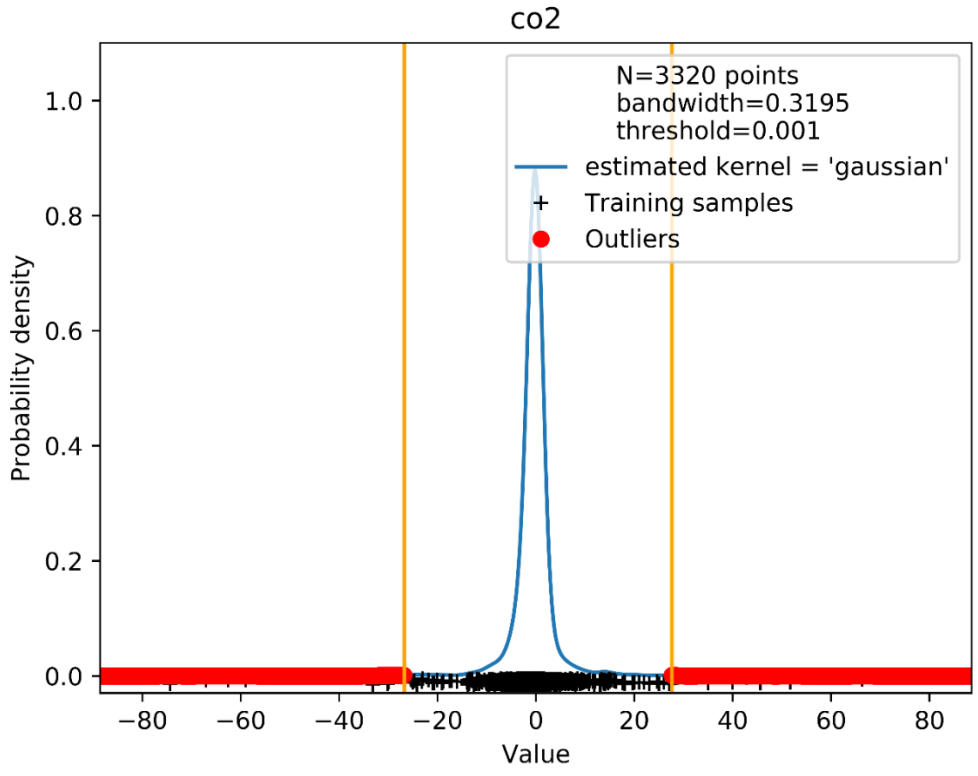

Figure 9. Kernel density estimation for CO2.

Some variables such as Lux, Noise, PIR, and State can change from their minimum value to their maximum value in an instant which is why they were only cleaned using their absolute values. In other words, only CO2, humidity, temperature and pressure were cleaned using the rate of change approach (contextual outlier). All parameters were cleaned using the global outlier detection limits described in Table 3.

Once data from each sensor had been cleaned, they were then combined with other data from the same room. If there were more than one sensor that measured the same variable, then the median of the values was used. As an example, if there were three CO2 sensors, then the median value of all three CO2 values were used. Median was used because it can potentially reduce the number of measurement errors if there are at least three sensors of each type in the room. If certain room didn't contain all nine measured variables, then data from that room was not used in the study.

4.2.2 Data selection

Next, let's discuss the data selection methods that were used. The first selection method aims to find out if training data should only contain sections with high variation. The second selection method attempts to determine if data from other rooms would improve the prediction results. Table 4 contains a brief description of both data selection methods that were experimented on. These data selection methods are then discussed further in the rest of the section.

Table 4. Data selection methods

Data selection method	Explanation
"var"	Only data sections where CO2 levels have high variation are included in this.
"all"	All possible data was used, even sections that have long periods of time, where CO2 remains near 415 ppm.

The selection method for variating data is discussed next. This experiment aims to find out if it's better to include all possible data for the training or just the data that has high variation in it. See Figure 10 to see a typical behaviour of CO2 levels in indoor air. It should be noted that most of the data consists of long, flat, "uneventful" sections, where variation is minimal.

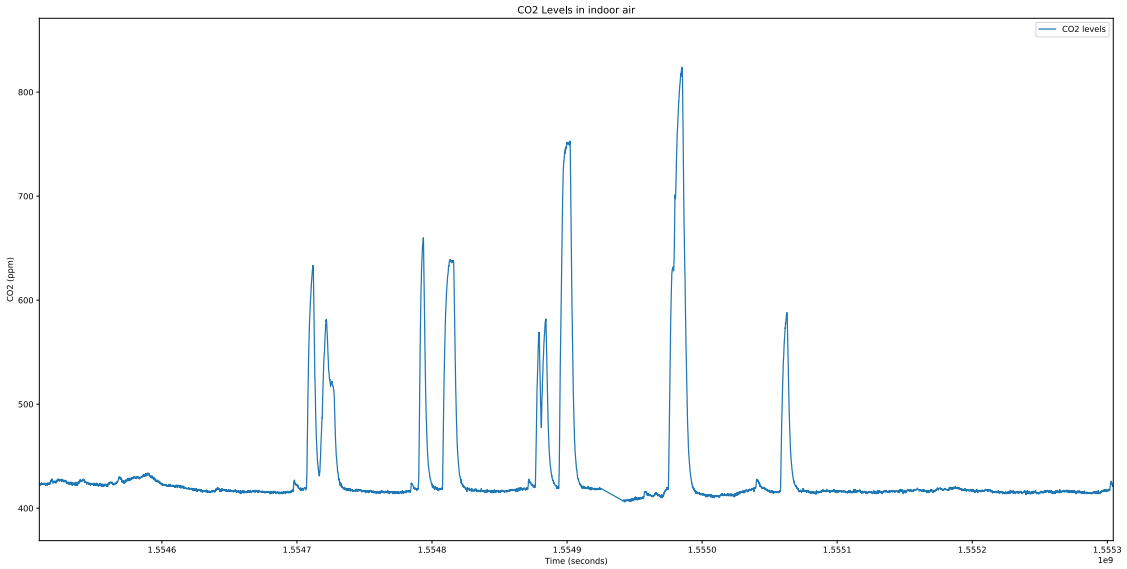

Figure 10. Typical cleaned CO2 data

My assumption was that including these uneventful sections in the training data will not improve the prediction accuracy. I expected the predictor to only learn to repeat the previous value since the CO2 level remains at ~415ppm most of the time. In order to test this assumption, the uneventful sections were excluded from training and then the training was repeated, by including all the data and finally comparing these two results. The experiment results regarding data selection, are explored in section 5.3.1. The data selection process is discussed next. Data selection process is visualized in Figure 11.

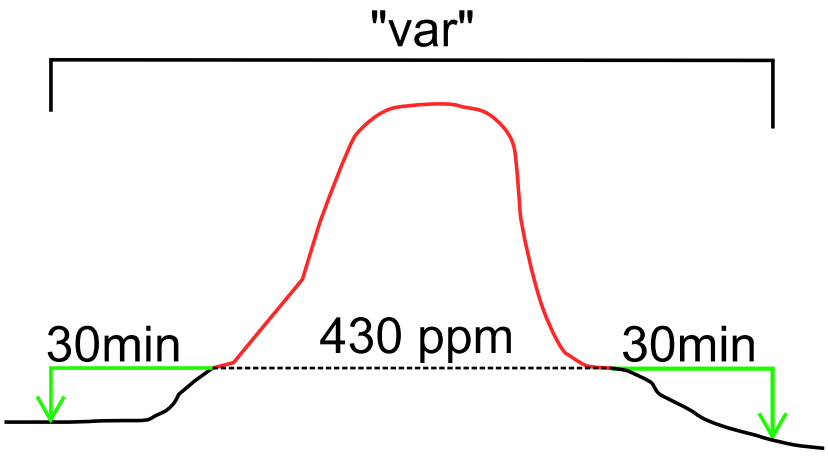

Figure 11. Training data selection

The high variation sections were chosen by first finding the sections where CO2 is above 430 ppm (red section in the image), then that section was extended by 30 minutes to a direction (green arrow in the image). If the CO2 value was still above 430 then the sections were extended by another 30 minutes to that direction until CO2 value was no longer above 430 at which point the section was cut. This was then repeated in the other direction. This ensures that the training data captures the transition from low CO2 to high, and back from high to low while excluding the data sections where CO2 value remains near 400 for extended periods of time. In addition, any section whose length was less than 40 minutes was excluded. The main interest in the study is to predict CO2 values in sections where there is a high variation of CO2 since that usually indicates that someone is in the room and ventilation may be needed. This means, that in most cases validation

and test datasets only contained variative sections of the data. This makes it easier to determine if the built models operate well on the variative data. The performance that was observed when training with all data (even sections with low variations) data was good, but the relative differences between various predictors became very small. In order to highlight the model's ability to predict variating sections, only the variating sections of the data are focused on.

Another data selection method was related to the training data, and if including data from other rooms would reduce the prediction error. Test data was separated in such a way that it only contains 20% of the data from one room. Then the rest of the data was used for training (64%) and validation (16%). Figure 12 visualises the way in which data was split to training, val, and test sets.

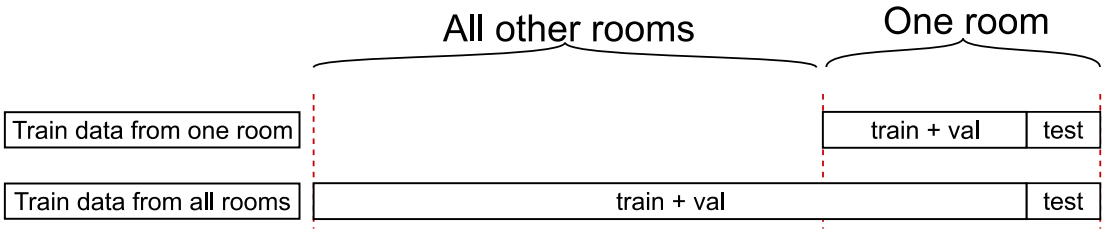

Figure 12. Data selection process for training data.

Both data selection methods predicted the same instances of the target room, the only difference was the training data that was used. In Figure 12 the upper selection method only uses training data from the same room one is trying to predict. Meanwhile, the lower selection method uses all available training data, even if it's not from the same room.

4.2.3 Train, validation and test data

Training data selection was done in the ways, described in section 4.2. The amount of training instances depends on which room was used. In most cases, Room A was used because it had a lot of training data. Table 5 depicts the training set sizes for all data that was used. In the table "_all" means that all data was included, while "_var" means that only sections with high variation are included. The values in Table 5 represent how much training, testing, and validation data each of the rooms contained. The amount of training instances varies slightly with each test. This is because when the history length changes, one gets either more or less training instances. Longer histories produce less training instances.

Table 5. Used training data.

Room name	Train_all	Train_var	Validation_all	Validation_var	Test_all	Test_var
ALL ROOMS	423389	139453	101621	34252	135136	40955
Room A	52549	15732	22337	6769	30997	10323
Room B	30515	11904	11704	4953	11629	4347
Room C	38380	16822	10967	4513	17796	11568
Room D	36696	7529	11100	2101	13425	3924

Training data was used to train the model, and validation data was used to tune the hyperparameters. Test set is a separately held out section for the data. This section is not

used for training model, adjusting hyperparameters, nor is it used for scaling of the values either. Test set is simply used to determine the fitness of a model. This is known as the "holdout cross-validation" method. Test set in this study always contains the last 20% of the data from a single room. This method was used at all stages of the study.

N-fold cross-validation was not used because it would cause training times, and related analysis times to increase too much. For instance, 3-fold cross-validation would cause training times to triple. Some experiments took up to 4 days of constant computing to finish and analyse. By using three-fold cross-validation it would take up to 12 days, which is far too long and would have meant that I could not keep up with the schedule that was laid out by VTT. Hold out cross-validation was used instead.

4.3 Proposed Neural Networks

This section describes the proposed neural networks for CO2 prediction, which was one of the research goals (RQ1). Various neural networks are discussed in section 4.3.1. Networks with various depths are explored in section 4.3.2. Experiment regarding widening the deep network is discussed in section 4.3.3. Shortcut connections are described in section 4.3.4.

4.3.1 Neural Networks used

This section describes the proposed neural networks, and the building blocks they were built from. The exact network configurations for each of the four networks are also shown. There are four building blocks that are used in this thesis: Dense unit for the MLP network, Long short-term memory (LSTM) node for the LSTM network, Gated recurrent unit (GRU) for the GRU network, and convolutional layer for the CNN.

LSTM and GRU networks used data from five previous minutes, that were fed to the network. Meanwhile, MLP and CNN networks took the previous ten minutes of data as the input. All the networks have one dense unit on the last layer. To make the results comparable across each network, the trainable parameter count for each network is set to ~11k by altering the number of neurons in the hidden layers of the networks. The feature set that each network used was the scaled CO2 value along with a difference set of CO2. These features are described in detail in section 4.5. Figure 13 contains all the networks that were used at this stage of testing.

The MLP network consists of five Dense layers, followed by an activation. The neuron count per layer was set to 49. A two per cent dropout layer was added before the last hidden Dense-49 layer, which randomly drops two per cent of the neurons from the previous layer. Dropout

CNN re-used most of the hyperparameters from MLP network. The main difference between hyperparameters was, that CNN had 35 hidden units instead of 49. In CNN, Downsampling of feature vectors was done by increasing the stride. Similar downsampling method was also used by He et al. (2016). For CNN, some studies had good results with dilated convolutions. In this test adding dilation to the network caused worse results, which is why dilation was not used in the CNN. The kernel size of the network was set to 1x2, which means it's a one-dimensional convolution.

The proposed LSTM network contains 34 LSTM nodes following the input layer, and four Dense-34 layers. Similarly, to the MLP network, a two per cent dropout was added before last the hidden dense layer. The built GRU network is almost the same as the LSTM network, but since GRU is simpler than LSTM unit, the neuron counts were increased to 37. Both LSTM and GRU models were trained in stateless mode.

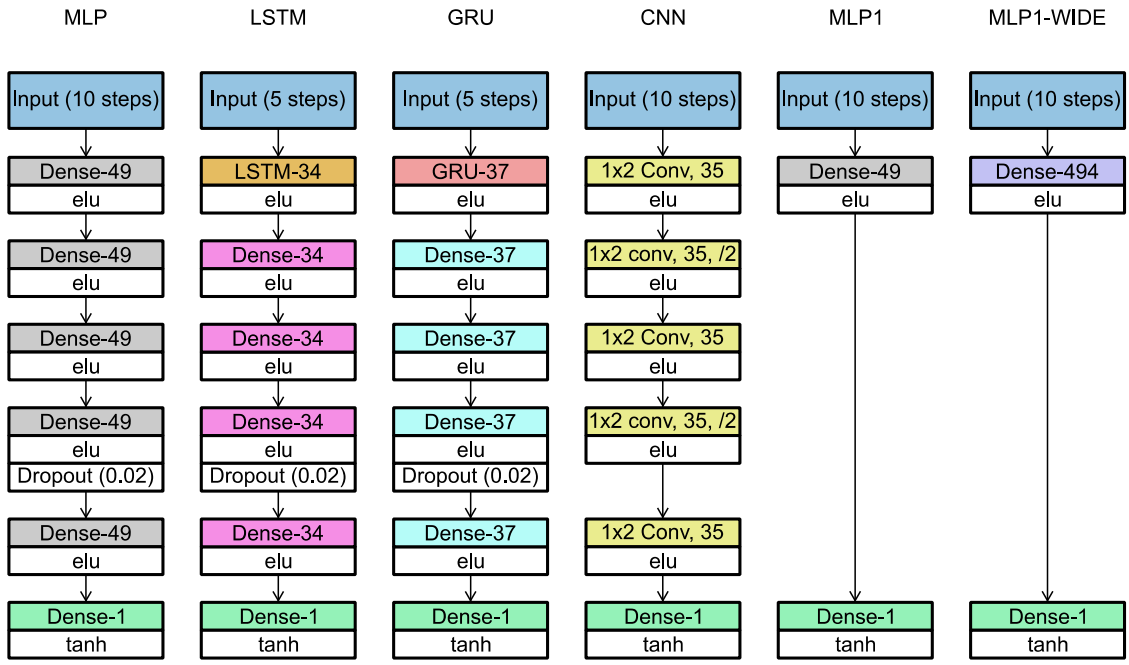

Figure 13. Networks that were tested.

In addition to proposed networks, four baseline models were included at this stage of testing: line fit using with 2 inputs, line fit with 3 inputs, previous value forward (PPV), and MLP1 which is a multilayer perceptron with one hidden layer. Line fit algorithms are linear regression models, where n previous input steps are used to plot a line. This is then extrapolated 10 minutes to the future, and the predicted value is then picked from the 10-minute mark. A total of two, line fit algorithms were used. The first one used two input steps, and the other three input steps. Various other line fit histories were experimented on, but two- and three-minute histories gave the best results. The line fit algorithms were kept the same for the whole duration of the study.

Dropout layers are only active during training. All the networks presented in this study only contain one dense unit on the last layer. This last dense unit then predicts CO2 value 10 minutes to the future.

4.3.2 Network depth

Another example of a structure that was studied in this thesis was, whether the network should be wide or deep. First, a deep network was built. Then a shallow model was built that had the same amount of trainable weights as the deep network. Finally, the results were compared. To test if deep of wide network is better the deep model was first built, then a wide model called MLP1-WIDE was created. The MLP1-WIDE has one hidden layer, and the wideness of the network was increased until it had the same number of parameters as the deep model.

4.3.3 Widening the deep network

This section describes the networks used when experimenting if its beneficial to make the deep network into a wider version. The networks used in this experiment are visible in Figure 14.

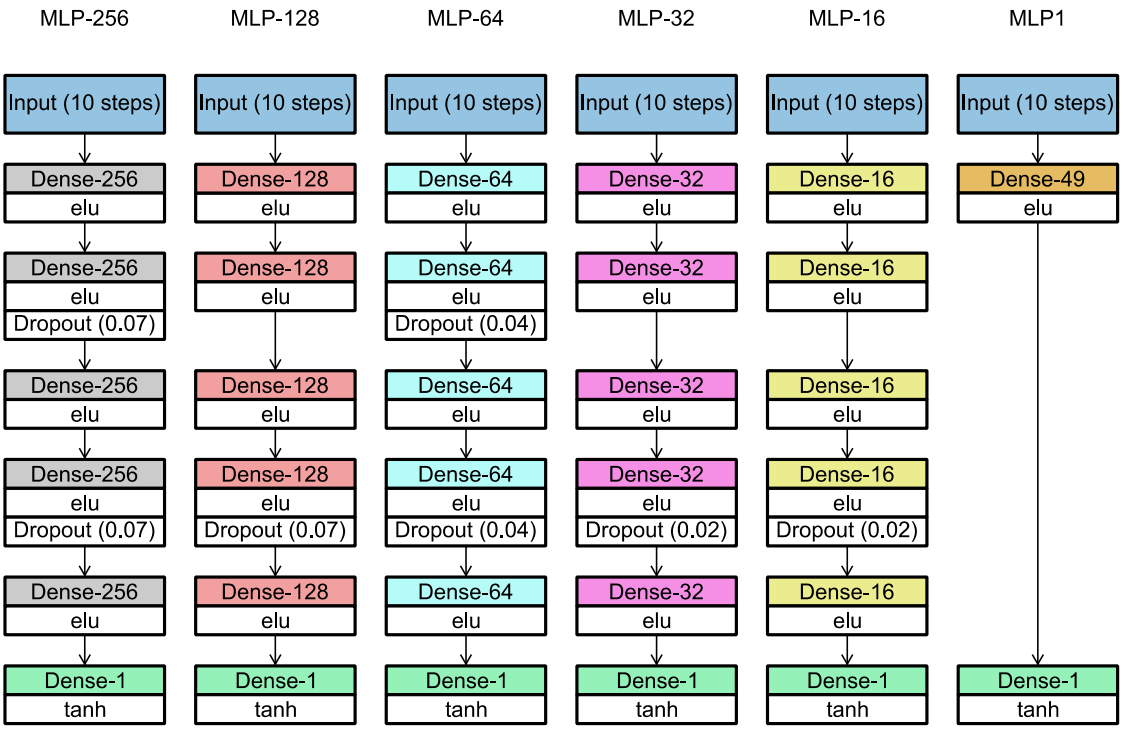

Figure 14. Various MLP networks

This experiment alters the number of dense units on each layer to five different values: 16, 32, 64, 128, 256. The models are trained with various dropout values to determine what is the best amount of dropout for each model. Once the best dropout rates were found for the validation data, then the test RMSE scores of each network were compared.

4.3.4 Shortcuts

This section describes what shortcut connections are, and how they were used. An example regarding the structures of a neural network can be seen below in Figure 15. Shortcut connections were found to be effective in earlier work by He, Zhang, Ren, and Sun (2016).

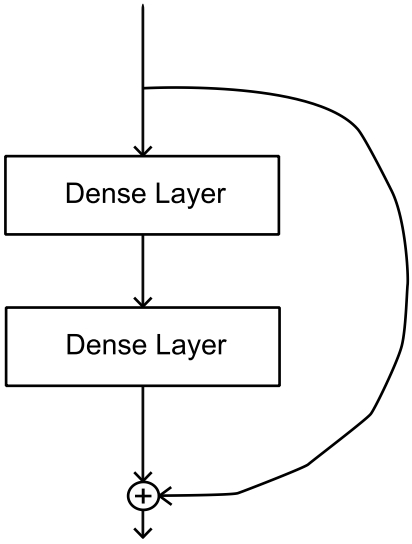

Figure 15. Shortcut connection. (He et al., 2016).

Shortcut connection is a connection that does not, directly connect to the next layer, but instead skips one or more layers. The shortcut connection is either addition or concatenation. In "addition shortcut", the values are added. In "concatenation shortcut" the values are stacked side by side. For instance, if the start of the shortcut contains 64 output, and it is combined with 64 outputs, then the result would be 128 outputs after the concatenation. This type of shortcut doesn't require the combined paths to have the same width. In additive shortcuts, equal width is required, and the resulting output has the same number of outputs as the combined paths had on their own. In "addition shortcut" combining 64 outputs with 64 outputs the resulting output shape has 64 units.

The network structures used in this experiment are visible in Figure 16 and Figure 17. In the figures "CS" stands for concatenation shortcut, "AS" stands for add shortcut, "BN" for batch normalization, and "LN" for layer normalization. For each network, various dropout values were tested. For the shortcut networks, various locations for the dropout node were experimented.

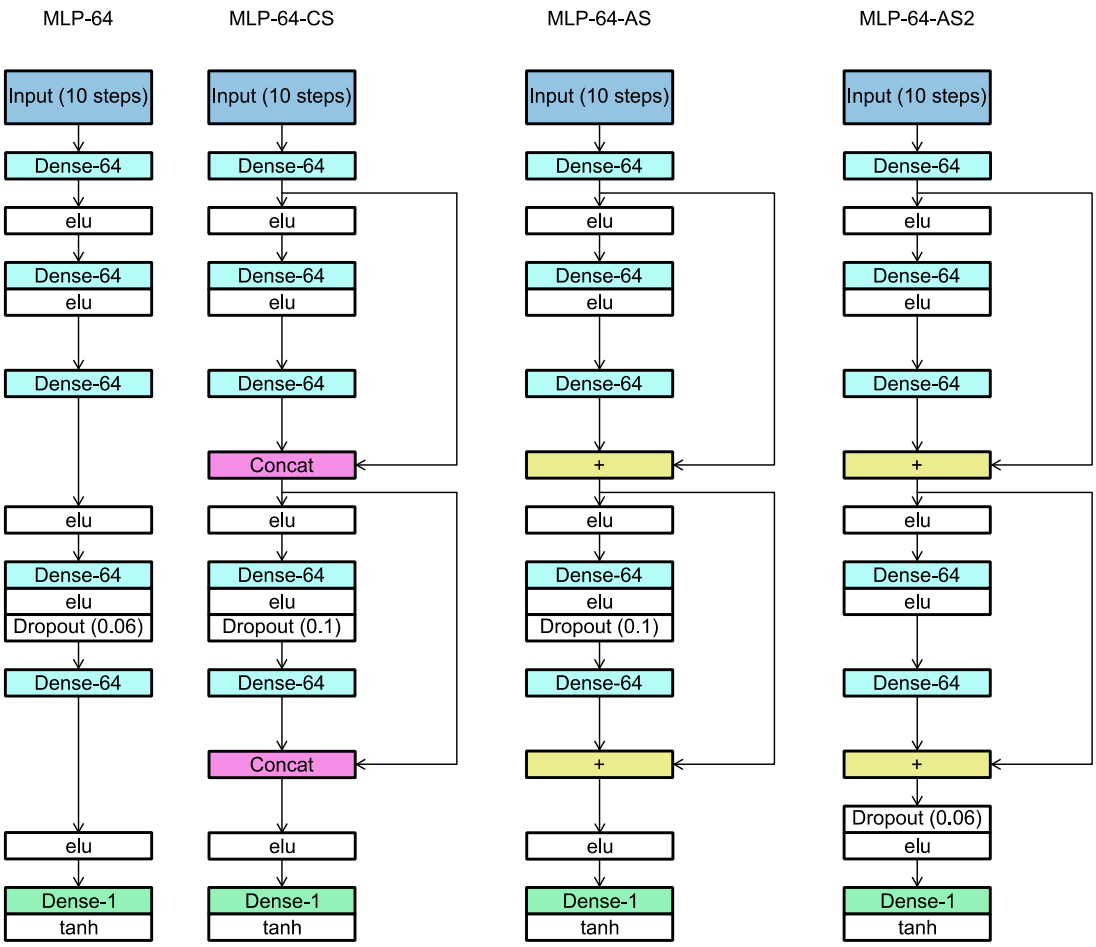

Figure 16. Shortcut networks

Normalization layers were placed before the shortcut reconnects to the main branch. I found this to be effective for ResNet-20 model for CIFAR-10 dataset in the past.

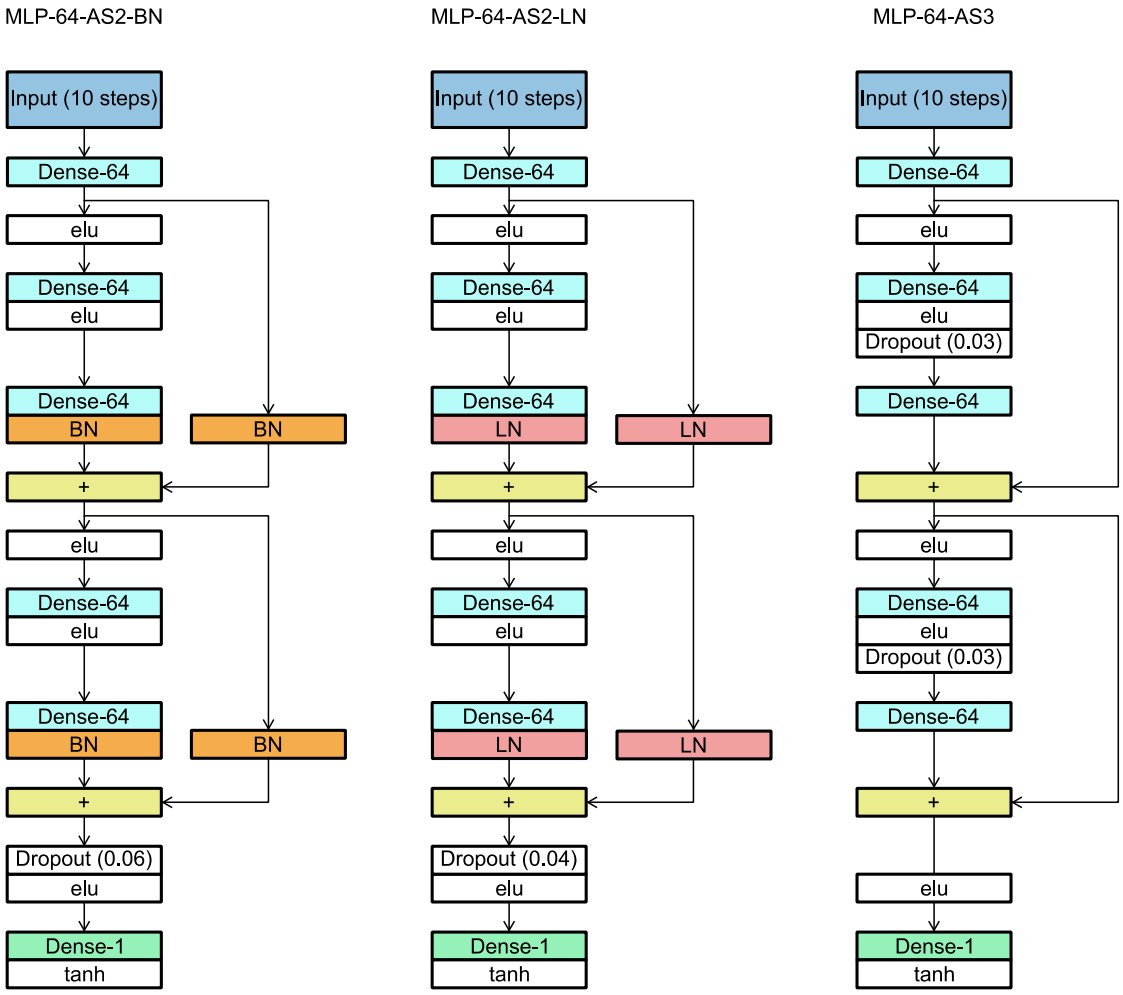

Figure 17. Shortcut networks. BN = Batch Normalization, LN = Layer normalization

The "MLP-64-AS2" variants have dropout right before the last dense layer. This dropout placement was inspired by earlier research by Szegedy, Ioffe, Vanhoucke, and Alemi (2017).

4.4 Hyperparameters

This section gives a brief overview of the hyperparameters used: activation function, learning rate, and other hyperparameters. Two separate activation functions were being searched in this thesis. The main activation, and the activation that is placed after the last layer. The main activation is used in most parts of the networks. Last activation is only used on the last layer of the network. Figure 18 displays the positioning of the activation functions in the network.

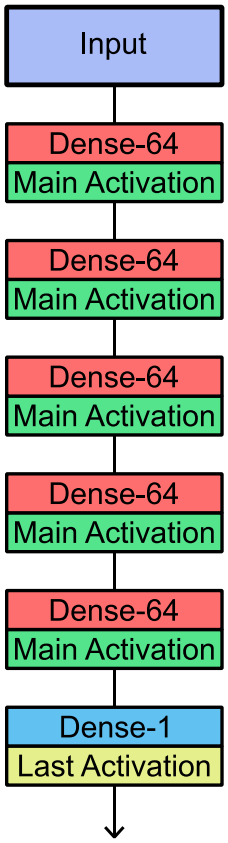

Figure 18. The location of activation functions.

In total 23 activations were tested. The full list of activation functions can be seen in Appendix F.

This thesis uses an adaptive learning rate in the training process. This means that after N epochs the learning rate was reduced multiplying the existing learning rate by X if there hadn't been any improvements in the error. Essentially X is a value between 0-1, which means that learning rate gets reduced once it's multiplied with X. The full list of various Hyperparameters can be found in Appendices G, H, and I.

4.5 Feature engineering

This section describes feature engineering. All nine measured variables could each have six types of features. Let's first assume history length is 7 minutes, in the experiments, the history length varied between 1 and 30 minutes. Because 60-second sampling interval there will be seven values for each variable using 7-minute history.

The features are the following: scaled, minimum (MIN), maximum (MAX), median, average (AVG), difference (diff). Scaled feature is simply a scaled variable, that is scaled using scikit-learn packages' RobustScaler function. There would be seven scaled values for each variable when history length was seven minutes. MIN, MAX, median, and AVG features were calculated using the scaled values from that seven-minute history length.

The "diff" feature is calculated by calculating the difference between each measured variable. More specifically by subtracting the previous minute's measured value from the current minute, which gives us six difference values. When the difference set was used, the history length was initially increased to eight, to make sure than we get seven differences. If a scaled feature was used at the same time as the difference feature, then

the oldest scaled value was removed. This ensures that both difference, and scaled values to contain the same amount of inputs. Once difference and scaled features had been calculated, then the MIN, MAX, median, and AVG features were calculated based on the scaled values.

Timestamp was used to build the following features: minute, hour, IsWeekday. Timestamp was converted from UTC to local time before it was used. Only the most recent timestamp value was used, even if the history was longer than one minute. Minute feature was calculated by extracting the minute from the timestamp and then scaling the value using RobustScaler. The hour feature was created in a similar manner as the minute. IsWeekday is a number which is either one, or zero. IsWeekday is zero when it's either Saturday or Sunday, otherwise, it's one.

To help to decide which features to use, correlation matrices were built. In Figure 19 one can see the correlation matrix between different variables. All correlation matrices used in this thesis were built by using the seaborn package. In Figure 19 the correlations between each variable was calculated to see how well each variable correlates with the current CO2 level. Movement sensor (pir_cnt) has the highest correlation here, followed by volatile organic compound (voc), and noise had the third-highest correlation. Door state has a weak negative correlation with CO2 levels.

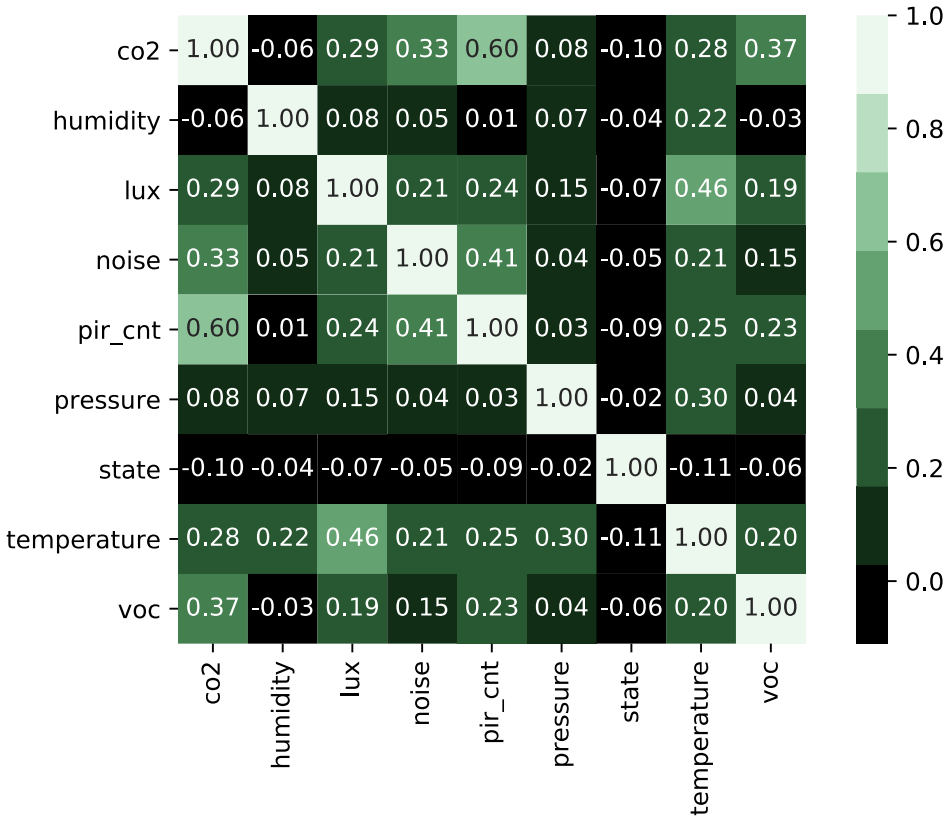

Figure 19. Correlation between variables

Perhaps a more interesting correlation is between crafted features and CO2 value ten minutes in the future, which were calculated to guide the feature selection process (Appendices A-E). The following variables were excluded as features due to low absolute values for correlation: pressure, temperature, humidity. VOC had high correlation; however, it was only sampled once per 15 minutes, which means that VOC data is delayed, and may reduce the accuracy. Even though CO2_diff had low correlation, it was

included as a feature because it gave good results at an earlier stage of experimentation. In other words, the following features were chosen for experimentation: Co2, Co2 diff, PIR, noise, lux, isweekday, hour, and door state, VOC. Various combinations of these features were tested, and the full list of feature sets can be seen in section 5.6.5. In an earlier stage of experimentation, only scaled CO2 and CO2 diff had been used for all four network types: GRU, LSTM, CNN, MLP. However, at a later stage, only MLP networks were used to experiment with features. Figure 20 contains the MLP networks that were used at the last stage of experimentation with various features sets. Other network types aside from MLP, had already been excluded at an earlier stage.

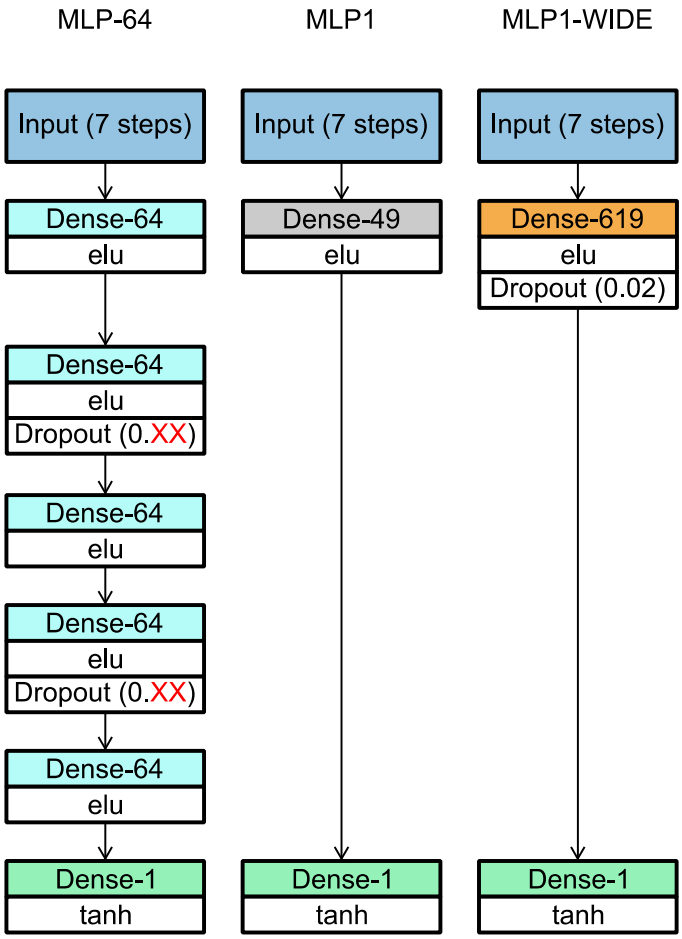

Figure 20. Networks used for feature engineering.

The dropout values for MLP-64 were dependent on the used feature set. The dropout values for each feature set can be found in Appendix M.

4.6 Testing methodology

Neural network training has some randomness involved in it. A network can have anywhere from a few parameters up to billions. Due to this, there are many local minima where the training can get stuck, which causes different results on different training sessions. To reduce the effects of randomness each test was repeated anywhere from 3 to 40 times and the average was then calculated. Usually, 40 times was chosen, but in some instances, this had to be reduced due to the large number of parameters that were explored. For instance, activation functions had to be initially tested 3 times to weed out those activations that clearly performed worse than the others. The networks were then sorted

based on their average error. In addition, statistical significance levels were calculated using Bartlett's test. Bartlett's test calculates if the variance between two groups are equal. If variances were equal, then one can use Student's T-test, otherwise, Welch's T-test should be used. The p-value was then read from either the Student's test or from Welch's test. By utilizing this methodology, one can be relatively sure that the chosen hyperparameters are the best ones and not caused by randomness. Data was split three ways: first, the 20% of the data was moved to the test set. The remaining 80% was split to training and validation sets by 80/20 split, which resulted in 64% of the total data in the training data (0.8*0.8), and 16% (0.2*0.8) to the validation data. The data selection is visualized is discussed in section 4.2.2. Hyperparameters for each model were tuned based on the variating sections for the validation data. The deciding factor, regarding which model is best was then done based on the RMSE error of the test data.

The model fitness was evaluated with the variating data sections (as discussed in section 4.2.2). This is because for the occupants it's more important that the ventilation control works well when someone is in the room, and variating sections should therefore have priority.

5. Results and Findings

This chapter presents the experiments that were performed, and the results of those experiments, along with the findings of this study. Several experiments were done during the experimentation phase. This chapter describes various experiments which aim to improve the performance at each stage. The emphasis in the study is the find the optimal hyperparameters, that should be used in the designed neural network. This means that p-value is used to determine the best hyperparameters. The magnitude of the difference is not a major concern until at later stages where all the best hyperparameters have been found and the model is then compared to earlier studies.

Random seeds were not fixed in this study. I discovered this strategy too late, which meant that it had to be excluded to meet VTTs deadlines. Due to this, there can be small differences in results when the same network is trained on the same dataset. However, this shouldn't affect the conclusions since each network was trained ~40 times, and p-values were used to determine if the difference could be dependent on randomness (or random initializations). This gives us a good certainty that results are valid while making the reproduction of the study slightly more difficult.

This chapter is organized in the following way. The sections are shown in the order in which the experiments were done. Each experiment builds upon the results of the earlier experiment. Software and hardware tools are discussed in section 5.1. Evaluation method is discussed in section 5.2. Data selection methods are discussed in section 5.3. Activation functions are discussed in section 5.4. Various network configurations are discussed in section 5.5. Experiments done for the MLP network are discussed in section 5.6. The findings are summed up and compared to earlier research in section 5.7.

5.1 Software, and hardware tools

This section describes the software and hardware tools that were used. First, let's discuss software tools. The programming language in this thesis was Python. The main tool for building and training the neural network is the Keras package that is included in TensorFlow v2. Data cleaning is done with the help of the following modules: scikit-learn, NumPy, pandas, and matplotlib.

The hardware tools are the following. The networks were trained using Intel Core i7-7600U CPU @ 2.80GHz. Training was also attempted on Tesla P100-PCIE-12GB GPU, but the training took twice as long when compared to CPU. Even with larger batch sizes the GPU still spent more time per epoch. Aggarwal et al. (2019) had the same observation in their study.

5.2 Evaluation criteria

This section describes how model fitness was determined. Hyperparameters for all the models were tuned based on the validation data. Once the hyperparameters had been found, the resulting model was then evaluated based on its RMSE error on the test data. RMSE was chosen because almost all previously mentioned CO2 prediction papers displayed RMSE as one of the measurements (Macarulla et al., 2017; Falk, 2018; Pantazaras et al., 2016). MSE and MAE errors are shown as well.

Error measurements are discussed next. It should be noted that all networks developed in this thesis minimize the MSE score which means that RMSE and MSE scores are prioritized over MAE. MAE error is calculated by averaging the absolute differences between the real and the predicted value. MSE error is calculated by averaging the squared differences between the real and predicted value. RMSE is a square root of MSE. The equations are shown below. Equation 17 displays ow MAE is calculated. Equation 18 depicts how MSE is calculated. Equation 19 shows how RMSE is calculated.

$$\mathrm{MAE} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\left|\mathrm{Y_i} - \widehat{\mathrm{Y}}_\mathrm{i}\right| \quad (17)$$

$$\mathrm{MSE} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\left(\mathrm{Y_i} - \widehat{\mathrm{Y}}_\mathrm{i}\right)^2 \quad (18)$$

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\left(\mathrm{Y_i} - \widehat{\mathrm{Y}}_\mathrm{i}\right)^2} \quad (19)$$

Comparison to previously used neural network structures is not directly possible. This is because of the reasons mentioned in section 2.5 (different training data, various sampling intervals, various prediction horizons, etc.). Due to these problems a baseline MLP model was built with 49 neurons in the hidden layer. Using 49 neurons resulted from using the same neuron counts as the deep MLP model that had 49 neurons on each layer (section 5.5). MLP1 is a simplified version of the deep model since MLP1 only contains one hidden layer. Using one hidden layer for the MLP1 model was decided because most other studies used one hidden layer. Khazaei et al. (2019), Zuraimi et al. (2017), and Falk (2018) all used one hidden layer. I found one paper that used two hidden layers: Skön et al. (2012).

MLP1 acts as a proxy model that is a close estimate compared to what most literature regarding CO2 prediction has used in the past. By this, we can be sure that the proposed models perform better than earlier research, and the improved performance isn't just the result of having easier data to predict. The full structure of the baseline model "MLP1" is visible in Figure 13. In addition, MLP1-WIDE will have the same parameter count as the developed deep network. The MLP1-WIDE is also a good representation of earlier research since it has the same parameter count as the models in this thesis while having a similar structure to what was used in earlier CO2 prediction papers. At the last stage: feature engineering (section 5.6.5) there is a comparison between the best deep model compared to MLP1, and MLP1-WIDE, which should give the reader a summary on how the deep model compares to earlier research.

5.3 Training data selection

This section describes how training data should be selected and is closely related to RQ2. Section 5.3.1 discusses if one should include only the training data, where there is a significant variation in CO2 levels. Section 5.3.2 discusses the effects of adding more training data from other rooms.

5.3.1 Included training data

The purpose of this experiment is to find out if it's enough to just train the network with the variating sections of the data. Roughly 60%-70% of the data contains sections, where CO2 value remain close to 415 ppm. My assumption was, that these low variation sections

don't improve the prediction accuracy. It would speed up the training speed per epoch if these sections can be omitted from training.

The setup for the experiment was the following. All data that was used, was originated from Room A, and the model that was used was MLP-49 that was discussed earlier in section 4.3.1. Fifteen networks were first trained by using all of the training data. Then another fifteen networks were trained using only the variating sections of that training data (as described in section 4.2.2).

The results are the following. MAE score for the test data was 7.651ppm when training with variating sections. MAE score was 7.493 when using all the available training data, which is slightly better. The p-value between these two experiments was 1.15 * 10^-15, which means statistically very significant, and this indicates that it's best to use all the available data for training. Table 6 below shows the results of this experiment.

Table 6. Network performance with various training data selection methods.

Room A		**Dataset size**					
Input Time Steps	**Data selection method.**	**Train set size**	**Test set Size**	**AVG TRAIN MAE**	**AVG VAL MAE**	**AVG TEST MAE**	**p-value compared the best method. Calculated for test MAE.**
16	All data for training. Variative data for validation and test.	~50k	6 176	2.510	3.181	**7.493**	N/A
16	Only variating data for each dataset.	~15k	6 176	6.930	9.874	7.651	1,15E-15 ***

The training took ~2 minutes when using all the data with 50k training instances. This resulted in 2*40 = ~80-minute training time across 40 networks. Training only with "var" data there is some performance degradation in the MAE score, but the model trains much faster. Due to this, it may be beneficial to tune hyperparameters using the "var" data, and only later use all the possible data, which would enable faster experimentation. However, in this study 80 minutes wasn't considered too long, which is why all data was used in the next experiments.

It can be concluded that it's better to use all the available data for the final model since ~2-minute training time is rather short. At this stage of experimentation, the decision to use RMSE had not yet been done, therefore the errors that were measured at this stage were done using MAE.

5.3.2 Training data from other rooms

The purpose of the experiment is to find out if adding more training data from other rooms helps to reduce the prediction error further. The setup for the experiment was the following. The experiment was repeated for three rooms: Room A, Room B, and Room C. The used neural network was the MLP-49 discussed in section 4.3.1. This model was trained using data from that same room. Then the same network was trained by using data from all rooms. This was done a total of 40 times for each room and for both data selection methods. The results are visible in Table 7.

Table 7. Training data selection. One room VS many rooms.

Test data from	Train data from	AVG TEST MSE	AVG TEST RMSE	AVG TEST MAE	p-value for TEST RMSE	
Room A	Room A	**131.59**	**11.47**	7.33	4.07E-22	***
	ALL ROOMS	134.07	11.58	**7.22**		
Room B	Room B	**348.66**	**18.67**	**11.8**	8.59E-30	***
	ALL ROOMS	363.86	19.07	12.15		
Room C	Room C	**38.22**	**6.18**	**3.37**	1.71E-37	***
	ALL ROOMS	40.14	6.34	3.57		

It was found that Room A, Room B, and Room C all had better RMSE scores when only training with data from that same room was included. P-values were statistically very significant in each case which would indicate that it's better to always use data from the same room when training the networks. The magnitude of the improvements range between 0,11 - 0.4 ppm (1-3 %). In this case, the training times when including all rooms was about ten times as high when compared to just using one room. Because using single room had better performance and trained significantly faster, it's better to use training data from the same room that one is predicting.

5.4 Activation functions

The purpose of this experiment is to find out which activation functions should be used in the network. The section is closely related to RQ1.1. The setup for the experiment was the following. Training the network 40 times using all 23 activations would take a long time, and I wouldn't be able to keep up with the schedule laid out by VTT. Due to this, each network was initially trained only a few times. This made it possible to quickly weed out those activation functions, that resulted in significantly higher errors. After this initial step, the most promising activation functions were then used to train the network 40 times for each activation. Two separate experiments were done at this stage. First one was to find out what is the best main activation (as discussed in section 4.4). The second experiment related to activation functions aimed to find out the best activation for the last layer.

The most promising main activations were ELU, SWISH, Mish, GELU, rrelu, and lisht. Table 8 contains the results of the experiment. The difference between the best four activations ELU, SWISH, MISH, GELU is relatively small which is also visible in the p-values. I decided to choose ELU activation as the main activation because it has the best RMSE score. Mish activation had essentially the same based on the p-value, but since MISH takes longer to train according to Misra (2019), it is better to use ELU.

Table 8. Experiment results for main activation.

Room A				Data Selection method: all = all data used. var = only data with high variation was used. var	var	var	
Input Time Steps	**Model**	**Main activation**	**Last activation**	**AVG TEST MSE**	**AVG TEST RMSE**	**AVG TEST MAE**	**p-value for TEST RMSE**
10	MLP-49	elu	tanh	**136.022**	**11.663**	7.309	N/A
10	MLP-49	swish	tanh	136.525	11.684	**7.222**	0.046 *
10	MLP-49	mish	tanh	136.558	11.686	7.233	0.075
10	MLP-49	gelu	tanh	136.674	11.691	7.237	0.011 *
10	MLP-49	rrelu	tanh	137.311	11.718	7.372	1.07E-04 **
10	MLP-49	lisht	tanh	141.145	11.879	7.559	9.22E-10 **

Next, let's discuss the results regarding the last activation. The best performing activations after the initial test showed that SWISH, MISH, tanh, and GELU activations to be the best ones. The results are shown in Table 9. Here the differences between various activations were very small. P-values indicated that there is no statistically significant difference between those four activations (on the last layer).

Table 9. The last activation function.

Room A				Data Selection method: all = all data used var = only data with high variation was used. var	var	var	
Input Time Steps	**Model**	**Main activation**	**Last activation**	**AVG TEST MSE**	**AVG TEST RMSE**	**AVG TEST MAE**	**p-value for TEST RMSE**
10	MLP-49	elu	swish	**135.216**	**11.628**	**7.265**	N/A
10	MLP-49	elu	mish	135.471	11.639	7.283	0.484
10	MLP-49	elu	tanh	135.581	11.644	7.292	0.276
10	MLP-49	elu	gelu	194.841	12.588	7.921	0.326

It should be noted that GELU activation had one very bad experiment run, where test RMSE went to 50 ppm. This raised the test RMSE much higher than the rest. Tanh had worked in one of my earlier experiments and did not produce anomalous readings like GELU did. Tanh activation also has comparable performance with SWISH and MISH activations. Therefore, tanh was the selected activation function.

5.5 Experimenting on various neural networks

The purpose of this experiment was to find out what type of neural network works best for predicting CO2 values. The section is closely related to RQ1.1. The setup of the experiment was the following. First four types of neural networks were trained: LSTM, GRU, CNN, and MLP. After this the baseline models were trained: line fit with two previous CO2 values, line fit using 3 previous CO2 values, MLP1, and PPV.

The experimental setup was the following. First four networks with similar parameter counts were built. Each of these networks had different network configuration. The proposed networks: MLP, LSTM, GRU, and CNN were built to contain 5 hidden layers and 11k parameters. The parameter counts of each network was matched to 11k by adjusting the wideness of the networks. This way results between each type of network is comparable due to the similar parameter counts. The matching parameter counts are also visible in the "Params" column of Appendix J. The network structures are described in section 4.3.1, and full lists of the used hyperparameters are visible in Appendices G, H, and I. Once the best model was found, then the next step would be to determine if it's better to use a deep model or a wide model. In the initial experiment, deep MLP outperformed the other models. Because of this, a wide MLP model (MLP1-WIDE) was built, which had one hidden layer and ~11k parameters.

The results of the first experiment are the following. The deep MLP model outperformed LSTM and CNN models. The difference was statistically significant. Difference between deep MLP and GRU model was very small, and the difference was not statistically significant. All the developed deep models outperformed the baseline models. The results of these experiments are visible in Table 10. Full results table is too large to fit on a page. Due to this, the complete results are visible in Appendix J. Deep MLP network had the best RMSE performance (11,55 ppm on variative data), but GRU also performed well (11,67 ppm). MLP was chosen for the following iterations due to its good performance and simplicity.

Table 10. Results of testing various network types (Room A).

		Data Selection method: all = all data used var = only data with high variation was used.						
		all	**var**	**all**	**var**	**all**	**var**	
Input Time Steps	**Model**	**AVG TRAIN MAE**	**AVG TEST MAE**	**AVG TRAIN MSE**	**AVG TEST MSE**	**AVG TRAIN RMSE**	**AVG TEST RMSE**	**p-value for TEST RMSE**
10	MLP	**2.679**	7.290	**47.750**	**136.011**	**6.910**	**11.662**	N/A
5	GRU	2.898	7.353	52.006	136.271	7.211	11.673	0.443
5	LSTM	2.956	7.255	52.950	137.032	7.276	11.706	2.38E-03 **
10	MLP1-WIDE	3.018	**7.198**	54.645	140.084	7.392	11.836	2.65E-33 ***
10	CNN	3.095	7.230	55.578	140.181	7.455	11.840	1.24E-18 ***
3	Linefit	3.381	7.516	57.289	147.883	7.569	12.161	-
2	Linefit	3.333	7.739	59.433	146.337	7.709	12.097	-
10	MLP1	3.430	7.564	61.046	148.759	7.811	12.196	9.14E-24 ***
1	PPV	3.354	9.080	79.927	194.461	8.940	13.945	-

The best performing model from an earlier stage was MLP. Now a wide model was built, as described in section 4.3.2. Both models had now ~11k trainable parameters. The result was that MLP with 5 hidden layers had better RMSE score (11,662) when compared to a shallow network with only one hidden layer (11,836). The difference was statistically very significant (p-value: 2,65E-33). The results are visible above, in Table 10.

5.6 Experiments done with MLP network

This section describes the various experiments that were done using the MLP network. In section 5.5, the MLP network was chosen to be developed further, which is why other network types were not used anymore.

The format of this section is the following. Section 5.6.1 discusses the implications of increasing the width of the deep network. Shortcut connections are explored in section 5.6.2. The effect of having less training data is observed in section 5.6.3. The amount of input time steps a network should take is discussed in section 5.6.4. Feature sets are discussed in section 5.6.5.

5.6.1 Making the deep network wider

The purpose of this experiment is to find out if it's worthwhile to add more neurons to the hidden layers. The section is closely related to RQ1.1. The setup of the experiment is the following. First, five deep MLP networks were built. These networks were described earlier in section 4.3.3. Each of these networks were trained 40 times and then compared to each other. The baseline models are the same as in the earlier experiment (section 5.5).

The results of this experiment are visible in Table 11. The full table with all possible details is visible in Appendix K. MLP-256 had the best test RMSE score, followed by MLP-64. It should be noted that MLP1 had a slight performance increase at this stage. The difference can be explained by the fact that learning rate was slightly higher (0.00075 compared to 0.00015) and learning rate was reduced at a slower pace (once per 5 epochs instead once per 3 epochs).

Table 11. Effects of the wideness of the deep MLP network.

	var = only highly variating data			
	var	**var**	**var**	**p-value for TEST RMSE**
Model	**AVG TEST MSE**	**AVG TEST RMSE**	**AVG TEST MAE**	
MLP-256	**134.445**	**11.595**	7.277	N/A
MLP-64	134.78	11.609	7.286	0.318
MLP-128	135.284	11.631	7.277	0.006 **
MLP-32	136.185	11.67	7.33	1.08E-07 ***
MLP-16	137.661	11.733	7.326	1.20E-14 ***
MLP1	142.116	11.921	**7.273**	1.74E-40 ***
Line fit	146.337	12.097	7.739	
Line fit	147.883	12.161	7.516	
PPV	194.461	13.945	9.08	

The p-value between MLP-64 and MLP-256 was 0,318. This means that there is no meaningful difference between the model accuracies. It should be noted that the dropout value for MLP-128 was not as precisely tuned as it was for MLP-256 and MLP-64, which could explain why it's doing worse than MLP-64. All the developed models performed better than the baseline models. MLP-256 contained 268 801 parameters, while MLP-64 contains 18 049. MLP-256 takes roughly ten times as long to train compared to MLP-64 while having essentially the same performance. Because of this MLP-64 was chosen to be used for the next experiment.

5.6.2 The use of shortcut connections

The purpose of this experiment is to find out if it's worthwhile to use various shortcut connections in the network. The section is related to RQ1.2. The setup for the experiment was the following. First, various networks were built. These networks were described earlier in section 4.3.4. Each of the networks was trained 40 times using data from Room A, and then results were compared.

The results of adding shortcut connections to the network are visible in Table 12. The full table is visible in Appendix L. The results for MLP-64 are different compared to an earlier phase of the study. This could be a result of using dropout nodes a different way during the training phase. For the shortcut experiment, one dropout node was used with 6% dropout (see Figure 16). For the other experiment where deep network was widened two dropout nodes were used with 4% dropout each (see Figure 14).

Table 12. Results of adding shortcut connections.

	Data Selection method: all = all data used var = only data with high variation was used.			
	var	**var**	**var**	**p-value for TEST RMSE**
Model	**AVG TEST MSE**	**AVG TEST RMSE**	**AVG TEST MAE**	
MLP-64	**134.947**	**11.617**	7.277	N/A
MLP-64-CS	135.368	11.635	7.256	0.12
MLP-64-AS3	135.657	11.647	**7.255**	0.013 *
MLP-64-AS2	136.089	11.666	7.292	1.76E-06 ***
MLP-64-AS	136.417	11.679	7.296	2.11E-04 ***
MLP-64-AS2-LN	139.646	11.817	7.452	1.11E-25 ***
MLP-64-AS2-BN	140.216	11.841	7.433	4.38E-21 ***
MLP1	142.116	11.921	7.273	2.04E-44 ***
Line fit	146.337	12.097	7.739	-
Line fit	147.883	12.161	7.516	-
PPV	194.461	13.945	9.08	-

The best performing model was MLP-64 with no shortcut connections. Shortcut networks with normalization layers had the worst RMSE scores, but still better than the baseline models. MLP-64-CS had essentially the same performance as MLP-64 since the difference was not statistically significant. This means that it is better to use the MLP-64 model since it has less parameters and gets the same performance. At this stage, all models

had better performance than MLP1, which is the proxy model that represents earlier research.

5.6.3 Using less training data

This experiment aims to find out what is the effect of having less training data. The section is related to RQ2. The percentage of training data that was used was 5, 10, 20, 40, 80, and 100% of the total usable training data from Room A. The amount of test data remained the same in all cases. For each of these, various dropout values were tested, and the best dropout values were found based on the validation RMSE. The full results of this study are visible in Appendix O. The short version of the results is visible in Table 13 and Figure 21.

Table 13. Effects of using less training data.

		Data Selection method: all = all data used var = only data with high variation was used.			
	Dataset size	var	var	var	p-value for TEST RMSE
Model	Train size	AVG TEST MSE	AVG TEST RMSE	AVG TEST MAE	
MLP-64	45 498	**134.947**	**11.617**	7.277	N/A
MLP-64	27 242	135.384	11.635	7.221	0.217
MLP-64	36 352	136.06	11.664	**7.207**	1.12E-04 ***
MLP-64	18 155	143.016	11.959	7.349	1.35E-39 ***
Line fit	46 104	146.337	12.097	7.739	-
Line fit	46 104	147.883	12.161	7.516	-
MLP-64	9 047	150.08	12.25	7.498	3.36E-28 ***
MLP-64	4 532	180.433	13.429	8.236	1.50E-32 ***
PPV	46 104	194.461	13.945	9.08	-
MLP-64	2 271	201.185	14.159	8.781	2.75E-21 ***

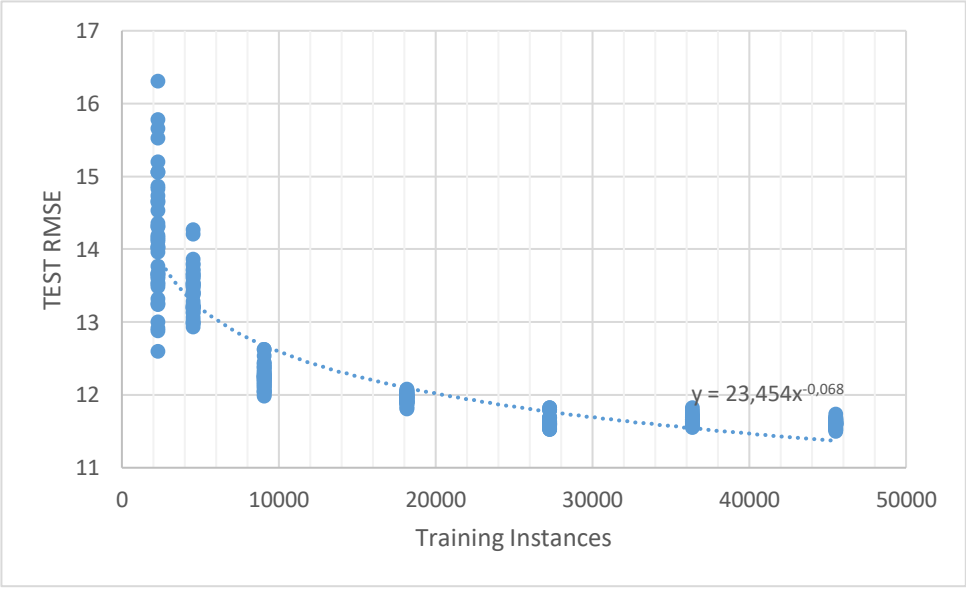

Figure 21. TEST RMSE as a function of training set size.

In this experiment, it was found that having more training data was better. The used model (MLP-64) had ~18.5k parameters. In this experiment, it was found, for the MLP to beat line fit, there should be at least as many training instances as there are parameters in the network. When there was less training data than parameters, then line fit was better. The best results were achieved by using 100% of the data and. In this case, the number of training instances was 2.52 times as high as the parameter count of the network. The performance degradation is very noticeable when the training set is small. The difference between worst and best results for the MLP-64 had 2,542 ppm difference, which is much higher than the difference between various models in earlier experiments. Due to this, the model must have enough training data, otherwise the results will suffer.

5.6.4 Experiment about input time steps

The purpose of this experiment is to find out how many input time steps one should feed to the MLP network. The initial experiment regarding this subject resulted in ten inputs to be used. However, at that point, the network didn't contain any dropout, which could mean that we get different results now. The section is related to RQ1.3.

The setup for the experiment is the following. MLP-64 network from Figure 13 was used, and it now uses dropout regularization. To ensure that this experiment is valid, each network predicts the same amount instances of the data and contain the same amount of training instances. The training frames were first built to have 14 input steps, and then the extra time steps were cut off from the end. This ensures that the same instances were predicted by each history length and that each network was trained with the same amount of training instances. Table 14 below shows the results of this experiment.

Table 14. Various amount of input time steps (history length)

Room A	Data Selection method: all = all data used var = only data with high variation was used.							
	var	var	var	var	var	var		
Input Time Steps	AVG VAL MSE	AVG TEST MSE	AVG VAL RMSE	AVG TEST RMSE	AVG VAL MAE	AVG TEST MAE	p-value for TEST RMSE	
7	**210,614**	**134,741**	**14,512**	**11,608**	**8,648**	**7,276**	N/A	
6	210,707	134,857	14,516	11,613	8,669	7,305	0,7	
9	211,097	135,035	14,529	11,62	8,688	7,289	0,35	
5	211,073	135,163	14,528	11,626	8,696	7,331	0,11	
10	211,423	135,228	14,54	11,629	8,678	7,281	0,15	
4	211,174	135,309	14,532	11,632	8,704	7,347	0,1	
8	210,773	135,414	14,518	11,636	8,657	7,304	0,07	
12	212,563	135,693	14,579	11,649	8,693	7,288	3,88E-03	**
3	212,127	136,203	14,565	11,67	8,772	7,414	3,84E-06	***
11	212,109	136,418	14,564	11,68	8,683	7,316	3,08E-05	***

The differences between various input lengths were rather small. This is evident from the p-values that were not statistically significant in most cases. Seven input steps produced the best results across all metrics: MSE, RMSE, and MAE. This is why seven input steps were chosen to be used in the next experiment.

5.6.5 Feature sets

The purpose of this experiment was to find out which features should be fed to the network. The experiment was conducted using various sets of features, which are listed in Table 15. All Features were scaled using sklearn's RobustScaler. A seven-minute history was used since it worked well in an earlier experiment. The section is related to RQ1.3.

The used features are the following. Carbon Dioxide (co2), movement sensor data (PIR), door state, illumination (lux), noise levels (noise), volatile organic compound (VOC). UNIX time is the UNIX time measured in seconds. UNIX time not used directly but instead three features were crafted from it: minute, hour, and isweekday. Minute, hour, and isweekday features were converted to local time before use. Isweekday has value 1 during the week, and 0 during the weekend.

The term "raw" that is found in the table, refers to the scaled value that was read from the sensor. Minimum (min), maximum (max), mean (avg), and median values were calculated from the scaled raw values based on the history length. For instance, history length seven (7 minutes) would calculate features from a seven-minute history. The term "diff" refers to the difference set that is obtained by subtracting the earlier value from the current one. To make sure that both diff and raw features have the same length first, and additional raw value was used. For instance, for 7-minute history, an additional minute was included. Then diff was calculated, and finally, the oldest raw value was dropped to ensure that both raw and diff had the same number of units. Features related to time (minute, hour, isweekday) only use the most recent timestamp for their calculation, since it wasn't considered important to include all the times, from the 7-minute horizon.

Some example features will be explained next. Feature set Number 1: "CO2: raw, diff" uses the raw and diff features based on measured CO2. An earlier experiment had found that 7-minute history (7-time steps) should be used, which means that a total of 14 input values would be fed to the network since Feature set Number 1 contains two features.

Feature set Number 7 contains features from various measurements: CO2, UNIX time, LUX, Noise, PIR, and Door state. Each of these can have several features built from them. UNIX time had only one feature crafted based on it, while the measured variables had several features each.

Table 15. Various feature sets that were used in this thesis.

Feature Set Name	List of used features
Unicorn	CO2: raw, diff
Kugelblitz	CO2: raw, diff PIR: raw Door state: raw
Redstone	CO2: raw, diff PIR: raw, min, max, avg Door state: raw, min, max, avg
Phoenix	CO2: raw, diff PIR: raw, min, max, median Door state: raw, min, max, avg
Badger	CO2: raw, diff PIR: min, max, avg Door state: min, max, avg
Salamander	CO2: raw, diff LUX: min, max, avg Noise: max, avg PIR: raw, min, max, avg Door state: min, max, avg
Maverick	CO2: raw, diff UNIX time: isweekday LUX: min, max, avg Noise: max, avg PIR: raw, min, max, avg Door state: min, max, avg
Schrödinger's Cat	CO2: raw, diff UNIX time: hour, isweekday LUX: min, max, avg Noise: max, avg PIR: raw, min, max, avg Door state: min, max, avg
Snowball	CO2: raw, diff UNIX time: hour, isweekday LUX: min, max, avg Noise: max, avg PIR: raw, min, max, avg Door state: min, max, avg VOC: raw, min, max, avg
Blue Whale	CO2: raw, diff UNIX time: minute, hour, isweekday LUX: min, max, avg Noise: max, avg PIR: raw, min, max, avg Door state: min, max, avg
CO2_RAW	CO2: raw

The network structures that were used are described in section 4.5. Table 16 below shows the results of this experiment.

Table 16. Network performance using various features.

			Data Selection method: all = all data used var = only data with high variation was used.			
			var	var	var	Test p-value
Input Time Steps	Model	Feature Set Name	AVG TEST MSE	AVG TEST RMSE	AVG TEST MAE	for RMSE
7	MLP-64	Kugelblitz	**114.253**	**10.688**	**6.875**	N/A
7	MLP-64	Redstone	115.494	10.746	6.945	0.033 *
7	MLP-64	Phoenix	116.319	10.784	6.985	0.001 ***
7	MLP1-WIDE	Kugelblitz	118.728	10.896	6.921	2.30E-16 ***
7	MLP-64	Maverick	123.69	11.119	7.519	1.09E-14 ***
7	MLP-64	Snowball	124.664	11.164	7.632	8.47E-25 ***
7	MLP-64	Badger	124.807	11.171	7.245	1.88E-23 ***
7	MLP-64	Salamander	125.147	11.185	7.483	7.70E-20 ***
7	MLP-64	Blue Whale	125.716	11.21	7.61	1.34E-16 ***
7	MLP-64	Schrödinger's Cat	127.155	11.274	7.671	9.53E-23 ***
7	MLP-64	Unicorn	131.749	11.478	7.194	3.25E-47 ***
7	MLP1	Kugelblitz	136.111	11.665	7.417	4.90E-34 ***
2	Line fit	CO2_RAW	143.629	11.985	7.648	-
3	Line fit	CO2_RAW	145.927	12.08	7.464	-
1	PPV	CO2_RAW	188.741	13.738	8.905	-

There is a noticeable improvement in the RMSE score when using the Kugelblitz feature set. Using just the "raw" and "diff" features (Unicorn) as in earlier experiments resulted in RMSE of 11.478 ppm. Meanwhile, the Kugelblitz feature set had the best performance with an RMSE of 10.688 ppm, which means that feature selection improved the performance by 0.790 ppm on their own. This difference is much larger than most improvements in the earlier experiments. For instance, the difference between MLP and GRU networks was only 0.011 ppm.

The Redstone feature set also performed well. There is no statistically significant difference between Kugelblitz and Redstone feature sets. Their performance was essentially the same. Because the Kugelblitz feature set is simpler and produces similar results it is best to use the Kugelblitz feature set.

The Kugelblitz feature set was also used with MLP1 and MLP1-WIDE. Here MLP1-WIDE has one hidden layer that has the same total parameter count in the network as MLP-64 has. Out of these three alternatives MLP-64 had the best performance.

5.7 Findings

This section discusses what were the most important findings. First a brief overview of the best hyperparameters, and then the results compared to existing literature.

RQ1 was related to the structure of the network. MLP was the best network out of the four tested network types: MLP, LSTM, GRU and CNN (RQ1.1). ELU activation was found to be best as a main activation, while tanh performed best on the last layer. The final network structure for the deep MLP network is visible in Figure 20. The best network configuration for CO2 prediction was the following. Deep MLP network using ELU as the main activation and tanh on the last layer. Normalization layers were not used in the final model. The network takes seven input time steps (7 minutes) and predicts ten minutes to the future. Shortcut connections (RQ1.2) did not improve the accuracy in this study, so it's better to not use shortcuts for MLP with five hidden layers. The best feature set (RQ1.3) was the Kugelblitz feature set described in section 5.6.5.

RQ2 was related to training data selection. It was found that the network should have at least as many training instances as there are parameters in the network. If this is not the case, then a line fit algorithm using two previous CO2 values would be superior. It was found that it's best to use all training data and not just sections that have high variation (RQ2.1). It was also found that it's best to have this training data from the same room one is trying to predict (RQ2.2). However, it should be noted that the amount of training data also factors into this one, if there isn't enough data it might be beneficial to use data from all rooms. In this study, it was found that if the parameter count is higher than the number of training instances, then it's advisable to either reduce the parameter count of the model, get more training data from the room, or train the model using data from all rooms.

In the final experiment regarding various feature sets it was found that the newly developed deep MLP model improved the RMSE scores compared to the proxy model "MLP1" that was built based on earlier research. Improvement in RMSE scores against MLP1 was 0.977 PPM, and against MLP1-WIDE 0.208 PPM (see Table 16). These scores were calculated for the variating sections of the data since the differences are easier to see there, and variating sections were the main interest in this thesis. It should be noted that the proxy model used the same features and shared most of the hyperparameters such as activation functions. This means that the performance improvement over previous studies is purely based on the network itself.

Next, let's discuss the results from previous studies. In the comparison, I'm going to use the test score from the "all" data. This means that also sections with low variation in CO2, are included in the test data.

Khazaei et al. (2019) use a one-minute sampling interval and predicted five minutes to the future. The resulting MSE score was 17 ppm. In my study prediction was made 10 minutes to the future, and MSE score was 36.8 ppm. The large difference could be caused by the fact that my study predicted 10 minutes to the future, while Khazaei et al. (2019) predicted five.

Macarulla et al. (2017) predicted CO2 levels 15 minutes to the future, with a 15-minute sampling interval. The best model reached RMSE of 41.10 ppm (Macarulla et al, 2017). My study had an RMSE of 6.07 ppm when predicting 10 minutes to the future which is better than the earlier study.

Falk use LSTM to predict 10 minutes to the future. One-minute sampling interval was used. (Falk, 2018.) The RMSE score was 9.09 ppm. My study had an RMSE of 6.07 ppm, which is better. The notable difference could result from using different training data, or from the fact that Falk excluded weekends from his data. The sampling intervals and prediction horizons were the same in both studies. Table 17 below contains the summarization of results of earlier studies and this study. Best model of each study is shown, and a baseline if it was used.

Table 17. Summarization of various studies

Study	Sampling interval (minutes)	Future prediction index (minutes)	Used history length (minutes)	Model	RMSE (ppm)	MSE (ppm)	MAE (ppm)
Falk (2018)	1	10	5	LSTM 1 hidden layer	9.09		
Falk (2018)	1	10	5? *	Linefit	48.77		
Khazaei et al. (2019)	1	5	1	MLP 1 hidden layer, 4 neurons		17	
Macarulla et al. (2017)	15	15		Deterministic model using mass balance equations.	41.10		
This thesis	1	10	7	MLP 5 hidden layers, 64 neurons each	6.07	36.8	2.82
This thesis	1	10	2	Linefit 2 inputs	6.845	46.85	3.35

* Five-minute history is assumed since the history for line fit was not specified in the study.

There is a noticeable difference between the line fit models in this thesis and the line fit in Falk's (2018) study. I would assume that Falk used longer history length to fit the line which would result in higher errors for the line fit model. In this thesis, a surprising finding is that the LSTM model performed worse than the MLP model did. For instance, the study by Falk (2018) concluded that LSTM had better performance than the MLP model.

Now let's explore which hyperparameters gave the biggest improvements to the RMSE score. Table 18 contains information regarding various experiments and how much each experiment improved the performance compared to the second-best alternative. This table can be used in future studies to determine which hyperparameters should be tuned further.

Table 18. Various hyperparameters, and improvements

Changed variable	**Chosen value**	**Difference compared to the next best hyperparameter.** **Measured in RMSE**	**Comment**
Data selection: all vs var	all	0.11 - 0.4 ppm	
Data selection: One room vs many	many	0.223 ppm	May depend on the used data. With enough data, one room may produce better results.
Main activation	ELU	0.021 ppm	
Last activation	tanh	0.016 ppm	
Network type	MLP	0,010 ppm	
Input time steps	7	0.005 ppm	
Feature set	Kugelblitz	0.058 ppm	Big difference compared to the feature set used earlier stages (0.790 ppm).

Based on the table it would seem, that data selection methods had the biggest difference compared to the other alternative. Feature selection improvement over the seconds best alternative is small. However, compared to the original features set (Unicorn) the new feature set (Kugelblitz) improved the performance by 0.790 ppm. It is possible that this feature set is not ideal, which is why both feature engineering and training data selection should be explored further.

To sum up the findings, it can be said that deeper produce lower RMSE errors. This is evident from the fact that the models proposed in this study, always outperformed the MLP1 model, when using the same features for each model. Improvement of the multilayer model was not evident before the study, because earlier CO2 prediction studies didn't explore networks that were deeper than 1-2 hidden layers.

6. Conclusions and future work

This chapter describes the most important findings, limitations, and discusses future work. This study was about finding an effective way to use a neural network to predict the future value of CO2 in indoor air.

Because results across different studies are not comparable because of different datasets, sampling intervals, and measurement units, it was decided that a benchmark model should be built instead. This way all the models would predict the same data and use the same sampling interval. The benchmark model was an MLP network with one hidden layer.

This thesis found that deep models have a slightly better RMSE score compared to shallow models that had been used in earlier research. Four neural networks were proposed: LSTM, GRU, CNN, and MLP. Out of these MLP had the best performance, and GRU also had similar performance. Due to its simplicity, the MLP model was chosen to be tuned further. The deep MLP architecture improved on the prediction RMSE compared to the benchmark model "MLP1" by 0.977 ppm (8%), which is noticeable. The improvement over "MLP1-WIDE", was smaller at 0.208 ppm (or 2%). All proposed models outperformed the benchmark model. Out of the proposed models CNN had the worst performance.

There are five limitations in this study, which are the following. Activation functions were only tested briefly, more iterations are needed, dropout layers could use further tuning. The fourth limitation was that the data selection method was chosen to be all data, which was a result of using MAE score instead of RMSE score. Finally, random seeds were not locked in this study.

The first limitation is related to activation functions. Because there were so many activation functions they were only tested briefly. Because differences between various activation functions very small in many cases it is conceivable that some improvements can be made by experimenting more with the activation functions. Furthermore, some research shows that it may be beneficial to omit activation function from certain parts of the network (Zhao, Zhang, Guan, Tang, & Wang, 2017; Gross & Wilber, 2016), which means further study is warranted regarding activation functions.

The second limitation is related to the number of iterations that were used. At an earlier phase of the study, it was found that ten-minute history length was ideal. However, at a later stage, it was found than seven-minute history was instead ideal. Essentially what this means is, that changing one parameter influences another. Due to time limitations, the iterations had to be stopped earlier than desired. Ideally, each experiment should be repeated iteratively until the results no longer change.

The third issue relates to the dropout layers. In certain experiments, several dropout layers were used, and each had the same amount of dropout. However, there is no reason why each dropout layer should have the same amount of dropout, and it is conceivable that performance might improve by experimenting with various dropout values across different layers. New developments regarding dropout can be considered for further study. For instance, Gaussian dropout outperformed vanilla dropout on several datasets such as MNIST, CIFAR-10, and IMAGENET ILSVRC-2012 (Shen, Tian, Liu, Xu, & Tao, 2017).

The fourth limitation is related to the data selection method. The experiment aimed to find out which type of data selection works the best. Should one use all available training data, or only use that training data that contains sections with high CO2 variation. At the time when this test was done the measurement criteria to use RMSE had not yet been done, which means that MAE score was used at this stage. Future studies may want to repeat this stage of the experiment by using the RMSE score.

The fifth limitation in this study was the fact that random seeds were not locked. However, this doesn't threaten the validity of the study. This is because p-values across several training sessions were used to determine if the difference between two hyperparameters, which means we can be relatively certain that the results are correct. However, the repeatability of the study is more difficult without locking the random seeds. In an ideal case, the random seeds should have been picked at the start. Picking a single random seed wouldn't be enough, because a specific random seed might be ideal for LSTM while being less ideal for MLP. This means that one should pick N random seeds and use the same ones throughout the study. For example, pick 40 random seeds, and train the network once for each random seed. Then repeat the training with the next predictor using the same random seeds. Future studies should use this type (or equivalent) way of initializing random seeds to ensure the repeatability of the studies.

Future research is discussed next. Five main topics for future research are explored. The future research topics are the following: CNN performance improvement, the use of hybrid models, LSTM/GRU can be trained with better data, transfer learning can be considered, a benchmark dataset could be created for CO2 prediction. Each of these topics are explored next.

CNN performance in this study was lacking, even though it has been previously used with good results in other prediction tasks, such as speech synthesis. Compared to the WaveNet paper by Oord et al. (2016) the sampling frequency in this study is lower. Due to this, further research regarding CO2 prediction with CNNs, should consider using data with higher sampling frequency, which may help in improving the prediction results of the CNN network. Alternatively, advanced network configurations can be considered such as WaveNet by Oord et al. (2016) or Transformer by Vaswani et al. (2017).

One idea for future studies would be to combine a neural network as a part of a larger hybrid model. It has been shown that a hybrid model has better accuracy than a single model (Divina et al., 2018; Zhang, 2003; Evitan, 2019). Hybrid models were left out of this study to limit the scope of this thesis.

Another idea for future studies would be to obtain higher quality data. In this study LSTM and GRU networks were trained in stateless mode because there were so many gaps in the training data. Because of this, further studies could further experiment on LSTM and GRU, by using better training data, to see if training in stateful mode would yield better results.

The largest improvements in the performance were achieved by the data selection method, and the selected features. Future studies could focus on data selection, and on feature engineering since they have the most potential for improvement.

This thesis found that deep models perform slightly better than shallow ones. However, it's hard to say if the improvement is meaningful for the occupants in the room. Future studies may be needed where CO2 predictors that are run online and plugged to a ventilation control system. This type of study would analyse differences (if any) between

various prediction schemes by studying how well the CO2 level stays within an acceptable range with various predictors. The measured indicators could be: 1. the maximum CO2 values reached (smaller is better), and 2. how long it takes to reach that maximum value (longer is better), and possibly other metrics. Ideally, the measured rooms should have their occupancies frozen for the duration of the study, meaning that the room would always have the same person (people) in it. This could be done via a controlled experiment, where a person is asked to repeat some task, while the CO2 values are being predicted and controlled, then it's repeated using a different predictor and the same person. In addition, the door could remain closed in 50% of the experiments, and closed in the other 50% of the experiments, to ensure the balanced data regarding the door state for each predictor. This would result in two experiments for each predictor.

Transfer learning is another idea for further studies. It was found that including data from other rooms did not help the RMSE score. Further studies could further extend on this experiment, by first training the network with all possible data from all rooms, then fine-tuning the last layer (or last few layers) of the network with data that is from the same room that is being predicted. This is known as transfer learning, which has been used in image recognition with good results. For instance, Shin et al. (2016) got improved accuracy on certain datasets when fine-tuning AlexNet for medical images. Essentially this type of experiment regarding training data could compare three results: 1. training data from all rooms, predict one room; 2. train data from one room, predict the same room; and 3. train a generic predictor with all data from all rooms, then fine-tune the last layer with data only from that same room.

Further studies could explore how to make papers related to CO2 prediction more comparable to each other. In this study, it was found that results are often not directly comparable since there isn't a common dataset that everyone would be using. One alternative would be to create a high-quality dataset for indoor air. Another way to make studies comparable would be for each of the authors, to include the exact network configurations and all the used hyperparameters. This would ensure that the study can be compared, even if the dataset itself is not available.

To conclude it can be said that deep models have slightly better performance than shallow models. Future studies regarding controlled ventilation situations may be required to determine if this improvement is meaningful for the occupants.

Acknowledgements: The research was partly implemented in SCOTT (Secure COnnected Trustable Things) project, which has received funding from Business Finland and the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program, and from Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium and Norway. I thank all contributors.

Conflicts of Interest: The author declares no conflict of interest.

References

Aggarwal, K., Kirchmeyer, M., Yadav, P., Keerthi, S. S., & Gallinari, P. (2019). Conditional Generative Adversarial Networks for Regression. arXiv preprint arXiv:1905.12868.

At What Height Should Sensors be Mounted? (n.d.). Retrieved September 9, 2019, from: https://www.critical-environment.com/support/faq/at-what-height-should-sensors-be-mounted

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691.

Bosch Sensortec GmbH. (2020). Barometric pressure sensors. Retrieved on March 30, 2020, from: https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/

Calì, D., Matthes, P., Huchtemann, K., Streblow, R., & Müller, D. (2015). CO2 based occupancy detection algorithm: Experimental analysis and validation for office and residential buildings. *Building and Environment, 86*, 39-49. doi:10.1016/j.buildenv.2014.12.011

Candanedo, L. M., & Feldheim, V. (2016). Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buildings, 112*, 28-39. doi:10.1016/j.enbuild.2015.11.071

Chao, H. J., Schwartz, J., Milton, D. K., & Burge, H. A. (2003). The work environment and workers' health in four large office buildings. *Environmental Health Perspectives, 111*(9), 1242-1248. doi:10.1289/ehp.5697

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

CO2 Sensor Location: Where to Mount Your CO2 IAQ Monitor. (2012). Retrieved September 9, 2019, from: https://www.co2meter.com/blogs/news/6056206-co2-sensor-location-where-to-mount-your-co2-iaq-monitor

Cohen, I. (n.d.). A Quick Guide to the Different Types of Outliers. Retrieved October 17, 2019, from: https://www.anodot.com/blog/quick-guide-different-types-outliers/

Dai, J., Song, H., Sheng, G., & Jiang, X. (2017). Cleaning method for status monitoring data of power equipment based on stacked denoising autoencoders. *Ieee Access, 5*, 22863-22870. doi:10.1109/ACCESS.2017.2740968

Divina, F., Gilson, A., Goméz-Vela, F., García Torres, M., & Torres, J. (2018). Stacking ensemble learning for short-term electricity consumption forecasting. *Energies, 11*(4), 949. doi:10.3390/en11040949

Dong, B., Andrews, B., Lam, K. P., Höynck, M., Zhang, R., Chiou, Y. S., & Benitez, D. (2010). An information technology enabled sustainability test-bed (ITEST) for occupancy detection through an environmental sensing network. *Energy and Buildings, 42*(7), 1038-1046. doi:10.1016/j.enbuild.2010.01.016

Evitan, G. (2019). In Depth EDA and Stacking with House Prices. Retrieved September 4, 2019, from https://www.kaggle.com/gunesevitan/in-depth-eda-and-stacking-with-house-prices

Falk, S. (2018). Predicting building-automation time-series data with supervised methods. (Master's thesis, University of Helsinki). Retrieved from: https://helda.helsinki.fi/handle/10138/234237

Gated recurrent unit. (n.d.). Retrieved February 28, 2020, from: https://en.wikipedia.org/wiki/Gated_recurrent_unit

Gross, S., & Wilber, M. (2016). Training and investigating residual nets. Retrieved March 3, 2020, from: http://torch.ch/blog/2016/02/04/resnets.html

Haverinen-Shaughnessy, U., & Shaughnessy, R. J. (2015). Effects of classroom ventilation rate and temperature on students' test scores. *PloS one, 10*(8). doi: 10.1371/journal.pone.0136165

Haverinen-Shaughnessy, U., Moschandreas, D. J., & Shaughnessy, R. J. (2011). Association between substandard classroom ventilation rates and students' academic achievement. *Indoor air, 21*(2), 121-131. doi:10.1111/j.1600-0668.2010.00686.x

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75-105.

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. *In Design research in information systems* (pp. 9-22). Springer, Boston, MA.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735

K30. (n.d.). Air and gas sensing technology. Retrieved on March 30, 2020, from: https://senseair.com/products/flexibility-counts/k30/#prod2

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. *In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 1725-1732). doi: 10.1109/CVPR.2014.223

Kasche, J., & Nordström, F. (2020). Regularization Methods in Neural Networks. (Bachelor's thesis, Uppsala University). Retrieved from: http://uu.diva-portal.org/smash/record.jsf?pid=diva2%3A1389238&dswid=-2006

Kernel density estimation. (n.d.). Retrieved April 16, 2020, from: https://en.wikipedia.org/wiki/Kernel_density_estimation

Khazaei, B., Shiehbeigi, A., & Kani, A. H. M. A. (2019). Modeling indoor air carbon dioxide concentration using artificial neural network. *International journal of environmental science and technology, 16*(2), 729-736. doi:10.1007/s13762-018-1642-x

Kumar, A., & Goyal, P. (2013). Forecasting of air quality index in Delhi using neural network based on principal component analysis. *Pure and Applied Geophysics, 170*(4), 711-722. doi:10.1007/s00024-012-0583-4

Kuremoto, T., Hirata, T., Obayashi, M., Mabu, S., & Kobayashi, K. (2019). Training Deep Neural Networks with Reinforcement Learning for Time Series Forecasting. *In Time Series Analysis*. IntechOpen. doi:10.5772/intechopen.85457

Li, J., Wall, J., & Platt, G. (2010). Indoor air quality control of HVAC system. *In Proceedings of the 2010 International Conference on Modelling, Identification and Control* (pp. 756-761).

Long short-term memory. (n.d.). Retrieved February 28, 2020, from: https://en.wikipedia.org/wiki/Long_short-term_memory

LP8 pin headers. (n.d.). Retrieved on March 30, 2020, from: https://senseair.com/products/power-counts/lp8

Macarulla, M., Casals, M., Carnevali, M., Forcada, N., & Gangolells, M. (2017). Modelling indoor air carbon dioxide concentration using grey-box models. *Building and Environment, 117*, 146-153. doi:10.1016/j.buildenv.2017.02.022

Maddalena, R., Mendell, M. J., Eliseeva, K., Chan, W. R., Sullivan, D. P., Russell, M., ... & Fisk, W. J. (2015). Effects of ventilation rate per person and per floor area on perceived air quality, sick building syndrome symptoms, and decision-making. *Indoor Air, 25*(4), 362-370. doi:10.1111/ina.12149

mcf88. (n.d.). MCF-LW12CO2. Retrieved on March 30, 2020, from: https://www.mcf88.it/wp-content/uploads/2018/11/mcf88-MCF-LW12CO2.pdf

Merema, B., Delwati, M., Sourbron, M., & Breesch, H. (2018). Demand controlled ventilation (DCV) in school and office buildings: Lessons learnt from case studies. *Energy and Buildings, 172*, 349-360. doi:10.1016/j.enbuild.2018.04.065

Misra, D. (2019). Mish: A Self Regularized Non-Monotonic Neural Activation Function. arXiv preprint arXiv:1908.08681

Murphy, J. (2016). Implementing Demand-Controlled Ventilation Strategies. Retrieved from: https://ashraemadison.org/downloads/CRC_2016_Presentations/murphy__dcv_and_ashrae_62.1__ashrae_crc__29_april_2016_.pdf

Nan, G. (2019). Who invented convolution neural networks? Retrieved February 27, 2020, from: https://www.quora.com/Who-invented-convolution-neural-networks

Norbäck, D., Torgen, M., & Edling, C. (1990). Volatile organic compounds, respirable dust, and personal factors related to prevalence and incidence of sick building syndrome in primary schools. *Occupational and Environmental Medicine, 47*(11), 733-741. doi:10.1136/oem.47.11.733

Norhidayah, A., Chia-Kuang, L., Azhar, M. K., & Nurulwahida, S. (2013). Indoor air quality and sick building syndrome in three selected buildings. *Procedia Engineering, 53*, 93-98. doi:10.1016/j.proeng.2013.02.014

Olah, C. (2015). Understanding lstm networks. Retrieved February 28, 2020, from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., & Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

Panasonic. (n.d.). EKMB1301113K. Retrieved on March 30, 2020, from: https://na.industrial.panasonic.com/products/sensors/sensors-automotive-industrial-applications/lineup/pir-motion-sensor-papirs/series/70516/model/73542

Pantazaras, A., Lee, S. E., Santamouris, M., & Yang, J. (2016). Predicting the CO2 levels in buildings using deterministic and identified models. *Energy and Buildings, 127*, 774-785. doi: 10.1016/j.enbuild.2016.06.029

PeakTech. (n.d.). PeakTech 8005. Retrieved on March 30, 2020, from: https://www.peaktech.de/productdetail/kategorie/schallpegelmessgeraete/produkt/p-8005.html

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems, 24*(3), 45-77. doi:10.2753/MIS0742-1222240302

Perceptron. (n.d.). Retrieved from Wikipedia on March 11, 2020, from: https://en.wikipedia.org/wiki/Perceptron

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. Neural Networks, 11(4), 761-767.

Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

Regression analysis. (n.d.). Retrieved from Wikipedia on April 1, 2020, from: https://en.wikipedia.org/wiki/Regression_analysis

Rödjegård, H. 2017. Carbon dioxide as an outdoor air quality monitor? Retrieved December 30, 2019, from: https://senseair.com/updates/news/carbon-dioxide-as-an-outdoor-air-quality-monitor/

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review, 65*(6), 386.

Satish, U., Mendell, M. J., Shekhar, K., Hotchi, T., Sullivan, D., Streufert, S., & Fisk, W. J. (2012). Is CO2 an indoor pollutant? Direct effects of low-to-moderate CO2 concentrations on human decision-making performance. *Environmental health perspectives, 120*(12), 1671-1677. doi:10.1289/ehp.1104789

Shen, X., Tian, X., Liu, T., Xu, F., & Tao, D. (2017). Continuous dropout. *IEEE transactions on neural networks and learning systems, 29*(9), 3926-3937.

Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging, 35*(5), 1285-1298.

Silicon Labs. (n.d.). I2C HUMIDITY AND TEMPERATURE SENSOR. Retrieved on March 30, 2020, from: https://www.silabs.com/documents/public/data-sheets/Si7021-A20.pdf

Skön, J., Johansson, M., Raatikainen, M., Leiviskä, K., & Kolehmainen, M. (2012). Modelling Indoor Air Carbon Dioxide (CO2) Concentration using Neural Network.

SP-Elektroniikka. (n.d.). Magneettikytkin ovikytkin karmipuoli ja ovipuoli MAKY2 NO/NC. Retrieved on March 30, 2020, from: https://www.spelektroniikka.fi/p6872-magneettikytkin-ovikytkin-karmipuoli-ja-ovipuoli-maky2-no-nc-en.html

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

Srivastava, N., Mansimov, E., & Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. *In International conference on machine learning* (pp. 843-852).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. *In Thirty-first AAAI conference on artificial intelligence*.

tf.keras.layers.GRU. (2020). Retrieved February 2, 2020, from: https://www.tensorflow.org/api_docs/python/tf/keras/layers/GRU

tf.keras.layers.LSTM. (2020). Retrieved on March 30, 2020, from: https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

Tsang, S-H. (2018). Review: DilatedNet — Dilated Convolution (Semantic Segmentation). Retrieved September 12, 2019, from: https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *In Advances in neural information processing systems* (pp. 5998-6008).

Wei, W., Ramalho, O., Malingre, L., Sivanantham, S., Little, J. C., & Mandin, C. (2019). Machine learning and statistical models for predicting indoor air quality. *Indoor Air, 29*(5), 704-726. doi:10.1111/ina.12580

World Health Organization Regional Office for Europe. (n.d.). Sick building syndrome. Retrieved October 9, 2019, from: https://www.wondermakers.com/Portals/0/docs/Sick%20building%20syndrome%20by%20WHO.pdf

Wu, K., Zhu, Y., Li, Q., & Wu, Z. (2017). A distributed real-time data prediction framework for large-scale time-series data using stream processing. *International Journal of Intelligent Computing and Cybernetics, 10*(2), 145-165. doi:10.1108/IJICC-09-2016-0033

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing, 50*, 159-175. doi:10.1016/S0925-2312(01)00702-0

Zhang, X., Wargocki, P., & Lian, Z. (2017). Physiological responses during exposure to carbon dioxide and bioeffluents at levels typically occurring indoors. *Indoor Air, 27*(1), 65-77. doi:10.1111/ina.12286

Zhao, G., Zhang, Z., Guan, H., Tang, P., & Wang, J. (2017). Rethink ReLU to Training Better CNNs. *arXiv preprint arXiv:1709.06247.*

Zuraimi, M. S., Pantazaras, A., Chaturvedi, K. A., Yang, J. J., Tham, K. W., & Lee, S. E. (2017). Predicting occupancy counts using physical and statistical Co2-based modeling methodologies. *Building and Environment, 123*, 517-528. doi: 10.1016/j.buildenv.2017.07.027

Appendix A. Time features (most recent timestep)

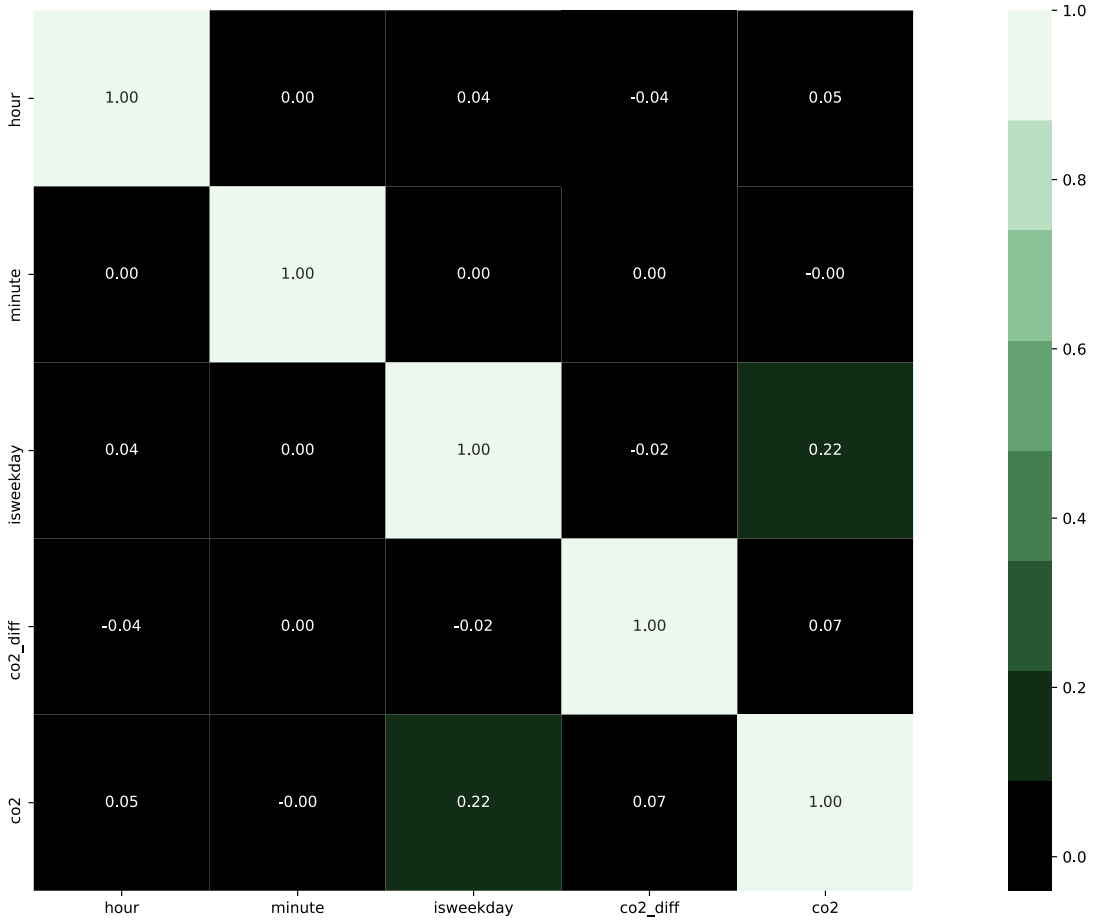

Figure 22. Correlation matrix of time features and the future CO2.

Appendix B. Average features (7 timesteps)

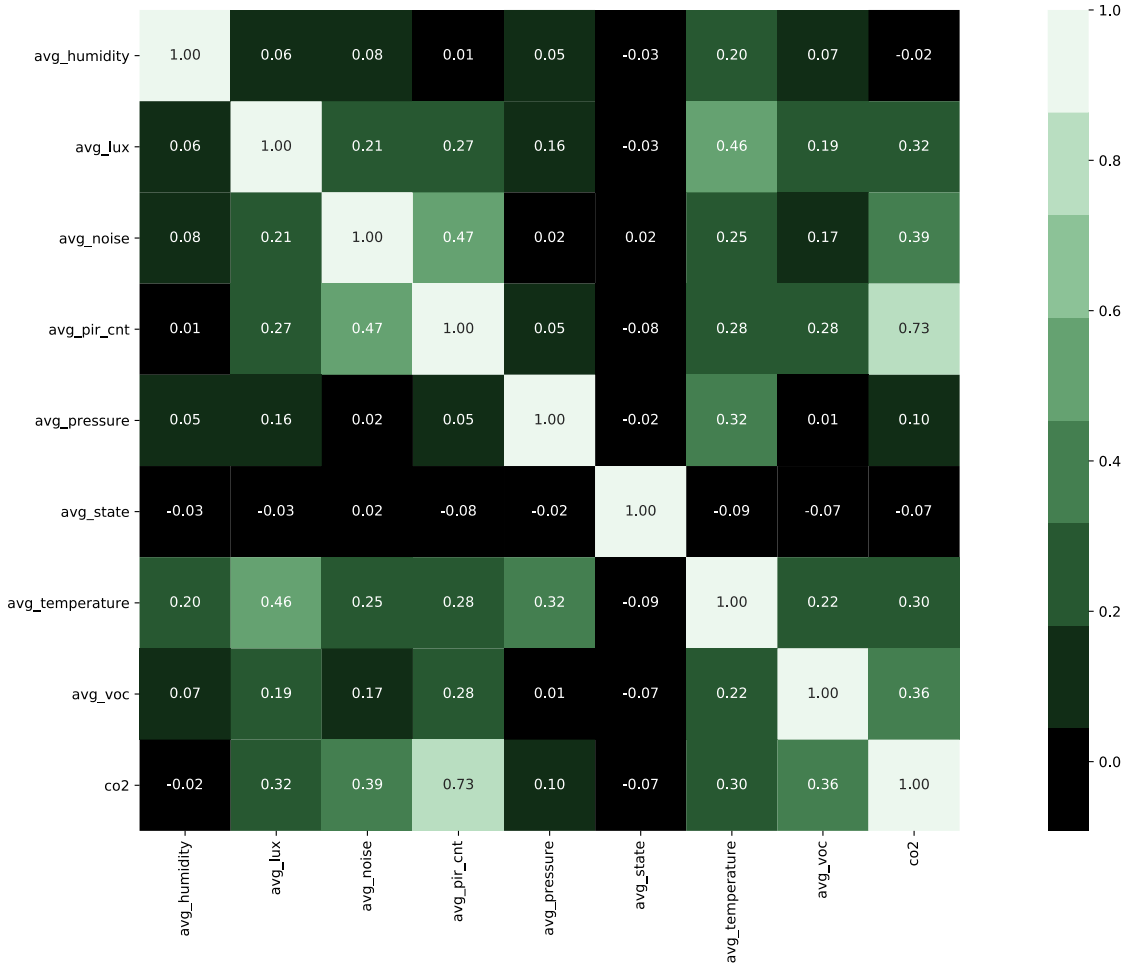

Figure 23. Correlation matrix between average values within 7 time steps and between the future CO2 value.

Appendix C. Max features (7 timesteps)

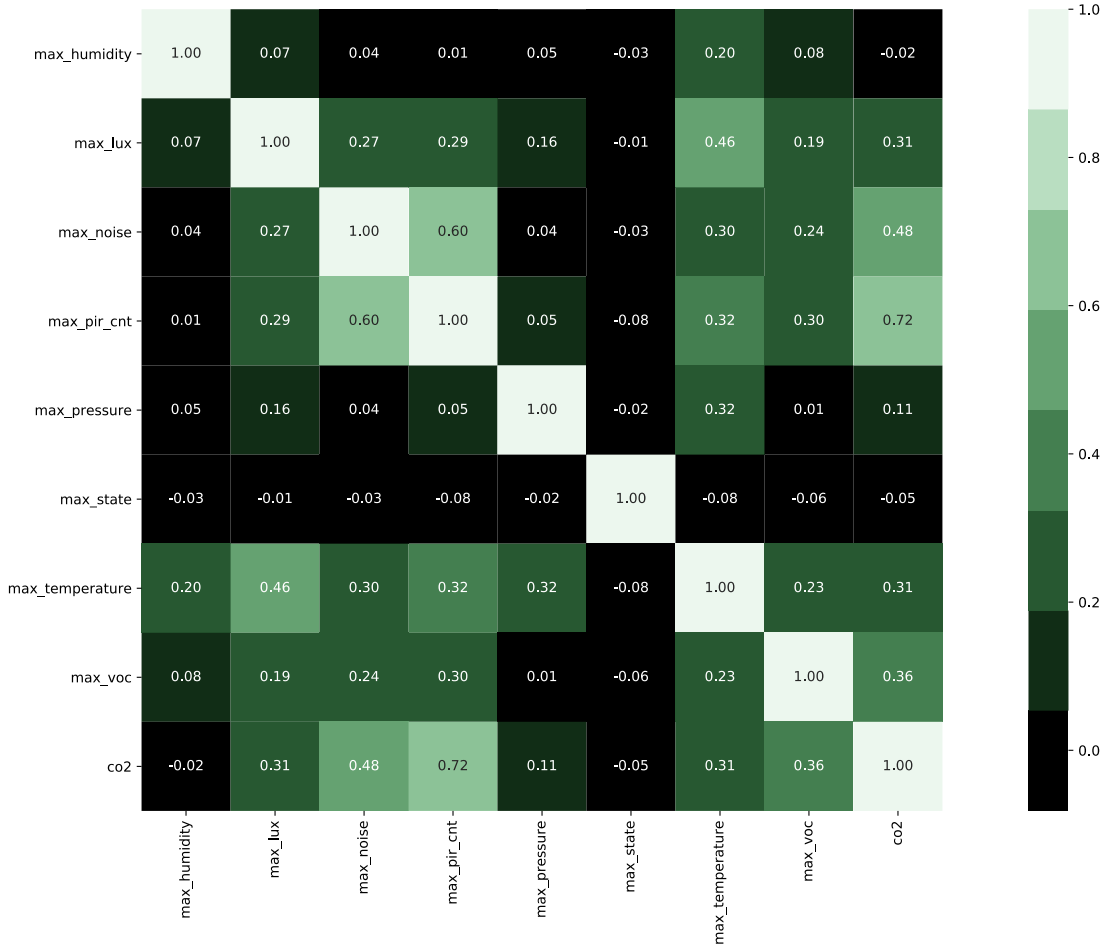

Figure 24. Correlation matrix between max values within 7 time steps and between the future CO2 value.

Appendix D. Min Features (7 timesteps)

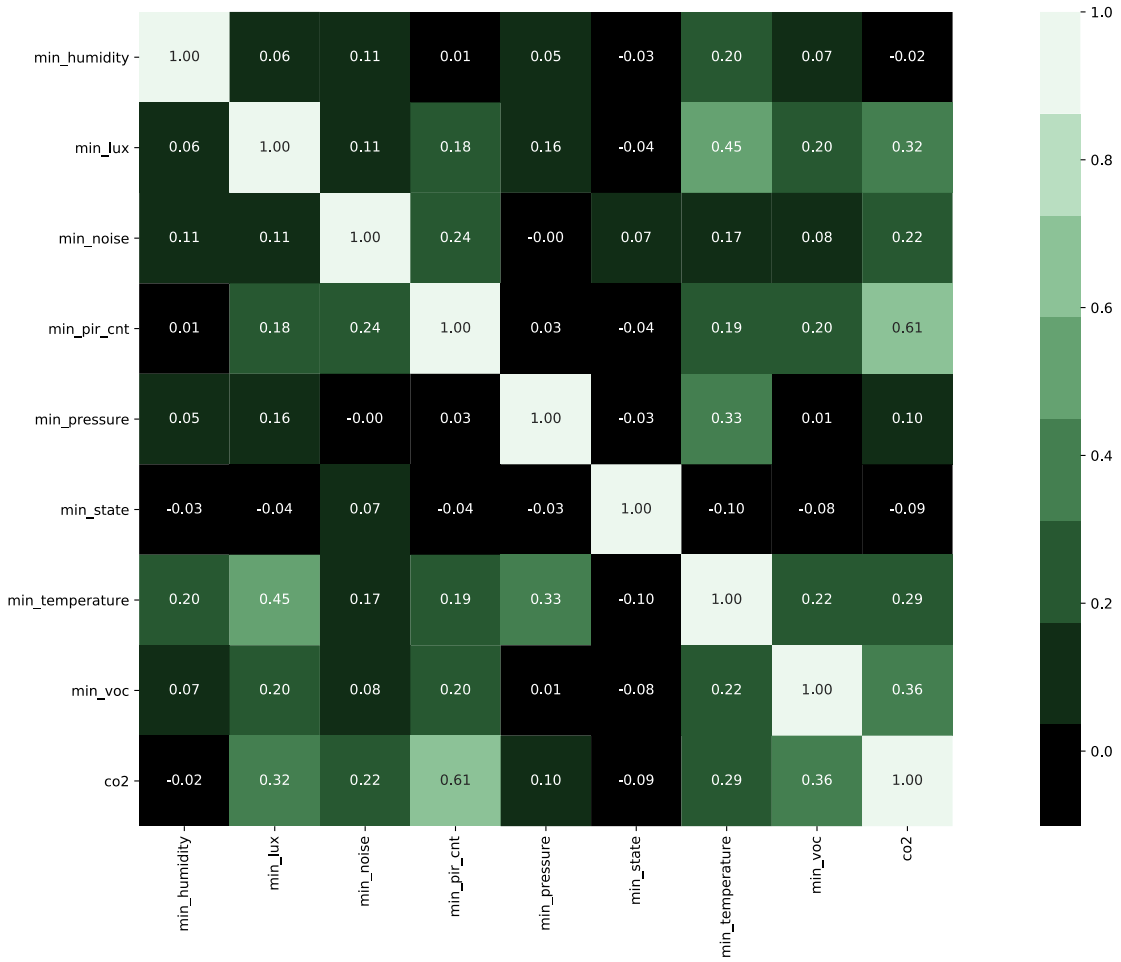

Figure 25. Correlation matrix between min values within 7 time steps and between the future CO2 value.

Appendix E. Median features (7 timesteps)

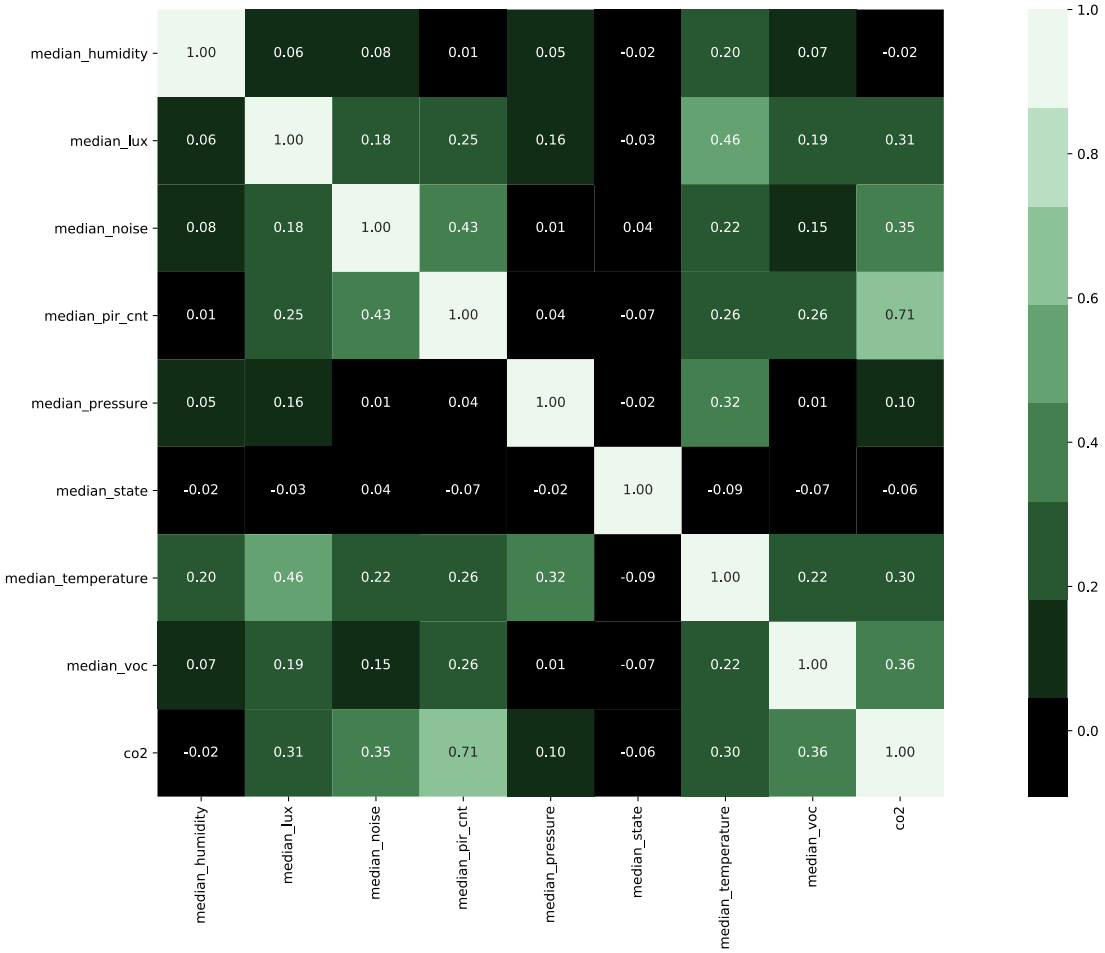

Figure 26. Correlation matrix between median values within 7 time steps and between the future CO2 value.

Appendix F. Activation functions

Table 19. Activations that were tested.

Activation	**Package**
ReLU	tf.keras.layers
tanh	tf.keras.layers
elu	tf.keras.layers
selu	tf.keras.layers
softmax	tf.keras.layers
softplus	tf.keras.layers
softsign	tf.keras.layers
sigmoid	tf.keras.layers
hard_sigmoid	tf.keras.layers
exponential	tf.keras.layers
linear	tf.keras.layers
LeakyReLU	tf.keras.layers
PReLU	tf.keras.layers
ThresholdedReLU	tf.keras.layers
swish	tf.nn
ISRLU	keras_contrib
rrelu	tfa.activations
softshrink	tfa.activations
tanhshrink	tfa.activations
gelu	tfa.activations
hardshrink	tfa.activations
lisht	tfa.activations
mish	tfa.activations

Appendix G. Hyperparameters for MLP / CNN (1/2)

Table 20. Hyperparameters for MLP & CNN (1 of 2)

Model	Parameter	Tested values	Used value	Description
MLP / CNN	Main Activation	23 different activations.	ELU	See chapter 5 regarding activation functions. Appendix F has a list of all activations that were tested.
MLP / CNN	Last Activation	23 different activations.	tanh	See chapter 5 regarding activation functions. Appendix F has a list of all activations that were tested.
MLP / CNN	inputs	5-39	10	How many input timesteps were used.
MLP / CNN	features	raw, scaled, raw + scaled	raw + scaled	
MLP / CNN	learning rate	0.00001 - 0.006	0.000075	What was the initial learning rate
MLP / CNN	Data selection	var, all	all	What section of data was used for training, all data VS only data that has more variation in it.
MLP / CNN	Normalization method	Layer, Batch, None	None	Normalization method that was used.
MLP / CNN	Gaussian Noise	0.006275, 0.0125, 0.025, 0.05, 0.1, 0.5, None	None	The amount of Gaussian noise to feed in the input layer.
MLP / CNN	Fluctuation	0, 5, 10, 15, 20, 25	0	How much neuron count fluctuates across successive layers. E.g. First layer has base*1,15 neurons; the following layer has base*0,85 neurons. Where base is the neuron count we want to be near (e.g. 64).
MLP / CNN	Train data from rooms	One room, All rooms	One room	Should data from all rooms be used, or just from the room we trying to predict.
MLP	neuronsperlayer	16, 32, 49, 64, 128, 256	64	
MLP / CNN	Hidden Layers	1, 5	5	

Appendix H. Hyperparameters for MLP / CNN (2/2)

Table 21. Hyperparameters for MLP & CNN (2 of 2)

Model	Parameter	Tested values	Used value	Description
MLP / CNN	Batch size	128	128	
MLP / CNN	patience		5	How many epochs to train before learning rate is reduced.
MLP / CNN	LR multiplier		0,8	How much should learning rate be multiplied by, when reducing the learning rate.
MLP / CNN	optimizer	RMSprop, Adam	RMSprop	
MLP / CNN	Early Stopping Epochs	12, 17	17	If no improvement in the network performance, continue training this many epochs.
MLP / CNN	kernel_initializer		he_normal	The way in which the weights were initialized.
MLP / CNN	Dropout	0 - 20	depends on various factors: network, features, timesteps, amount of training data, etc.	How much dropout to use.
CNN	Kenel size	1,2,3,4	2	Size of convolutional kernel
CNN	Padding	same, causal	causal	Padding method for convolutional features.
CNN	CNN params	strided conv, dilated conv	strided	Dilated convolution + max-pooling VS convolution without dilation and using stride to downsample.

Appendix I. Hyperparameters for LSTM / GRU

Table 22. Hyperparameters for LSTM and GRU

Model	Parameter	Tested values	Used value	Description
LSTM / GRU	Main Activation		ELU	See chapter 5 regarding activation functions.
LSTM / GRU	Last Activation		tanh	See chapter 5 regarding activation functions.
LSTM / GRU	inputs	1,5,10	5	How many input timesteps were used.
LSTM / GRU	features		raw + scaled	
LSTM / GRU	learning rate	0,000001 - 0,006	0,000015	What was the initial learning rate
LSTM / GRU	Data selection		all	What section of data was used for training, all data VS only data that has more variation in it.
LSTM / GRU	Normalization method	Layer, Batch, None	None	Normalization method that was used.
LSTM / GRU	Gaussian Noise		None	The amount of Gaussian noise to feed in the input layer.
LSTM / GRU	Fluctuation		0	How much neuron count fluctuates across successive layers. E.g. First layer has base*1,15; the following layer has base*0,85 neurons. Where base is the neuron count we want to be near (e.g. 64).
LSTM / GRU	Train data from rooms		One Room	Should data from all rooms be used, or just from the room we trying to predict.
LSTM / GRU	neuronsperlayer		~34	
LSTM / GRU	Hidden Layers		5	
LSTM / GRU	Batch size		128	
LSTM / GRU	patience		5	How many epochs to train before learning rate is reduced.
LSTM / GRU	LR multiplier		0,8	How much should learning rate be multiplied by, when reducing the learning rate.
LSTM / GRU	optimizer		RMSprop	
LSTM / GRU	Early Stopping Epochs	12, 17	12	If no improvement in the network performance, continue training this many epochs.
LSTM / GRU	kernel_initializer		he_normal	The way in which the weights were initialized
LSTM / GRU	Dropout	0 - 12	depends on various factors: network, features, timesteps, amount of training data, etc.	How much dropout to use.

Appendix J. Testing various network types

Table 23. Testing various network types.

Room A			Dataset size		Data Selection method: all = all data used; var = only data with high variation was used.								
					all	var	all	var	all		var		
Input Time Steps	Model	Params	Train size	Test Size	AVG TRAIN MAE	AVG TEST MAE	AVG TRAIN MSE	AVG TEST MSE	AVG TRAIN RMSE	AVG TEST RMSE	AVG TEST RMSE		Test p-value for RMSE
10	MLP	10 879	45 498	5 714	**2,679**	7,290	**47,750**	**136,011**	**6,910**	~6,502	**11,662**	N/A	
5	GRU	11 101	45 498	5 714	2,898	7,353	52,006	136,271	7,211		11,673	0,443	
5	LSTM	10 915	45 498	5 714	2,956	7,255	52,950	137,032	7,276		11,706	2,38E-03	**
10	MLP1-WIDE	10 869	45 498	5 714	3,018	**7,198**	54,645	140,084	7,392		11,836	2,65E-33	***
10	CNN	11 411	45 498	5 714	3,095	7,230	55,578	140,181	7,455		11,840	1,24E-18	***
3	Linefit	2	46 104	5 780	3,381	7,516	57,289	147,883	7,569		12,161	-	
2	Linefit	2	46 104	5 780	3,333	7,739	59,433	146,337	7,709		12,097	-	
10	MLP1	1079	45498	5714	3,430	7,564	61,046	148,759	7,811		12,196	9,14E-24	***
1	PPV	N/A	46 104	5 780	3,354	9,080	79,927	194,461	8,940		13,945	-	

Appendix K. The effect of widening the network

Table 24. Testing how network wideness affects the performance.

Room A				Dataset size			Data Selection method: all = all data used; var = only data with high variation was used.											
							all	var	all	var	all	var	all	var	all	var	all	
Input Time Steps	Model	Params	Train size	Test Size	TEST_ALL		AVG TRAI	AVG TEST	AVG TEST		AVG TRAI	AVG TEST	AVG TEST		AVG TRAI	AVG TEST	AVG TEST	Test p-value for RMSE
10	MLP-256	268 801	45 498	5 714	18 002		**46,747**	**134,445**	**41,828**		**6,837**	**11,595**	**6,467**		2,691	7,277	2,858	N/A
10	MLP-64	18 049	45 498	5 714	18 002		49,205	134,780	41,897		7,015	11,609	6,473		2,751	7,286	2,875	0,318
10	MLP-128	68 865	45 498	5 714	18 002		47,010	135,284	42,037		6,856	11,631	6,483		**2,673**	7,277	**2,856**	0,006 **
10	MLP-32	4 929	45 498	5 714	18 002		50,079	136,185	42,533		7,077	11,670	6,522		2,809	7,330	2,959	1,08E-07 ***
10	MLP-16	1 441	45 498	5 714	18 002		51,846	137,661	43,232		7,200	11,733	6,575		2,891	7,326	3,028	1,20E-14 ***
10	MLP1	1 079	45 498	5 714	18 134		56,735	142,116	45,479		7,532	11,921	6,744		3,192	**7,273**	3,256	1,74E-40 ***
2	Linefit	2	46 104	5 780	18 134		59,433	146,337	46,518		7,709	12,097	6,820		3,333	7,739	3,327	
3	Linefit	2	46 104	5 780	18 134		57,289	147,883	48,194		7,569	12,161	6,942		3,381	7,516	3,477	
1	PPV	N/A	46 104	5 780	18 134		79,927	194,461	58,836		8,940	13,945	7,670		3,354	9,080	3,298	

Appendix L. Shortcut networks

Table 25. Performance of various shortcut networks.

Room A					Dataset size			Data Selection method: all = all data used; var = only data with high variation was used.												
								all	var	all	var	all	var	all	var	all	var	all		
Input Time Steps	Model	Params	Train size	Test Size	TEST_ALL Size	AVG TRAIN MSE	AVG TEST MSE	AVG TEST MSE	AVG TRAIN RMSE	AVG TEST RMSE	AVG TEST RMSE	AVG TRAIN MAE	AVG TEST MAE	AVG TEST MAE	Test p-value for RMSE	
10	MLP-64	18 049	45 498	5 714	18 002	48,798	**134,947**	**41,942**	6,985	**11,617**	**6,476**	2,737	7,277	**2,878**	N/A	
10	MLP-64-CS	22 273	45 498	5 714	18 002	48,865	135,368	42,096	6,990	11,635	6,488	2,753	7,256	2,896	0,12	
10	MLP-64-AS3	18 049	45 498	5 714	18 002	48,680	135,657	42,152	6,977	11,647	6,492	2,738	**7,255**	2,887	0,013	*
10	MLP-64-AS2	18 049	45 498	5 714	18 002	50,253	136,089	42,496	7,089	11,666	6,519	2,829	7,292	2,960	1,76E-06	***
10	MLP-64-AS	18 049	45 498	5 714	18 002	**48,528**	136,417	42,477	**6,966**	11,679	6,517	2,752	7,296	2,919	2,11E-04	***
10	MLP-64-AS2-LN	18 049	45 498	5 714	18 002	49,204	139,646	43,551	7,014	11,817	6,599	**2,734**	7,452	2,960	1,11E-25	***
10	MLP-64-AS2-BN	18 049	45 498	5 714	18 002	48,814	140,216	43,718	6,987	11,841	6,612	2,748	7,433	2,957	4,38E-21	***
10	MLP1	1 079	45 498	5 714	18 134	56,735	142,116	45,479	7,532	11,921	6,744	3,192	7,273	3,256	2,04E-44	***
2	Linefit	2	46 104	5 780	18 134	59,433	146,337	46,518	7,709	12,097	6,820	3,333	7,739	3,327	-	
3	Linefit	2	46 104	5 780	18 134	57,289	147,883	48,194	7,569	12,161	6,942	3,381	7,516	3,477	-	
1	PPV	N/A	46 104	5 780	18 134	79,927	194,461	58,836	8,940	13,945	7,670	3,354	9,080	3,298	-	

Appendix M. Network performance with various feature sets

Table 26. Results of testing with various feature sets.

Room A						Dataset size			Data Selection method: all = all data used; var = only data with high variation was used.													
									all		var		all		var		all		var		all	
Input Time Steps	Model	Params	Dropout	Feature Set	Train size	Test Size	TEST_ALL Size	AVG TRAIN MSE	AVG TEST MSE	AVG TEST MSE	AVG TRAIN RMSE	AVG TEST RMSE	AVG TEST RMSE	AVG TRAIN MAE	AVG TEST MAE	AVG TEST MAE	for RMSE
7	MLP-64	~18 500	0,03	2	49 740	6 176	18 926	37,987	**114,253**	**36,785**	6,163	**10,688**	**6,065**	2,531	**6,875**	**2,817**	N/A
7	MLP-64	~18 500	0,03	3	49 740	6 176	18 926	37,610	115,494	37,250	6,132	10,746	6,103	2,523	6,945	2,842	0,033 *
7	MLP-64	~18 500	0,03	4	49 740	6 176	18 926	37,480	116,319	37,570	6,122	10,784	6,129	**2,513**	6,985	2,855	0,001 ***
7	MLP1-WIDE	18 571	0,02	2	49 740	6 176	18 926	44,825	118,728	38,733	6,695	10,896	6,224	2,894	6,921	3,062	2,30E-16 ***
7	MLP-64	~18 500	0,04	7	49 740	6 176	18 926	38,929	123,690	48,706	6,238	11,119	6,965	2,589	7,519	3,766	1,09E-14 ***
7	MLP-64	~18 500	0,03	9	49 740	6 176	18 926	37,782	124,664	49,396	6,146	11,164	7,023	2,597	7,632	3,896	8,47E-25 ***
7	MLP-64	~18 500	0,02	5	49 740	6 176	18 926	41,367	124,807	40,820	6,432	11,171	6,389	2,584	7,245	2,935	1,88E-23 ***
7	MLP-64	~18 500	0,03	6	49 740	6 176	18 926	37,950	125,147	43,501	6,160	11,185	6,593	2,543	7,483	3,275	7,70E-20 ***
7	MLP-64	~18 500	0,03	10	49 740	6 176	18 926	**37,465**	125,716	50,097	**6,121**	11,210	7,070	2,585	7,610	3,856	1,34E-16 ***
7	MLP-64	~18 500	0,03	8	49 740	6 176	18 926	37,847	127,155	50,730	6,152	11,274	7,113	2,569	7,671	3,857	9,53E-23 ***
7	MLP-64	~18 500	0,03	1	49 740	6 176	18 926	49,452	131,749	41,927	7,032	11,478	6,475	2,760	7,194	2,883	3,25E-47 ***
7	MLP1	1 471		2	49 740	6 176	18 926	51,849	136,111	44,771	7,200	11,665	6,690	3,130	7,417	3,294	4,90E-34 ***
2	Linefit	2		CO2	50 346	6 242	19 058	60,887	143,629	46,850	7,803	11,985	6,845	3,382	7,648	3,354	-
3	Linefit	2		CO2	50 346	6 242	19 058	58,707	145,927	48,736	7,662	12,080	6,981	3,418	7,464	3,510	-
1	PPV	N/A		CO2	50 346	6 242	19 058	81,919	188,741	58,700	9,051	13,738	7,662	3,416	8,905	3,313	-

Appendix N. Abbreviations

Adam = Adaptive Moment Estimation. A commonly used optimizer when training neural networks.

ANN = Artificial Neural Network

ARIMA = Autoregressive integrated moving average

BN = Batch normalization

BP = Backpropagation

CGAN = Conditional Generative Adversarial Network

CNN = Convolutional neural network

CO2 = Carbon dioxide

DBN = Deep Belief Network. DBN is a network that contains several RBMs.

Dense = a Layer of perceptrons

DNN = Deep Neural Network. A network that has two or more hidden layers ("Deep learning", n.d.).

DSR = Design Science Research

ELU = Exponential Linear Unit

GRU = Gated Recurrent Unit

HVAC = Heating, ventilation, and air conditioning

LF = Line fit model

LN = Layer normalization

LSTM = Long short-term memory

MAE = Mean absolute error

ML = Machine Learning

MLP = Multilayer perceptron

MSE = Mean squared error

NLPD = Negative Log Predictive Density

PIR = Passive infrared sensor. Also called: Motion detection

PPV = Previous Value Forward prediction

RBM = Restricted Boltzmann Machine

SBS = Sick building syndrome

SE = Standard error

SGA = Stochastic Gradient Ascent

TanH = Hyperbolic Tangent

VOC = Volatile Organic Compound

Appendix O. Training data amount

Table 27. The effects of removing some of the training data.

Room A					Dataset size			Data Selection method: all = all data used; var = only data with high variation was used.																
								all	var	all	var	all	var	all	var	all	var	all	var	all	var	all	var	
Input Time Steps	Model	Params	DROP_RATE	Train size	Test Size	TEST_ALL Size	AVG TRAIN MSE	AVG TEST MSE	AVG TRAIN RMSE	AVG TEST RMSE	AVG TRAIN MAE	AVG TEST MAE	AVG TRAIN MSE	AVG TEST MSE	AVG TRAIN RMSE	AVG TEST RMSE	AVG TRAIN MAE	AVG TEST MAE	AVG TRAIN MSE	AVG TEST MSE	AVG TRAIN RMSE	AVG TEST RMSE	AVG TRAIN MAE	AVG TEST MAE	Test p-value for RMSE	
10	MLP-64	18 049	0,06	45 498	5 714	18 002	48,798	**134,947**	6,985	**11,617**	2,737	7,227	48,798	**134,947**	6,985	**11,617**	2,737	7,227	48,798	**134,947**	6,985	**11,617**	2,737	7,227	N/A	
10	MLP-64	18 049	0,14	27 242	5 714	18 002	40,306	135,384	6,348	11,635	2,549	7,221	42,183	42,282	6,258	11,664	2,418	**7,207**	40,306	135,384	6,348	11,635	2,549	7,221	0,217	
10	MLP-64	18 049	0,06	36 352	5 714	18 002	39,159	136,060	6,258	11,664	2,418	**7,207**	44,731	46,518	6,688	12,097	**2,366**	7,349	39,159	136,060	6,258	11,664	2,418	**7,207**	**2,853**	1,12E-04***
10	MLP-64	18 049	0,12	18 155	5 714	18 002	**36,320**	143,016	**6,026**	11,959	2,366	7,349	**36,320**	143,016	**6,026**	11,959	2,366	7,349	**36,320**	143,016	**6,026**	11,959	2,366	7,349	3,000	1,35E-39***
2	Lineft	2		46 104	5 780	18 134	59,433	146,337	7,709	12,097	3,333	7,739	59,433	146,337	7,709	12,097	3,333	7,739	59,433	146,337	7,709	12,097	3,333	7,739	3,327	-
3	Lineft	2		46 104	5 780	18 134	57,289	147,883	7,569	12,161	3,381	7,516	57,289	147,883	7,569	12,161	3,381	7,516	57,289	147,883	7,569	12,161	3,381	7,516	3,477	-
10	MLP-64	18 049	0,14	9 047	5 714	18 002	52,441	150,080	7,239	12,250	2,938	7,498	52,441	150,080	7,239	12,250	2,938	7,498	52,441	150,080	7,239	12,250	2,938	7,498	3,124	3,36E-28***
10	MLP-64	18 049	0,14	4 532	5 714	18 002	39,069	180,435	6,248	13,429	2,772	8,236	39,069	180,435	6,248	13,429	2,772	8,236	39,069	180,435	6,248	13,429	2,772	8,236	3,640	1,50E-32***
1	PPV	N/A		46 104	5 780	18 134	79,927	194,461	8,940	13,945	7,670	9,080	79,927	194,461	8,940	13,945	7,670	9,080	79,927	194,461	8,940	13,945	7,670	9,080	3,298	-
10	MLP-64	18 049	0,12	2 271	5 714	18 002	79,319	201,185	8,901	14,159	8,249	8,781	58,834	68,314	6,248	8,940	4,709	4,368	79,319	201,185	8,901	14,159	8,249	8,781	4,368	2,75E-21***