



Leonhard Leopold, BSc.

# **Wear Prediction of Refractory Lining Using Neural Networks**

## **Master's Thesis**

to achieve the university degree of  
Master of Science

Master's degree programme: Computer Science

submitted to

**Graz University of Technology**

Supervisor

Univ.-Prof. Dipl.-Ing. Dr. mont. Franz Pernkopf

Institute of Signal Processing and Speech Communication

Graz, September 2022



# Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

---

Date

---

Signature



# Abstract

In the steel making process, vessels and containers need to be protected from the extreme temperatures reached during the production. Thus, these metallurgical vessels are lined with bricks. Over time this lining wears down and needs to be replaced. It is vital to recognise high abrasion to avoid a breakage of the vessel which could be devastating during steel production. Costly laser measurements can be made to determine the state of the remaining bricks. However, for some vessels this is not possible as the environment is hostile and they cannot be opened to correctly conduct these measurements. With the purpose of offering a more cost-effective alternative, this thesis aims to use sensor measurements made during the process to predict the wear of the refractory lining. Since, multiple statistical approaches have been tested, the focus lies specifically on neural networks to improve the predictions. A myriad of neural network architectures were assessed with the aim of finding the best performing model. Multiple data sets with diverse measurements from different manufacturers were evaluated. The findings show that this approach improves the performance on most of the data sets while providing competitive results for all of them which suggest that predicting the wear of the lining of metallurgical vessels is a viable alternative. A 2-dimensional convolutional neural network was used to model the refractory wear. The results improved after applying pre-processing steps like a Gaussian filter to smooth the data and by selecting the most impactful features using SHAP values.



# Contents

<b>1</b>	<b>Introduction</b>	<b>11</b>
1.1	Problem Description . . . . .	11
1.2	Goal and Motivation . . . . .	12
1.3	Summary and Thesis Outline . . . . .	13
<b>2</b>	<b>Background and Related Work</b>	<b>15</b>
2.1	Background . . . . .	15
2.2	Related Work . . . . .	15
2.3	Summary . . . . .	17
<b>3</b>	<b>Data set</b>	<b>19</b>
3.1	Data Structure . . . . .	20
3.2	Summary . . . . .	22
<b>4</b>	<b>Fundamental Methods</b>	<b>23</b>
4.1	Artificial Neural Networks . . . . .	23
4.2	Convolutional Neural Networks . . . . .	25
4.3	Recurrent Neural Networks - LSTMs and GRUs . . . . .	26
4.4	Residual Neural Networks . . . . .	28
4.5	MixUp Data Augmentation . . . . .	29
4.6	Gaussian Filter . . . . .	30
4.7	Feature Selection with SHAP Values . . . . .	31
4.8	Summary . . . . .	34
<b>5</b>	<b>Modelling Approaches</b>	<b>35</b>
5.1	Model Training Process . . . . .	35
5.1.1	Prepossessing . . . . .	35
5.1.2	Training . . . . .	37
5.1.3	Evaluation . . . . .	39
5.2	Model Development Process . . . . .	39
5.2.1	Initial Models . . . . .	40
5.2.2	Fine Tuning . . . . .	42
5.2.3	Process of Finding the Best Models . . . . .	44
5.3	Summary . . . . .	46
<b>6</b>	<b>Results</b>	<b>47</b>
6.1	Metalzone and Slagzone Data . . . . .	47
6.2	Heart, Bottom Inlet, Bottom Outlet Data . . . . .	53
6.3	Comparison . . . . .	57

6.4 Summary . . . . .	58
<b>7 Conclusion</b>	<b>59</b>
<b>8 Outlook and Future Work</b>	<b>61</b>
<b>Bibliography</b>	<b>63</b>
<b>Acronyms</b>	<b>68</b>



# List of Figures

3.1	Structure of a RH degasser (taken from [12]). . . . .	19
3.2	Cross section of a RH degasser (taken from [12]). . . . .	20
4.1	Composition of a single artificial neuron (taken from [16]). . .	23
4.2	Architecture of simple neural networks and deep neural networks (taken from [18]). . . . .	24
4.3	Example of a 2 dimensional convolution (taken from: [25]). . .	26
4.4	Recurrent neural network (taken from: [27]). . . . .	27
4.5	Building block of a residual network (taken from: [31]). . . . .	28
4.6	Beta distribution with $\alpha = \beta = 0.2$ . . . . .	29
4.7	Feature is filtered with Gaussian with different $\sigma$ values. . . .	30
4.8	SHAP values for image classification (taken from: [41]). . . . .	32
4.9	Visualization of the calculated SHAP values on some of the campaigns in the test set using the Metalzone data set. . . . .	33
5.1	The bias-variance tradeoff - overfitting and underfitting (taken from: [44]). . . . .	38
5.2	Initial comparison of different model architectures. . . . .	42
5.3	Different model architectures are trained and compared using the Metalzone set. . . . .	46
6.1	Architecture of the 2D CNN with 6 layers. . . . .	48
6.2	Architecture of the 2D CNN with 2 layers. . . . .	48
6.3	Comparison of model performances using different amount of features using the Metalzone data set. . . . .	50
6.4	Residual plot of the results of the Metalzone data set using 5 features and a two layer 2D CNN. . . . .	51
6.5	Residual plot of the results of the Slagzone data set using 5 features and a two layer 2D CNN. . . . .	52
6.6	Architecture of the GRU model. . . . .	53
6.7	Comparison of model performances using different amount of features - Bottom Inlet data set. . . . .	55
6.8	Residual plot of the results of the Bottom Inlet data set using 6 features and a GRU ResNet. . . . .	56
6.9	Residual plot of the results of the Bottom Outlet data set using 6 features and a GRU ResNet. . . . .	56
6.10	Residual plot of the results of the Heart data set using 6 features and a GRU ResNet. . . . .	57

# List of Tables

3.1	Structure of a single campaign . . . . .	21
3.2	Properties of all used data sets. . . . .	22
6.1	Top 15 Mean Absolute SHAP Values - Metalzone. . . . .	49
6.2	Top 15 Mean Absolute SHAP Values - Slagzone. . . . .	49
6.3	Top 15 Mean Absolute Shap Values - Bottom Inlet. . . . .	54
6.4	Top 15 Mean Absolute Shap Values - Bottom Outlet. . . . .	54
6.5	Top 15 Mean Absolute Shap Values - Heart. . . . .	54
6.6	Comparison of different approaches on all data sets. . . . .	57

# 1 Introduction

In the steel industry metallurgical vessels are used in a variety of application like transportation or steel refinement. The metal is mostly handled in its liquid form which is reached at high temperature of around  $1600^{\circ}\text{C}$ . Thus, the used vessels need to be able to withstand the heat. The vessels are lined with a layer of bricks in order to prevent a breakage. However, the lining wears out over time and after repeated usage. Hence, this lining must be replaced after a certain amount of times the vessel has been used. The remaining thickness of the lining can be measured using laser scans. This approach, however, is rather cost-intense and is infeasible for certain vessels during production due to the hostile environment. An alternate approach would be to estimate the state of the lining based on other sensor measurements or process data made during the production process. Thus, the focus of this thesis is to predict the state of the remaining refractory material on the basis of several different other sensor measurements available during production.

## 1.1 Problem Description

The lining of metallurgical vessels is vital in providing insulation from the heat during the steel-making process. After repeated usage, the lining wears off. Currently, a common approach to measure the remaining thickness of the refractory lining is done by using lasers [1]. This is then used to assess how many times the vessel can be used before the lining needs to be changed.

Unfortunately, it is often not possible to measure the state of the lining this way. In some places of the vessels the refractory lining is not accessible due to its integration in the system. In many cases, installing measurement equipment is impractical or simply very expensive. Just like in any other industry, if there is a cheaper solution that produces similar results, it is usually preferred [2].

Due to the vastly different production environments and setups across the industry, it is infeasible to build a physical model to test and evaluate the wear over time. Thus, using a statistical approach in order to estimate the remaining refractory lining, without measurements done by expensive equipment, is a promising approach. The prediction can be calculated on the basis of several sensor measurements that are made during the usage of the vessel like temperatures, durations or chemical compositions. The

different input features are discussed in detail in Chapter 3.1.

However, using this approach one significant problem arises. On one hand, these measurements utilized as independent variables for the prediction are collected every time the vessel is used. On the other hand, the dependent variable, meaning the target of the prediction can only be assessed post-mortem. This means that the actual remaining thickness of the refractory martial can solely be inspected after the vessel goes out of service in order to replace the lining. Hence, it is difficult to predict the effect a single treatment has on the used vessel, since the target variable is an aggregation of multiple treatments using the vessel.

## 1.2 Goal and Motivation

It is crucial to be accurate with measuring or predicting the wear of the refractory lining in vessels used in the steel industry. If mistakes are made, it is possible for the molten metal to burn through the vessel, which represents a major safety hazard and could endanger working staff, instruments and equipment while causing substantial financial losses [1]. Further, unplanned down-times additionally reduce the yield of the manufacturer.

The steel industry is a very old business and constantly changes over time, while the demand of steel is steadily rising [3]. The change from manual to machine production in the First Industrial Revolution was followed by two others with the addition of electricity as well as faster transportation methods and later the inclusion of the computer in the production process [4].

In the 21<sup>st</sup> century most sectors of industry focus on incorporating more modern technology. Productivity in manufacturing increased due to the incorporation of technologies like Cloud-based solutions, Artificial Intelligence and Internet of Things. This trend is designated as Industry 4.0 [5][6].

In the spirit of this trend, we provide in this thesis a modern solution to an old problem encountered during the steel production process. With the usage of machine learning, a model is created that aims to accurately predict the remaining lining thickness of a metallurgical vessel. This provides an affordable and safe solution assuming accurate prediction models. Moreover, accurately predicting the lifetime of the lining could potentially prolong the usage of the vessel, making this solution even more cost-effective.

### 1.3 Summary and Thesis Outline

In this chapter, the difficulties in measuring the refractory lining wear in the steel industry were introduced. Furthermore, possible dangers of failure in this process are laid out. In the wake of the currently ongoing 4<sup>th</sup> Industrial Revolution, an approach that utilises machine learning to predict the remaining state of the lining with the help of sensor measurements available during the production process, was proposed. In Chapter 2, some background knowledge is established, and related work in this area is highlighted. In Chapter 3, the data sets are explained in detail before all methods tested to solve the problem are covered in Chapter 4. Many different approaches were compared in Chapter 5. Finally, in Chapter 6 the best models that were found are showcased and the results are presented. Lastly, in Chapter 7 the thesis is concluded before an outlook of possible further approaches is discussed in Chapter 8.



## 2 Background and Related Work

### 2.1 Background

During the steel-production various substances like iron ore are combined in a blast furnace that heats and combine them to make crude (or pig-) iron [7]. Afterwards, the produced goods are transferred into a so-called converter that applies a process named 'basic oxygen furnace process' (BOF) or into an electric arc furnace (EAF). Here, oxygen is blown through lances into the molten pig iron. This lowers the carbon content in the mixture, thus producing low-carbon steel. This can then be refined and used to create various steel products [8]. One of these cycles of, for instance, making steel from molten pig ore is referred to as a heat. The same is the case for other steel manufacturing machinery like vessels used for the transportation of liquid steel in the plant. The data set of this thesis, discussed in Chapter 3, consists of several measurements made during these heats.

During this procedure extremely high temperatures of around 1600°C can be reached. This creates a challenge in the construction of metallurgical vessels like a converter. Therefore, these structures are lined with refractory material like bricks to insulate. However, this lining of bricks wears out over time and needs to be replaced [9]. The state of the refractory material after several usages is referred to as Remaining Brick Length (RBL). Usually, this is measured using lasers. As already stated in the Problem Description in Chapter 1.1, this is often expensive or infeasible [2].

It is vital to be aware of the state of the remaining lining, since a breakthrough of hot liquid metal is naturally extremely dangerous. Whenever, the RBL reaches a critical threshold, the lining is replaced. One such cycle of refractory usage is designated as a single campaign [10].

### 2.2 Related Work

Here, work is highlighted that deals with similar problems as in this thesis. Some papers focus on more general topics like the different application areas where abrasive wear in the steel-industry causes problems like containers that come in contact with extreme temperatures or cutting tools that wear out during their usage.

Refractory lining in the steel-industry is not the only application where measuring the wear is an issue. Shah, Vakharia, Chaudhari et al. [11] face

the problem of predicting the wear of cutting tools. They also wear off over time and need to be replaced. They can be visually inspected to determine the wear. However, the milling process needs to be stopped in order to do this, which delays the production. Like this work, they use neural networks to estimate the wear without interrupting the process. The measurements taken are acoustic and vibration signals. They use these to try to accurately predict the wear on the cutting tool.

When focusing exclusively on the steel industry, Varga [9] detailed abrasive wear in different areas in production. This is the case especially whenever high temperatures are involved. The goal of the work is to identify core components that led to failure and high maintenance efforts, not just the lining of metallurgical vessels. The work analyses the most volatile parts of a sinter plant to find which components lead to most of the downtime and replacement costs.

Specifically related to refractory lining wear measurements in steel production, Väinämö and Röning [1] face a problem during the laser scanning process of the lining. The exact positioning of the vessel is done using CCD-cameras and multiple specially located ring-shaped objects on its surface. At different angles the position of rings are not recognised precisely. Using neural networks the authors attempt to reduce an error that arises. This is another problem that arises when using laser measurements during the production. This thesis aims to find an alternative solution.

Lastly, there are multiple works that deals with the same problem as this thesis, but uses different methods of prediction. Forrer [10] wrote about the wear prediction of refractory lining in steel-production vessels. Contrary to this thesis, other methods of prediction were used. Linear regression, support vector regression and Gaussian processes were applied. It was found that the linear regression model performed the best out of these approaches. The author also points out that the size of the data set has a significant impact on the performance.

Mutsam and Pernkopf [2] try to tackle the same problem as this thesis. Their methodology differs, however. An iterative least squares approach and one based on convex optimization using aggregated features and linear inequality constraints are used. In an unreleased report, they also tested a myriad of other approaches. Since the same data set was used as in this thesis, their work represents a perfect opportunity to compare the results achieved in this thesis. This comparison is shown in Chapter 6.3.



### 2.3 Summary

In this chapter, some necessary background knowledge and terminology from the steel industry was introduced. A campaign consists of multiple heats and the RBL is measured only at the end of campaign. This is important in the next chapter where the data set is introduced.

Afterwards, related work was highlighted. This overview ranged from wear prediction in other application areas to different issues during steel production. Also, work that tackled the same problem as this thesis, but used different approaches, was highlighted.



### 3 Data set

In this chapter, the structure of the data set is introduced. It actually consists of two different data sets from two different companies and plants. However, even though vastly different measurements were made, the goal of predicting the wear on the refractory lining is the same.

The first data set originates from the Jindal Shadeed Iron and Steel company. Here, measurements were made in two areas of a vessel. This vessel is used to transport metal products and can be separated into the metalzone and slagzone. The slag sits at the top of the metal and is a by-product of steel production when smelting metals. Slag is more abrasive and usually cause more wear over time. In both areas, measurements are made and the remaining lining is only inspected post-mortem. The second data set

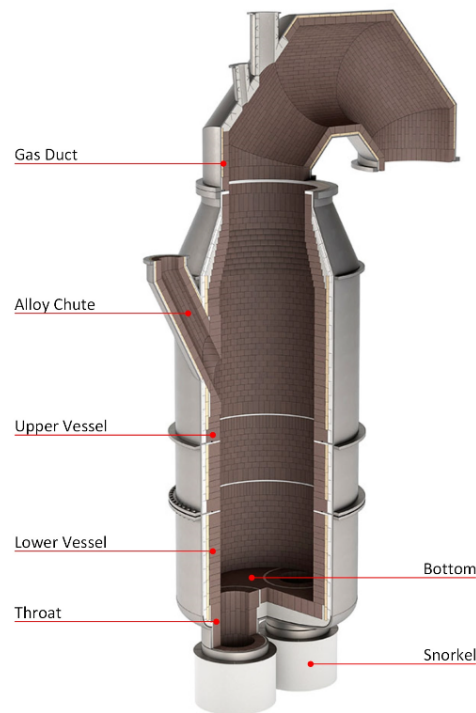


Figure 3.1: Structure of a RH degasser (taken from [12]).

is provided by the voestalpine Stahl GmbH in Linz and consists of three subsets of measurements made in a Ruhrstahl-Heraeus (RH) degasser. Here, in this application, steel is refined as detailed in Chapter 2. The same myriad of measurements are made at three different positions in the metallurgical vessel. These three subsets are located at the bottom inlet, bottom outlet and

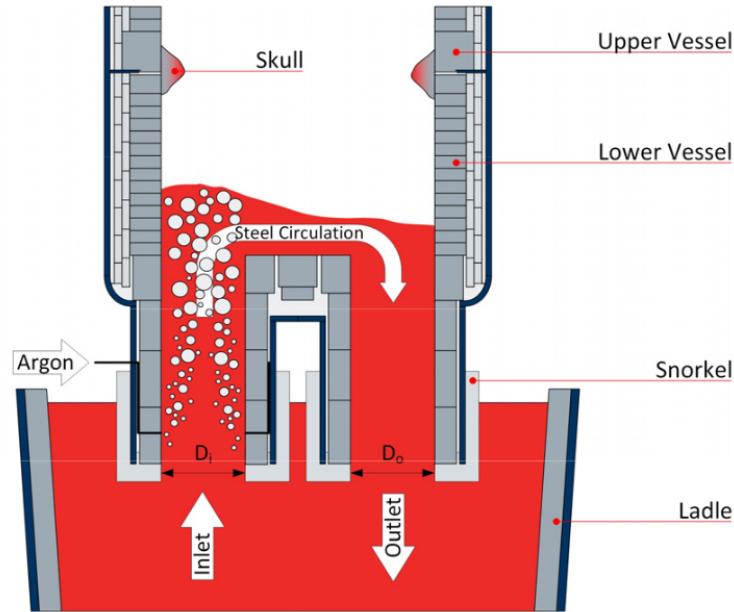


Figure 3.2: Cross section of a RH degasser (taken from [12]).

the heart of the vessel. Figure 3.1 and Figure 3.2 show the structure of a RH degasser. The bottom inlet and outlet are supporting pillars of the RH vessel where the liquid metal is injected and released as visualized in Figure 3.2. The heart is at a central position of the bottom of the container which is labeled as 'Bottom' in Figure 3.1.

### 3.1 Data Structure

The data consists of sensor measurements taken during processing at every single heat. These feature are highly varying types of measurements. Some temperatures are assessed at different areas of the vessel or different times during processing. Additionally, the starting and ending temperatures are used in order to represent some of the progression during the process. Other than that, multiple different chemical measurements were made. For instance, the prevalence of elements like nitrogen, argon, carbon, aluminium, calcium and others are included. Once again, temporal distinctions are made with most of these measurements to show how they change over time. Additionally, the data includes the duration of certain stages of the process and the overall time the heat requires.

It is not exactly known which features have an actual impact on the result. Thus, most available measurements were included with the assumption that

some might correlate with the lining wear. Consequently, feature selection is an prominent topic in this thesis. When training a model on features that are mostly noise, some non-relevant patterns might be learned. If only the most useful features are used, the model should in theory produce better results [13]. This is further discussed in Chapter 4.7.

The input features are provided in 3 dimensional matrices. The first dimension represent the different campaigns. They are used as separate data point during training. The other two dimensions are fed to the models to find patterns. The second dimension are the heats. The difficulty is that every campaign consists of a different amount of heats. This is because the campaign is ended and the refractory lining is changed after a varying number of usages. The third dimension represents the different features, i.e. sensor measurements made at every heat. So, at every campaign during a specific heat, a single measurement is made. Thus, a single campaign has the following structure shown in Table 3.1. Each data set is composed of campaigns, like in this example. The different data set contain varying

Heats	Feature #1	Feature #2	Feature #3	Feature #.	Feature #.
Heat #1	0.15	0.95	0.41	.	.
Heat #2	0.98	0.87	0.57	.	.
Heat #3	0.11	0.46	0.19	.	.
Heat #.	.	.	.	.	.
Heat #.	.	.	.	.	.

Table 3.1: Structure of a single campaign

numbers of campaigns, heats per campaign and sensor measurement made per heat. Generally, about 50-60 sensor measurements are included at every heat. The number of heats per campaign mostly ranged from 100 to 150. A mayor issue was the amount of data provided, since the available data contained only around 100 to 200 campaigns per data set.

In total 5 different data sets where used from 2 different plants. There differences in structure is summarized in Table 3.1. This prediction problem is a supervised learning task. Here, labels are needed as targets during training. The structure of the target set is much simpler than the feature set. Every campaign has a single scalar target value. This value is the RBL measured after the refractory lining was removed and replaced. Thus, each campaign consists of a 2D Matrix of multiple measurements taken across many heats and a single target, which is the RBL at the end of the campaign. This thesis aim is to find the correlation between these two.

Data set	# Features	# Campaigns	# Heats	Avg # Heats
Metalzone	53	137	61-161	119.6
Slagzone	53	273	18-98	60.5
Bottom Inlet	62	125	80-226	162.8
Bottom Outlet	62	128	80-226	162.8
Heart	62	132	80-221	162.7

Table 3.2: Properties of all used data sets.

## 3.2 Summary

In this chapter, the structure and properties of the data sets have been introduced. The feature set contains multiple campaigns that consist of varying numbers of heats, where a variety of sensor measurements are available. The target for each campaign is a single scalar value representing the RBL which was measured at the end of the campaign when the refractory lining was replaced.

## 4 Fundamental Methods

In this chapter, the different methods are discussed. This includes all the different neural network archetypes used, as well as certain pre-processing steps. Small alterations to layers like regularizers and different layers that may increase the performance, like Pooling or Batch Normalization layers are not covered here in detail, but will be discussed in Chapter 5.2.2.

### 4.1 Artificial Neural Networks

Artificial neural networks (ANNs) are a complex network of neurons that can be used to solve nonlinear problems. These neurons are inspired by the functionality of the brain and are thus able to adapt to experiences in order to better achieved certain objectives. Grouped together in a network, they can be used to approximate optimal solutions for problems like regression or classification. The building blocks of ANNs are neurons as visualized in Figure 4.1 that weight multiple inputs and calculate the output by applying an activation function. The output  $y$  is defined as

$$y = \sigma\left(\sum_{i=1}^n x_i \cdot w_i\right) + b$$

where  $n$  is the number of inputs.  $x_i$  is one of the inputs and  $w_i$  its corresponding weight and  $b$  denotes the bias.  $\sigma$  defines the activation function. This can be the linear identity function  $\sigma(x) = x$  which, with a single neuron, would represent a linear regression model. However, with a non-linear activation function, like ReLU [14]  $\sigma(x) = \max(0, x)$ , sigmoid [15]  $\sigma(x) = \frac{1}{1+e^{-x}}$  or tangens hyperbolicus (tanH) more complex tasks and can be learned.

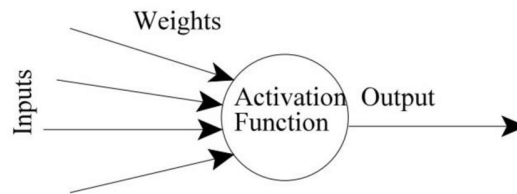


Figure 4.1: Composition of a single artificial neuron (taken from [16]).

The output of such a neuron is a scalar value which can be used as one of the inputs to another neuron in an subsequent layer [17]. This is the basic building block of neural networks, which consist of multiple layers of

neurons that are interconnected in a way where the inputs of a neuron are the outputs of each of the neurons of the previous layer. Neural networks with multiple hidden layers are referred to as deep neural networks as visualized in Figure 4.2. These architectures are flexible in the number of inputs and outputs they can have and thus can be used for regression, multi-class classification tasks, but also for application like our wear prediction, where multiple features are used as input to predict a single target value. The model learns by updating the weights of each neuron starting from

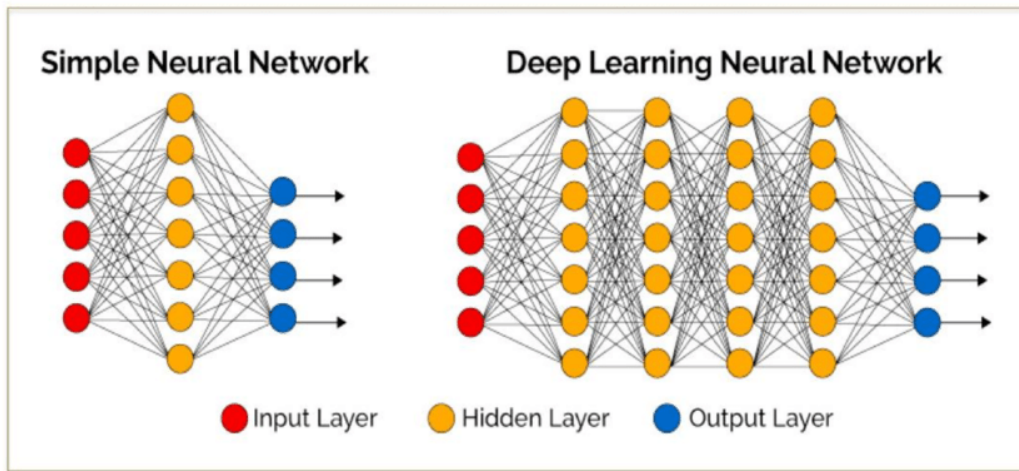


Figure 4.2: Architecture of simple neural networks and deep neural networks (taken from [18]).

the last layer in a procedure called back-propagation, which is a gradient-descent based optimization approach used for ANNs. A loss function is used to evaluate the error of the model. It is chosen depending on the task, but usually measures the difference between the model prediction and the expected value. Based on the loss function, gradient descent is used after each epoch of training to update the weights. The rate at which the weights in the model change is dictated by the learning rate. Choosing a correct learning rate is important since a high learning rate might overshoot the optimal solution while it takes longer for the model to converge when using a low learning rate. Many methods for adapting the learning rate during the training have been presented [19] [20]. Over time, the neural network model learns patterns in the input data and is able to more accurately predict the output [21].



## 4.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a variant of ANNs that make use of convolution kernels to learn patterns in 2D image-like data. Instead of multiplying the weights with the input vector, CNNs utilize its weights in kernels that are applied on all of the input, thus reducing the amount of parameters that need to be trained. The A 2D convolutional kernel  $\mathbf{K}(\cdot, \cdot)$  is a matrix of dimension  $K \times K$ . It moves across the input data and determines whether a certain feature is present. It does this by calculating the output  $y[m, n]$  at  $m$  and  $n$  as

$$y[m, n] = \mathbf{X} * \mathbf{K} = \sum_i^K \sum_j^K X[m + i, n + j] \cdot L[i, j]$$

where  $x(m, n)$  is the element at  $m$  and  $n$  of the input matrix  $\mathbf{x}$  of size  $M \times N$ ,  $\mathbf{k}$  is the kernel matrix of size  $K \times K$ ,  $y[m, n]$  is the output at position  $m$  and  $n$ . (e.g.  $m = 0, n = 0 =$  top left). The kernel moves across the input data and results in higher values if the values of the input data includes the pattern the kernel wants to emphasize. Doing this process for multiple times in a layered fashion (i.e. the output of the CNN layer is the input for the next layer) enables a CNN to recognise complex objects. In the instance of object recognition, the first layer might use kernels to emphasize straight lines or corners. Then the next layer might learn to find circles or squares. This can be repeated until complex objects can be recognised [22]. If continued further and trained on enough input data this method is able to, for instance, discern the difference between different animal species [23]. Recent research also shows that CNNs are able to outperform humans when specialized on finding certain patterns. Prominently, they are also used in the medical field in order to identify forms of cancer [24].

As shown in Figure 4.3, the output of a convolutional layer is smaller than the input. The output size is given by  $M_{out} = ((M - k + 2P)/S) + 1$  and  $N_{out} = ((N - k + 2P)/S) + 1$   $y = ((x - K + 2P)/S) + 1$  where  $M_{out}$  is the height and  $N_{out}$  the width of the output. The input  $\mathbf{X}$  has the dimensions  $M \times N$  and the kernel  $\mathbf{k}$  with size  $K \times K$ .  $P$  is the padding size, which can be used to increase the size of the output to possibly apply more layers of convolutions. Padding can be performed with zeros or by mirroring the border values.  $S$  denotes the stride which dictates the number of elements the kernel moves after each calculation. When a stride of more than 1 is used, the output size is drastically reduced. This can be used to obtain a more concentrated output. Typically, CNNs are used for images due to convolutions kernels being a good way of capturing objects in 2D inputs.

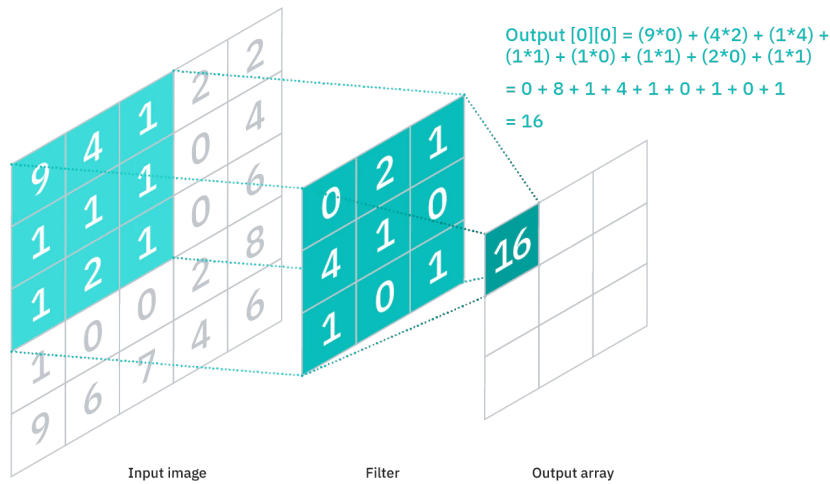


Figure 4.3: Example of a 2 dimensional convolution (taken from: [25]).

However, inputs with other dimensionalities are also possible. A prominent example are images with different color channels. CNNs train kernels for each of these channels [25].

This approach can also be used to find patterns in non-image data. Due to the nature of CNNs, patterns can be processed invariant of its spatial position in the input data which means it is a valuable tool in analyzing time series data. Above, the focus was on 2D CNNs. However, the same can be done with one-dimensional kernels and input data. Both approaches have been tested in this thesis. As stated in Chapter 3.1, the data set consists of a series of different sensor measurements over time. In order to find patterns in these measurements or to detect spikes invariant to its position, CNNs might be a promising approach for this problem.

### 4.3 Recurrent Neural Networks - LSTMs and GRUs

Recurrent neural networks are used for sequential data. As shown in Figure 4.4, this approach interconnects inputs in a sequential manner in the network. In the case of text inputs, the previous words influence the current input to the model. This is valuable to capture the context of a certain word. However, this approach suffers from issues like the vanishing gradient problem [26], where during back-propagation the gradient of a layer is getting increasingly smaller if the effect of the previous layer is insignificant. Thus,

the weights will not be updated appropriately during training. In order to combat this problem, long-short-term-memory (LSTM) or gated recurrent unit (GRU) layers are used. They utilize memory cells to store previous inputs and gates to decide what do to with these inputs [27]. LSTMs have

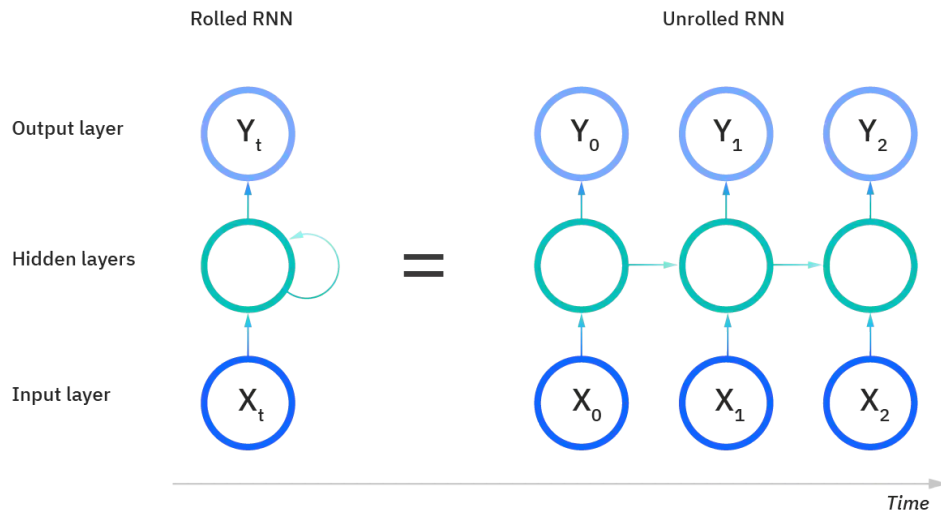


Figure 4.4: Recurrent neural network (taken from: [27]).

been introduced in 1997 [28]. They utilise three different gates, each having their own weights that determine how they should act after being trained on the input data. These three gates are the input gate, forget gate and output gate, which determine what information is kept, what data will be updated and what the value of the next hidden state will be [29].

GRUs are a novel lightweight alternative to LSTMs introduced in 2014 [30]. GRUs only have two gates, the reset and update gate while omitting the three gate structure of LSTMs. The update gate replaces the functionality of the input and forget gate of the LSTM. This enables it to be trained with fewer parameters which allows for faster training times. However as stated by Chung, Gulcehre, Cho et al [29], in terms of performance neither approach outperforms the other consistently. The best choice depends on the task at hand.

Both of these architectures might be valuable since the sensor measurement in the data set are time series data. This approach is expected to learn that a single heat might only cause substantial loss in refractory lining when considering the context with other heats. This means that the relationship among the heats for predicting the wear is taken into account by these models.

## 4.4 Residual Neural Networks

In contrast to Recurrent neural networks, residual neural networks introduce a different solution for the vanishing gradient problem. In particular in very deep neural networks, this variant of neural networks is a prominent solution to keep the performance high. As shown in He, Zhang, Ren et al. [31], the training and test error increased for fully connected artificial neural networks when the number of layers increases. This is due to the aforementioned vanishing gradient problem.

In residual neural networks (ResNet), so-called skip connections. These blocks take the output of a layer and combine it with the output of another layer while skipping several layers in-between. An example of this can be seen in Figure 4.5. This improves the performance since it simplifies the

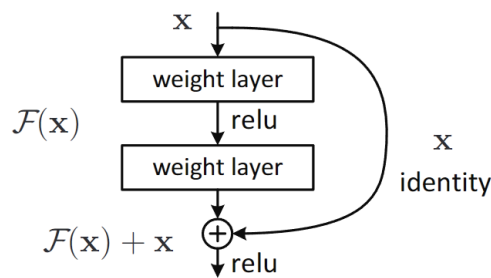


Figure 4.5: Building block of a residual network (taken from: [31]).

network and potentially allows it to skip layers that are a detriment to the model. As a result, a deep architecture does not negatively influence the result while allowing the model to learn more complex patterns.

Residual neural networks have been highly successful in recent years. For instance, ResNets were used to win the ImageNet ILSVRC challenge where the goal was to classify images. A network with over 100 layers managed to achieved the best performance with the aid of this type of architecture [31][32]. Residual neural networks, that are pre-trained on the millions of images from the ImageNet database, are available to be used for other projects by adapting the parameters to the tasks at hand. This transfers the knowledge to the new domain. This can help in recognising patterns and reduces training time. Different pre-trained ResNet models were used in this thesis to potentially increase the performance [33].

## 4.5 MixUp Data Augmentation

Many patterns and correlations in the training data can only be found with a sufficient amount of training samples. However, in many scenarios training data can be limited and it might be hard to obtain larger data sets. As stated in Chapter 3.1, the data set used in this thesis is rather limited in size. Thus, we are faced with the task of training models with an insufficient amount of data. Data augmentation can be a valuable tool to increase the performance of any model discussed in this thesis. There are several approaches like using Generative Adversarial Networks (GAN) [34] or creating copies of the given data with slight alterations due to noise injection [35].

A rather simple approach is MixUp data augmentation where two data samples are combined to create a new one [36]. Every single value of the new sample is a weighted combination of the two original samples. For a regression problem, like the task at hand, a new label needs to be generated as well. This can be done by the exact same process, i.e.

$$\begin{aligned}\hat{\mathbf{X}} &= \gamma \mathbf{X}_i + (1 - \gamma) \mathbf{X}_j \\ \hat{y} &= \gamma y_i + (1 - \gamma) y_j\end{aligned}$$

where  $\mathbf{X}_i$  and  $\mathbf{X}_j$  are the randomly chosen ‘parent’ samples,  $y_i$  and  $y_j$  are the corresponding labels to the chosen samples and  $\gamma$  is the weight. Zhang, Cisse, Dauphin et al. [36] generally recommends using a Beta distribution for  $\gamma \sim \text{Beta}(\alpha, \beta)$  with  $\alpha, \beta \in [0.1, 0.4]$  to generate the weight. With  $\alpha$  and  $\beta$

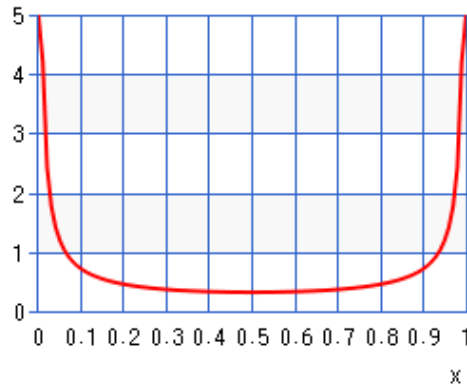


Figure 4.6: Beta distribution with  $\alpha = \beta = 0.2$ .

in this range, the weight is either close to 0 or 1, as depicted in Figure 4.6. Thus, it is unlikely for the generated new sample to be the average of both parent samples, but more likely a small deviation from one of them.

## 4.6 Gaussian Filter

A technique applied to increase the performance of the models are Gaussian filters applied to the input. They are used to smooth data and remove noise. It just leaves the overall trajectory of the sensor measurements and smooths the rapid changes between the individual heats. This might be useful when suspecting that the small changes in sensor measurements that occur at every heat might not be meaningful, but only the general changes in measurement over time. In Figure 4.7, a feature from one campaign of the

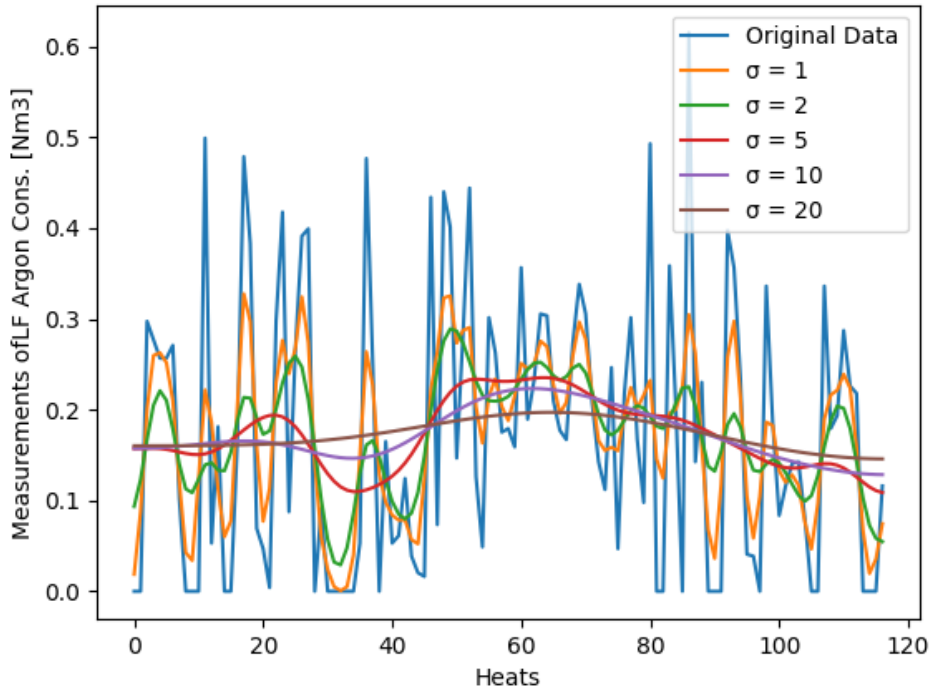


Figure 4.7: Feature is filtered with Gaussian with different  $\sigma$  values.

data set is visualized. Additionally, the filtered data is shown with different values for  $\sigma$ , which determines the smoothing strength of the filter kernel. The filter kernel is given by

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{(-\frac{x^2}{2\sigma^2})},$$

where a larger *sigma* introduces more smoothing to the result [37].

## 4.7 Feature Selection with SHAP Values

As shown in Chapter 3.1, the data sets consists of dozens of different sensor measurements. The large amount of features was provided without certain knowledge of which features actually have any correlation with the wear in refractory lining. Selecting which features genuinely impact the result is valuable information. Moreover, reducing the amount of features might improve the performance by removing noisy sensor measurements which are harmful for the performance. Feature selection [13] reduces overfitting and avoids making decisions based on fluctuation in random noise. Training on less data also reduces the training time and number of parameters that need to be trained in machine learning models.

There are several methods for feature selection. For instance, in a linear regression model, the coefficients give information about the correlation of features to the target. However, a neural network is mostly a black box where no correlation between input and output can be directly derived. Other feature selection approaches are, for instance, simple heuristics. McCombe [38] defines three major categories of feature selection.

- Wrapper methods which utilize a greedy search approach of finding the best features, where different subsets are tested and evaluated. When a feature improves the results, it is added to the subset.
- Filter methods that make use of i.e. the Pearson correlation to determine the usefulness of a feature while not considering the combinations and interactions of features.
- Embedded Methods, which is a combination of the two aforementioned approaches where the feature selection is part of the model tuning process.

One recent approach for feature selection are SHAP (SHapley Additive exPlanations) values [39], which aim to explain the impact of each input feature to the output. This approach is based on Shapley values from cooperative game theory, which attempts to answer the question of how valuable an individual in a cooperation is and subsequently what payout they should expect for their contribution. These values are calculated by taking all possible coalitions between players and calculating the difference the presence of the player in question makes to the output. The Shapley value of this selected player is the average of all the marginal distributions.

This approach can be translated to neural networks, players can be translated to input features or even pixel, thus the contribution of a measurement can be evaluated in how important it is to the output [40]. In Figure 4.8, it can be seen how certain pixels (in red) increase the likelihood of an image being from a certain class while others (in blue) decrease the probability. With this visual representation it is easy to see which parts of an image are important and which are not. In this case, the goal was a classification task. However,

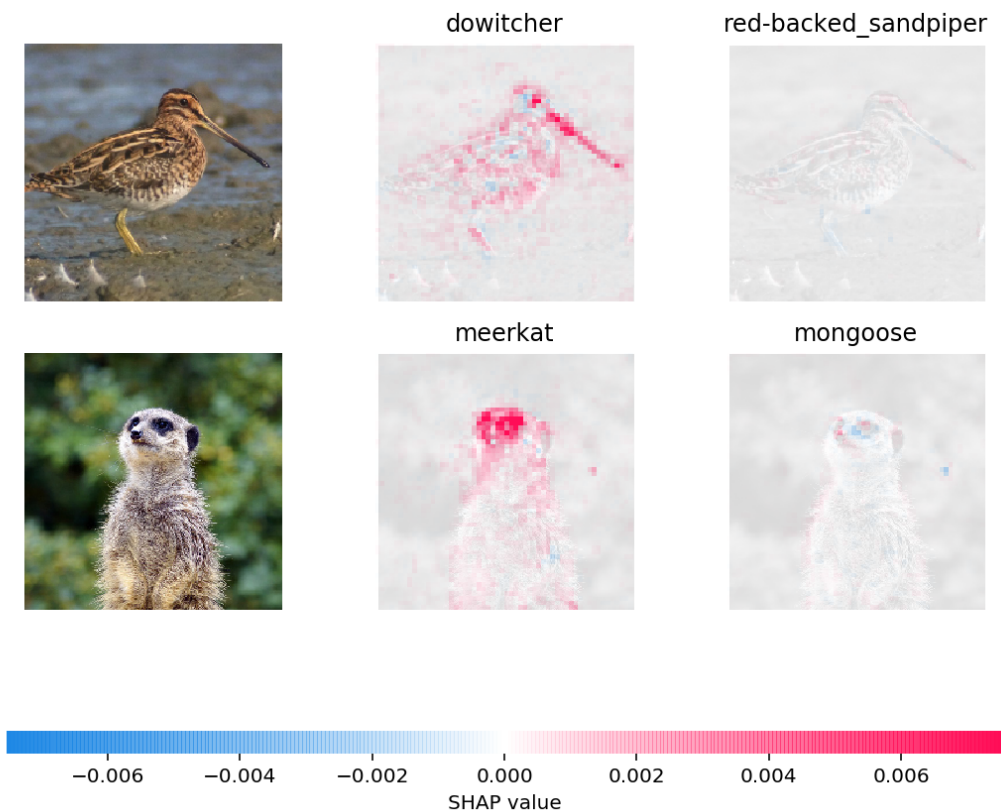


Figure 4.8: SHAP values for image classification (taken from: [41]).

the same can be done for the task at hand, which is a regression task. Since the data set consists of multiple sensor measurements taken over several heats, the 2D matrix of inputs can be interpreted and processed as an image. If a certain column of the matrix has high SHAP values, this feature is important to the prediction. In Figure 4.9, the SHAP values of some of the campaigns in the test set can be seen. Due to the lack of colored pixels the conclusion can be made that most of the features are not very influential to the results and are mostly noise. We suppose that the steel quality is responsible for the clusters of the SHAP values.



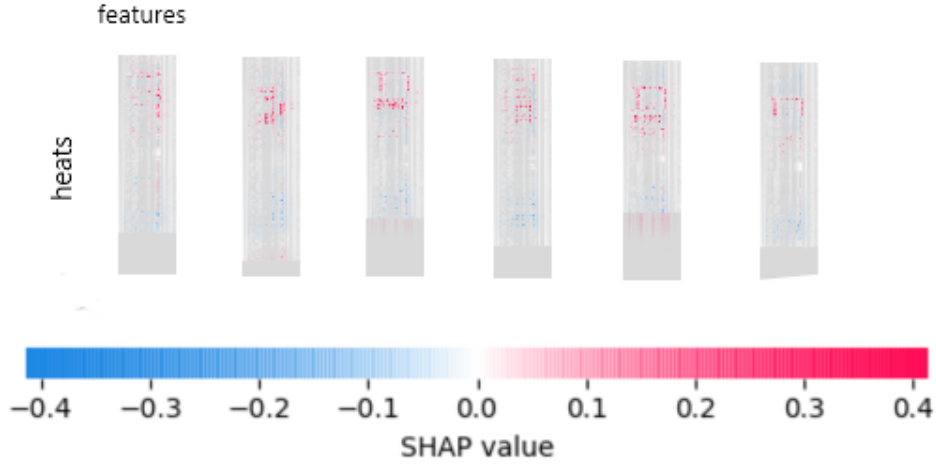


Figure 4.9: Visualization of the calculated SHAP values on some of the campaigns in the test set using the Metalzone data set.

In order to figure out which features are the most essential ones, multiple campaigns from the data set were evaluated. Each of the columns represents a certain type of sensor measurement, are aggregates over multiple campaigns of the test data set. For every single feature the following was calculated

$$\hat{S}_i = \frac{1}{C} \cdot \sum_{c=1}^C \left( \frac{1}{J_c} \cdot \sum_{j=1}^{J_c} (|s_{ijc}|) \right),$$

where  $\hat{S}_i$  is the aggregated SHAP value for feature  $i$ ,  $C$  is the number of campaigns in the test set,  $J_c$  is the amount of heats, in this specific campaign  $C$  and  $s_{ijc}$  is the SHAP value of a heat  $j$  in a campaign  $c$  from the test set and the specific feature  $i$ . It is important to note that the absolute value of the SHAP value is chosen since we are interested in the most impactful features. Thus, it does not matter whether the feature negatively or positively impacts the outcome. Additionally, this helps with aggregating the features since, the mean absolute value of a feature column is taken. Lastly, this value is averaged over all campaigns in the test set in order to finally get a representation of how influential a feature is. The most impactful features are used, while the others are discarded. This is the way feature selection was applied in this thesis. In Chapter 6, the best SHAP values for all data sets are shown. Additionally, in this chapter it is shown that feature selection does have a significant impact on the results.

## 4.8 Summary

Overall, many different approaches were used to predict the decay of a given campaign with all its sensor measurements. The focus of this thesis is on the usage of neural networks to tackle this problem. The basics of artificial neural networks have been explained. Furthermore, apart from fully-connected layers, other architectures were discussed. Convolutional neural Networks use convolutional kernels to attempt to find pattern in the input data. Next, recurrent neural networks, especially LSTMs and GRUs, were introduced. They aim to model sequential data to better predict the output. Residual neural networks, which add skip-connections to the architecture, and its pre-trained variants were also examined and introduced.

Due to the limitations of the number of samples in the data set explained in the previous chapter, MixUp data augmentation can be used to increase the size of the data set. Furthermore, a Gaussian filter might remove noise in the data that potentially causes difficulties for the machine learning algorithms. Lastly, feature selection with the help of SHAP values was discussed that might increase the model performance by removing unimportant sensor measurements.

# 5 Modelling Approaches

Plenty of approaches have been tested and compared with the goal of finding the best performing model. In this chapter, the training and model selection process are outlined which are used to find the best performing model.

## 5.1 Model Training Process

In this section, the training process of the model is introduced in order to reproduce the results. Firstly, preprocessing is performed. Some techniques improve the results, other decrease the model performance while others do not make any significant impact. Thus, their influence was tested in isolation and ultimately only preprocessing steps with a positive impact on the results were selected.

### 5.1.1 Preprocessing

The provided data contained some non-relevant information that needed to be removed. This includes semantic information like the campaign number or the heat ID. While they are important, a neural network does not need this kind of information since there are no important patterns to be learned. For instance, the campaign number is constant for each individual data point while the heat ID incrementally increases by one every row.

- **Feature selection:** Due to the vast number of sensor measurements made every heat, removing any further non-relevant features could vastly improve the results. Therefore, each feature is analyzed individually. The feature values and the standard deviation of each feature in a campaign are evaluated. If feature values are all exactly the same across all campaigns contained in the data set, the feature is removed. Afterwards the SHAP values are calculated of the remaining features and the most important features are kept while the rest is discarded. The number of sensor measurements that are kept is flexible. In Chapter 6, different selections for the number of features are compared in order to find the best configuration.
- **Feature normalization:** Further than that, the features were normalized between zero and one. This is performed to standardize the range of measurement in the context of the rest of the data set. For instance, a temperature of 100 degrees Celsius might be considered high in most

situations, but when normalized in the context of steel-production, is extremely low.

- **Gaussian filter:** After the features are selected and normalized, a Gaussian filter is applied to the remaining ones. This 1D Gaussian filter smooths the data to remove noise and only leaves the main trend of the sensor measurements. Different  $\sigma$  have been tested. A  $\sigma$  between 10-20 delivers the best results.
- **Zero padding:** Using neural networks, all the training data needs to be of the same dimensions. This is an issue, since every campaign consists of a different number of heats. Therefore, they need to have the same length. This is satisfied by first finding the maximum number of heats in any of the campaigns in the data set. Next, new rows are added to each campaign until all of them are at the size of the largest one. These rows are filled with zeros since in these added heats no measurements were made that contributed to the outcome. This is called zero-padding.
- **Data Augmentation:** The main issue with the data set is its limited size. As mentioned before, only around 100-200 campaigns were provided each. Since the amount of sensor measurements made during every heat is relatively high, much data is needed to correctly determine which features actually influence the refractory wear results significantly. Forrer [10] faced a very similar problem as this thesis and showed that the size of the data set has a vast impact on the results. Thus, different approaches were tested to increase the amount of data in the training set. We introduced MixUp data augmentation to increase the amount of training data in Chapter 4.5.
- **Cross Validation:** Moreover, the amount of data used to train the model is decreased when taking a considerable part as test set that is used to evaluate the trained model learned from the training data. Initially, five-fold cross validation was used, where four fifths are used to train the model while one fifth is used to evaluate the model. We noticed that the results significantly increased when using leave-one-out cross validation. Here, not five folds are created, but one for every single data point. This means, if, for instance, the data set consists of 100 campaigns, 99 campaigns are used for training and only the remaining campaign is used to evaluate the model. The disadvantage is that 100 separate models need to be trained, before the average performance can be determined. This takes a considerable amount of time. However, the size of the training set is substantially expanded,

which makes a substantial impact in this application. Hence, in this thesis, we prefer leave-one-out cross validation.

- **Target Scaling:** Until now, only the input data was refined. However, also the labels of the data, which is the measured RBL, have to be prepared. As already mentioned, the sensor measurement data is padded to the same size. However, when this is done important information is lost. The number of heats in a campaign has influence on the remaining refractory lining. Given a campaign with 100 heats and one with 150 heats with the same RBL after the respective campaign, the prior campaign experienced more wear across a heat on average than the latter one. If we now pad them to the same length, this vital information is lost. This is the reason zero padding was used in order to represent not-existing measurements. Other than that, the targets can be scaled down by the respective number of heats in their campaign. However, this did not improve the results and was therefore omitted.

### 5.1.2 Training

To train the model, a loss, an optimizer and the number of epochs need to be chosen.

- **Loss function:** The RMSE is used as a loss function. It is also used to evaluate the results after training. It is calculated by:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (p_i - y_i)^2}$$

where  $N$  is the number of data samples (i.e. campaigns),  $p_i$  is the predicted value for the given campaign while  $y_i$  is the expected target value. First, the residual, which is the difference between target and prediction, is squared. The mean of these squared residuals are calculated before the square root is taken [42].

- **Optimizer:** Adam was selected as an optimizer [19]. It is a state-of-the-art optimizer algorithm that improves the stochastic gradient descent approach by combining the advantages that AdaGrad and RMSProp have over stochastic gradient descent. The former includes a flexible learning rate for all parameters while the latter tries to speed up the training process by adding a decay factor to the rather slow AdaGrad approach.

- **Early Stopping:** When training a neural network the training data is used multiple times to train the model. Repetition is utilized to better learn patterns in the data. One pass-through of the data is called an epoch [43]. It needs to be specified how many epochs are done while training the model. This, however, is a far from trivial choice. On one hand, if the number of epochs is too high, the models learn small insignificant patterns in the data. This is called overfitting and leads to a high performance on the training data, but a much lower performance on the test set and in extension as well on any new data. On the other hand, when the number of epochs is too low, the model has not yet learned from the data which results in weak performances across the board.

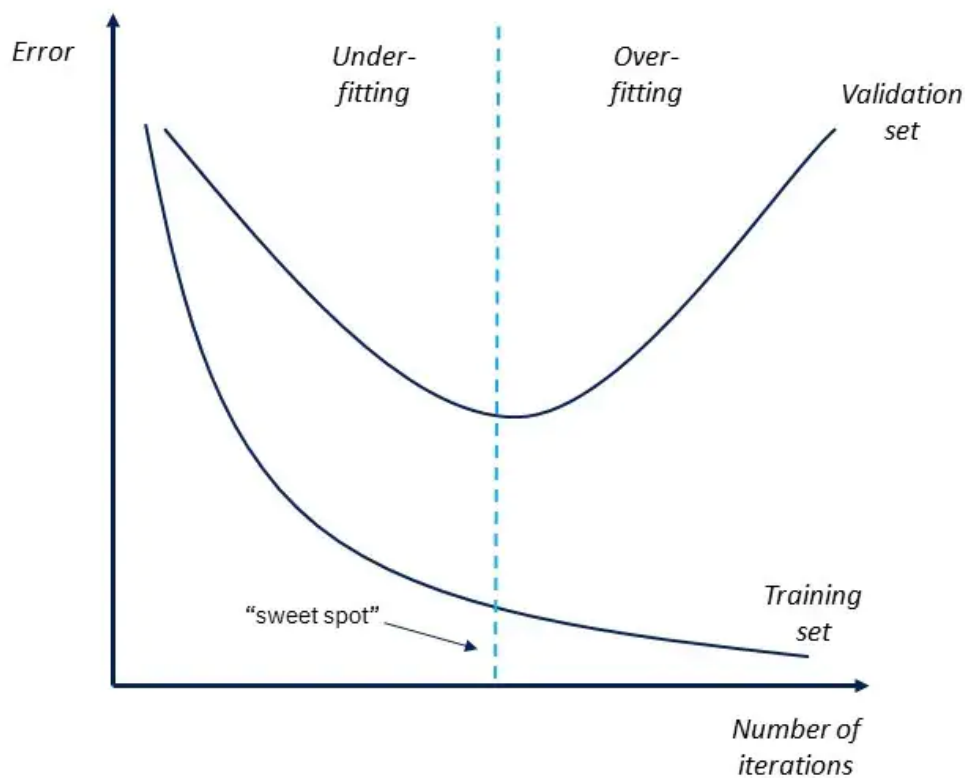


Figure 5.1: The bias-variance tradeoff - overfitting and underfitting (taken from: [44]).

This is commonly referred to as the bias-variance trade-off, where high bias means that the relevant relations in the data were missed and variance designate the sensitivity to minor fluctuations. As shown, in Figure 5.1 the

sweet spot is in the middle. This spot cannot be empirically determined. The method chosen to approach this point is *early stopping*. This is a technique that observes the training loss and stops the training after the loss does not improve after a certain number of epochs. This can also be done using a separate validation set. However, due to the limited number of samples in all data sets and the strong correlation between training set size and performance, this was omitted. This stops the model before learning any noise from the data and after most meaningful relation have been captured by the model. This means we do not overfit or underfit the data. In this application, the maximum number of epochs was chosen to be 250, but early stopping usually stopped the training process much earlier.

### 5.1.3 Evaluation

After the model is trained, there needs to be a way to evaluate and compare the results on new independent test data. This is why some data was reserved to test the model. When using cross validation multiple models are trained until every data sample of the test set was used. In the case of leave-one-out cross validation, only one campaign is used for tested and thus omitted from training. This means that the number of campaigns in the data set is equal to the number of models trained. After each model is trained, the remaining campaign is fed to the model and the resulting prediction is saved. After all the models are trained, the individual predictions are used to arrive at the final RMSE that represents the performance of the model on the test set.

## 5.2 Model Development Process

The centerpiece of this thesis is the process of finding the best model. There is no logical choice of model with the given type and amount of data. Should each heat be treated as a separate input or should each campaign be processed as a matrix? Should each measurement be treated like time-series data? These questions, among others, influence the architecture of the chosen model. Since it is not obvious how the data is optimally used, a multitude of different approaches are tested. This was done manually, by building different models and comparing them. In order to do this, the performance on the test set was assessed. If the performance on the training set is evaluated, there is the danger of selecting a model that over-fits the data and produces poor performances on any new data. Due to the low amount of samples available in the data set, it is possible that a certain model works best with part of the data, while not being able to capture meaningful

patterns when using another part of the data set. Thus, it is necessary to evaluate each model on all the data. This was done using cross validation, even though it substantially increases the training time. Therefore, five-fold cross validation was used in order for each data point being part of the test set once while keeping the training time manageable. This setting was employed while finding the best model. The final evaluation was done using leave-one-out cross validation, since it captures the models performance on new data more accurately.

### 5.2.1 Initial Models

At first, a randomly built model of multiple different architectures were constructed. This was needed to see which approach works best and should further be pursued. The exact number of models parameters are not important at this stage and will only be discussed when fine tuning the best model in Chapter 6. Here is a quick overview of the models tested at this stage.

- **2D CNN** (with  $\sim 200.000$  parameters) & **Deep 2D CNN** (with  $\sim 170.000$  parameters):

First, a 2-dimensional convolutional neural network was chosen. Each campaign was padded to the same length of heats. Therefore, the input is a 2 dimensional matrix where the rows denote the number of heats and the columns represent the sensor measurements (i.e. features). This model searches for patterns in this matrix that relate to the wear in the refractory lining. Here, it does not matter whether it is a pattern that occurs in a row or a column or even a combination. This architecture, should be able to detect whenever a certain measurement has an unusual progress over several heats as well as a certain heat where multiple measurements are unusual. After a 2D CNN layer, max pooling (discussed in Chapter 5.2.2) and finally a fully connected layer was used. It is important to note that in all layers of this model the ReLU function was used as an activation function. It is unclear how complex the patterns that correlate with the target value are. Consequently, a shallow and a deeper neural network was used. The deeper architecture can potentially capture more compound patterns in the data, but comes with the drawback of having more parameters that need to be trained. This is an issue due to the small sample sizes.

- **1D CNN** (with  $\sim 50.000$  parameters) & **Deep 1D CNN** (with  $\sim 30.000$  parameters):

When treating the heats as separate entities that amalgamate to the target value, it makes sense to test one dimensional convolutional



neural networks as well. Here, the kernels are applied to single rows, contrary to 2D CNNs where the kernel is a two-dimensional matrix. Again, a model with fewer layers was compared to a deeper alternative.

- **2D Conv ResNet** (with  $\sim 400.000$  parameters):  
Next, the concept of residual neural networks was tested. The first model, which is a 2D CNN, was changed to have some skip connections. Of course this can be done with any other model, so a random one was chosen to evaluate the difference.
- **Fully Connected ANN** (with  $\sim 15.000$  parameters) & **Deep Fully Connected ANN** (with  $\sim 20.000$  parameters):  
Moving away from convolutional neural networks, the sensor measurements of each heat can be treated as a separate input array. To test the effectiveness a fully connected model was tested, where the input was an array of sensor measurements. Again, a deeper model was also created.
- **LSTM RNN** (with  $\sim 90.000$  parameters) & **GRU RNN** (with  $\sim 65.000$  parameters):  
Lastly, recurrent neural networks were tested. For this a model using LSTM layer was created. The same was done with GRU layers instead of LSTM layers. Here, the sequentiality of the data is explored. Certain measurements taken at subsequent heats might have a connection to the RBL measured at the end.

The number of parameters of these models strongly depend on the number of features used in the input data. When a smaller number of sensor measurements are chosen during feature selection, the size of most models drastically decrease. Furthermore, the majority of parameters are needed when the data is flattened, for instance, after convolutional layers in order to combine the values to a single output scalar. Thus, the deeper CNN models require less parameters. The number of trainable parameters of the final models are significantly smaller. They are reported in Chapter 6. These unrefined models were all compiled, trained on the same data using five-fold cross validation and evaluated. Their results are shown in Figure 5.2 where the Metalzone data set was used. Multiple models performed satisfactorily, however, among further tests, the 2D CNNs turned out to be to best option for this individual data set. The fully connected models and recurrent neural networks are decent alternatives, but failed to significantly improve during the fine tuning of these models.

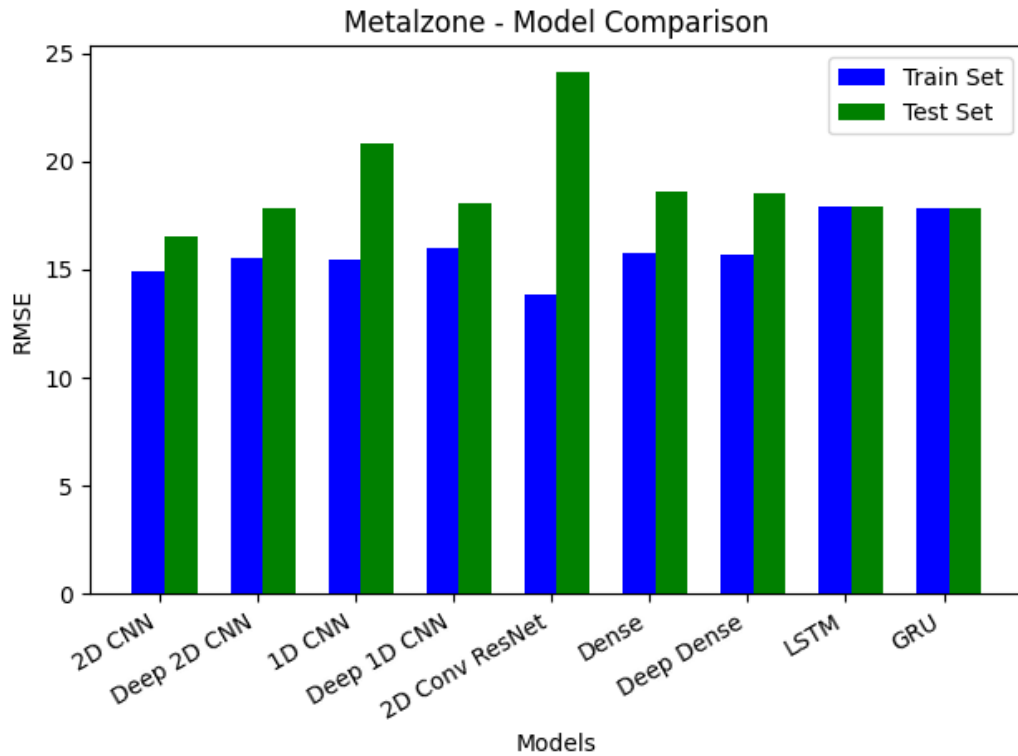


Figure 5.2: Initial comparison of different model architectures.

### 5.2.2 Fine Tuning

The most promising neural network type was chosen for each respective data set, before it was manually fine tuned. Multiple different alterations to the models were tested and evaluated. Again, each altered model was evaluated using the same data set and five-fold cross validation. This section will give an overview of all the different techniques tested with the aim of achieving the best results.

- **Dropout:** Dropout layers can be used to prevent overfitting and encourage generalization. These layers work by setting randomly chosen inputs of the layer to zero, thus eliminating it. This makes the model less dependent on certain features of the input and helps the model to recognise patterns with certain parts missing. Nevertheless, the performance did not increase when including dropout layers. Here, we assume that the limited amount of training data is problematic for the model to effectively generalize. However, for an application with more data, dropout layers can be indispensable to improve the performance on unseen data

- **Pooling:** Especially for convolutional neural networks, pooling layers can be useful. They try to summarize the input while reducing their size. This in turn reduces the number of parameters that needs to be learned and the number of computations that need to be performed. Pooling layers achieves this by taking, for instance, a 2x2 piece of the input and keeping only the maximum value in this grid. The other values are discarded. In this example, the height and width of the output is half of the input and the total number of output values is quartered. Instead of taking the maximum, the average as well as the minimum is also viable and was tested [45]
- **Batch Normalization:** Batch normalization was also investigated. It normalizes the input data of each layer. The data gets usually rescaled and recentered after the activation function. Thus, the model can train faster and is more stable [46].

Above, different additional layers of the models were discussed. However, the layers used in the initial models can also be modified and configured to achieve a better performance.

- **Regularizers:** First, regularizers were examined. They are utilized to better generalize to new data by applying a penalty to the loss to limit high-valued weights. Models can over-fit when trained too long. This method helps to avoid this. The L1 regularization, or Lasso regularization, and the L2 regularization, or Ridge regularization, were applied. The prior makes use of the absolute value of the coefficient to limit the model and thus often leads to a sparse model since weights often become zero. Contrary to this, the latter applies the square of the magnitude of the weights which does not lead to sparse models [47]. Keras was used in the implementation which offers three different places where these regularizers can be used [48]. Kernel regularizer penalizes the kernel weights. Bias regularizers control the bias on a layer. A bias is another learned weight that is able to shift the activation function in either direction. Each neuron has a single bias value for all preceding neurons [49]. Activity regularizers regularize the output of the chosen layer to keep the values from exploding. Every combination of regularizers has been explored and evaluated.
- **Weight initializers:** Another method examined are different weight initializers. Here, Keras allows the custom initialization of kernel and bias weights. Weight initializers can help the model to converge faster and also alleviate the possibility of gradient to vanish or explode [50]. Ones, zeros, random values and normally distributed values have been

tested. Regardless of the parameters of these distributions, the results hardly changed. Thus, moving forward the default values of zeros for the initial bias values and a normal distribution ( $\sigma = 0.01$ ) for the kernel weights were kept [51].

- **Model Depth:** The model size is one of the most important aspects to choose. In the prior chapter, deep and shallow models were tested. Nevertheless, in order to achieve the best performance the exact number of layers was manually found by compiling different model, before training and evaluating them. If a model has too little layers, the pattern in the input data cannot be captured fully. On the contrary, if a model is too deep, more parameters need to be learned and more computations have to be performed. Given, the small amount of data, a deep model likely struggles [52].
- **Number of filters/units and kernel size per layer** - Further, the number of units of a fully connected or recurrent layer and the number of filters and the kernel size of convolutional neural networks were altered and tested. This is, again, a rather difficult task where the correct configuration can only be empirically chosen. In order to achieve this, multiple models with different archetypes were tested and compared.
- **Skip connections & pretrained ResNets** - Lastly, the goal is to fine tune the residual neural network models. Different skip connections were used in different architectures. There is also the option of using pre-trained ResNets. For instance, there are models pretrained on the ImageNet data set [53]. Their weights are pretrained in order to identify patterns in images. If the patterns in the given data set behave similarly, this might be very helpful, especially due to the limited number of samples in the data set.

### 5.2.3 Process of Finding the Best Models

Fine tuning of the models by all the methods mentioned in Chapter 5.2.2 showed that most alterations did not significantly improve the results. Bias regularizers, kernel regularizers and activity regularizers hardly effected the results in most tests. The same can be said for different kinds of initializers. Batch normalization also did not have a consistent positive impact on the result. The aforementioned default values for initializers work well enough, so they do not need to be changed. However, when more data becomes available regularizers and batch normalization might become more useful.

Other configurations, on the contrary, actively hurt the performance of

the models. As already mentioned, dropout layers did not improve the results. While helpful to encourage generalization and prevent overfitting, the amount of samples are simply too low for this approach to be effective. Also, pretrained ResNets did not improve the performance. They were trained on image data. Thus, they are able to effectively recognise many patterns usually found in image processing. The data in this thesis contains different patterns. Additionally, the pre-trained layers actively hurt the model since it adds a substantial level of complexity.

Looking at the pre-processing steps, MixUp Data Augmentation and scaling the target values did not improve the results. On the contrary, the Gaussian Filter and feature selection on the basis of SHAP values greatly aided the models performance.

What actually made a noticeable impact on the performance was the selection of the amount of layers. Furthermore, the number of units or filters per layer also made a difference. The kernel size of the latter also substantially changed the results. These are the essential building blocks of a model and therefore have a greater impact than additional modifications that might solve problems like overfitting. Here, many different architectures worked well. Now, the goal is to find the best model for our data sets.

**Metalzone/Slagzone:** For the Metalzone/Slagzone data set, 2D CNNs work the best. However, different architectures perform similarly. For instance, on one hand, a 2D CNN with one convolutional layer, one Max Pooling layer and one fully connected layer works well if a kernel size of 3x3 is used, but performs worse with a kernel size of 6x6. On the other hand, a 2D CNN using four convolutional and max pooling layers and 2 fully connected layers works best with a kernel size of 6x6, but struggles with a kernel size of 3x3. Thus, different combinations need to be tested.

After every single modification mentioned, like regularizers, the model was evaluated, every configuration that decreased the performance was omitted. The same was performed for different model depths and layer sizes. Consequently, a model for each reasonable combination was compiled and compared. This rather large list was pruned by worse performing combinations. An example of this can be seen in Figure 5.3.

To test several models in order to find the best performing model, a pipeline was build using Python and the library Keras, that enabled the automatic training and comparison of these models.

**RH-degassser:** For the Bottom Inlet/Bottom Outlet/Heart data sets, the best models found for the Metalzone/Slagzone sets did not work. Therefore, this process determining the best configuration of hyper-parameters was

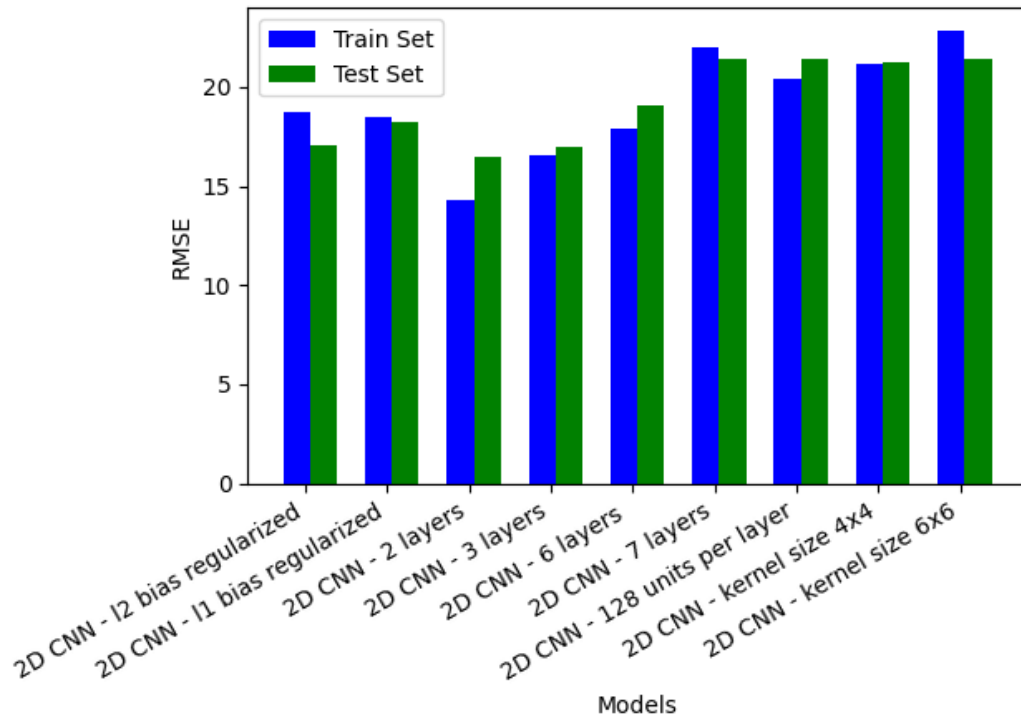


Figure 5.3: Different model architectures are trained and compared using the Metalzone set.

performed for this data set as well. The exact models that worked for both data sets are discussed in the next chapter.

### 5.3 Summary

In this chapter, the entire training and model selection pipeline was introduced. Pre-processing is done and the model is trained before they can be evaluated and compared. The model selection process was done manually. Different additional layers like dropout, polling, or batch normalization layers were tested. Alterations to layer, specifically weight regularizes and initializers, were examined. Additionally, the number of layers, kernel sizes, number of filters and units were assessed. Lastly, the change in performance when utilizing skip connections or pretrained ResNets have been evaluated. The best performing settings were combined and evaluated in order to find the optimal composition and the best model. This was done separately for the Metalzone/Slagzone and Bottom Inlet/Bottom Outlet/Heart data sets, since completely different architectures lead to the best performances.

## 6 Results

In this chapter, the best models, that were found during the model refinement process presented in the last chapter, are introduced in more detail. These models are then further optimized by determining the optimal number of features to use by using SHAP values. Lastly, the performances of these models presented are showcased, discussed and compared.

### 6.1 Metalzone and Slagzone Data

As already mentioned in Chapter 5.2.3, 2 dimensional convolutional neural networks achieve the best results for this data set. Different amounts of filters per convolutional layer perform well i.e. 128, 64 and 32. However, in order to keep the model small, lower number of filters were preferred. The same approach was applied to the number of neurons of fully connected layers.

Finally, after all the optimization and pruning, two different models produce the best results for this data set. The first one is a deep 2D CNN with as seen in Figure 6.1. First, there are four convolutional layers with 32 filters and a kernel size of  $3 \times 3$ . They use zero padding to prevent shrinking the input and to keep its size stable. Between these layers, 2D max pooling layers with a pool size of  $2 \times 2$  are interwoven. Lastly, the data is flattened into a single vector and fed into two fully connected layers with 32 and 16 units, respectively, before the output is calculated. The discussed techniques in Chapter 5.2.2, i.e data augmentation, batch normalization, dropout layers, regularizers and weight initializers do not significantly improve the model performance.

The second model is a shallower version of the prior one. As pictured in Figure 6.2, the model consists of only one convolutional layer and one fully connected layer. 32 filters with a kernel size of only  $2 \times 2$  were used. The fully connected layer has also only 16 units. Compared to the prior model, this one is significantly smaller. Even though these architectures are fairly different, both deliver promising results. Out of all models that were tested, these were two of the most light-weight ones. Due to the efficiency of convolutional layers, the number of parameters the model needs to learn is relatively low, especially, since the dimensionality of the input is greatly reduced through the pooling layers before the fully connected layers are reached. Both models have between 50.000 and 100.000 trainable parameters depending on the number of features used.

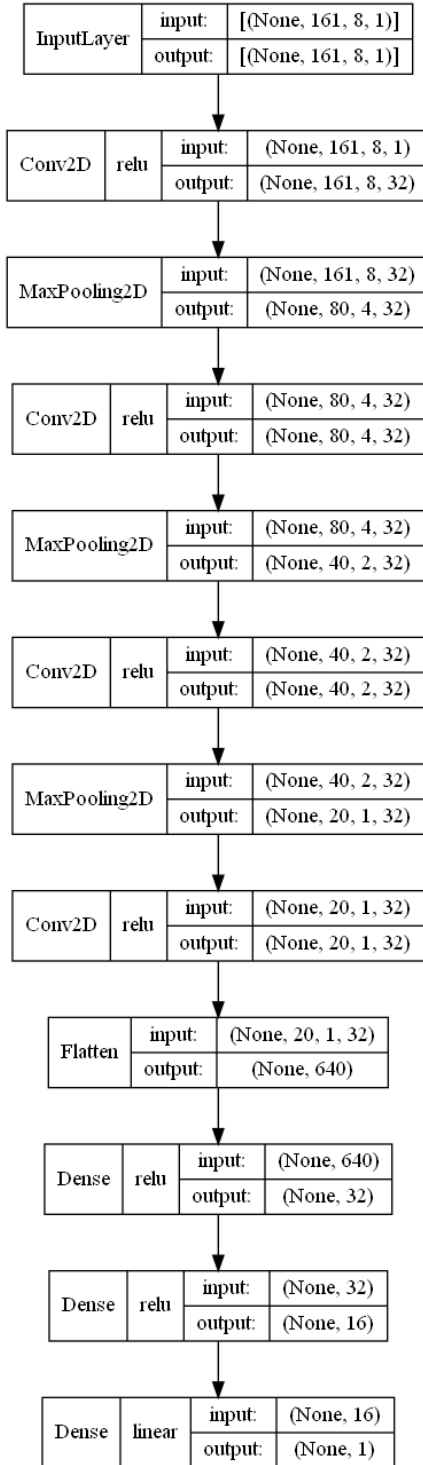


Figure 6.1: Architecture of the 2D CNN with 6 layers.

Due to the low number of samples in each data set these models likely perform better since model with a larger number of parameters need more training data. As mentioned in Chapter 4.7, SHAP values can be used to select the most important sensor measurements (i.e features) of the data set. However, with the manual model selection this creates a chicken and egg problem. Should the most important features be selected first while the optimal model is found based on the feature selection? Or should the best working model be determined first? The latter approach was chosen for several reasons. Firstly, SHAP values are calculated after a model is trained, before the influence of each feature is evaluated. For this process a model is needed.

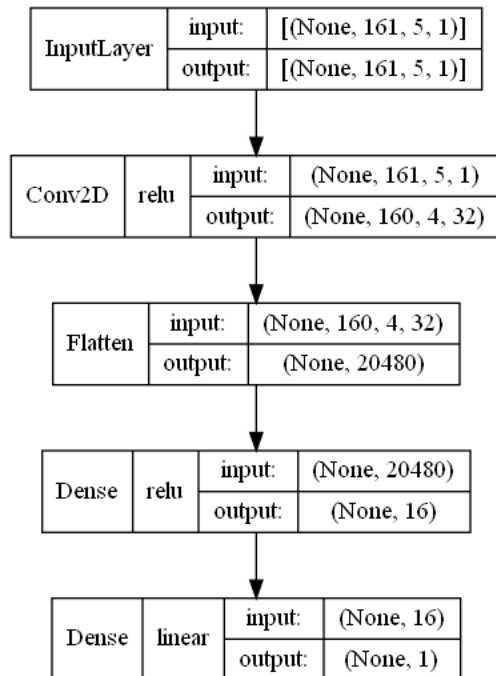


Figure 6.2: Architecture of the 2D CNN with 2 layers.



Thus, it is preferential to use the model that is found beforehand. Second, it is difficult to determine the optimal number of features to use since different models might work best with different numbers of selected features. Therefore, the optimal model should be found first, before each number of features is tested to find the best result. Thus, with the best models for the full data set already established, feature selection is performed. Here, the deeper convolutional neural network is problematic. Each of the convolutional layers reduce the size of the input unless padding is applied. The 2x2 max pooling layer halves the size of the input. This model can handle the original input data since it consisted of around 60 measurements made in more than 100 heats. However, when only using, for instance, 5 measurements, the model encounters problems. It is not possible to use a kernel when one of the dimensions of the data is smaller than the kernel size. Therefore, for lower choices in the number of measurements, this model cannot be used. The second model does not suffer from this problem as it only requires a size of 2 values in either dimension.

Both models were tested with a different number of features. The SHAP values were similar for both models. In Table 6.1 and Table 6.2, the 15 most impactful features can be seen. They mostly consist of measurements of chemical components made at the end of the heat.

#	Feature	SHAP val.	#	Feature	SHAP val.
1	LF End CaO [%]	0.0114	1	LF End Al [%]	0.0444
2	LF End SiO <sub>2</sub> [%]	0.0093	2	Tap+LF Al [kg]	0.0309
3	LF End Mn [%]	0.0084	3	Tap+LF SPL [kg]	0.0225
4	LF End Al [%]	0.0079	4	LF End SiO <sub>2</sub> [%]	0.0191
5	LF Spar [kg]	0.0075	5	LF End Si [%]	0.0172
6	LF End N [ppm]	0.0063	6	Tap+LF Lime [kg]	0.0169
7	Tap+LF Lime [kg]	0.0061	7	Tap+LF Dolo [kg]	0.0144
8	LF End Si [%]	0.0051	8	LF End MgO [%]	0.0138
9	LF Start temp. [C]	0.0050	9	LF End CaO [%]	0.0131
10	Tap+LF Dolo [kg]	0.0047	10	LF Power [kWh]	0.0117
11	LF End MgO [%]	0.0046	11	LF End N [ppm]	0.0104
12	Tap+LF Spar [kg]	0.0046	12	LF End Mn [%]	0.0085
13	LF End Al <sub>2</sub> O <sub>3</sub> [%]	0.0044	13	LF Ca wire [kg]	0.0072
14	Tap Weight [t]	0.0042	14	LF Dolo [kg]	0.0065
15	LF P ON Time [min]	0.0040	15	LF Lime [kg]	0.0062

Table 6.1: Top 15 Mean Absolute SHAP Values - Metalzone.

Table 6.2: Top 15 Mean Absolute SHAP Values - Slagzone.

It is obvious that most measurements barely have an impact on the result as the SHAP values are relatively low. Most of them can likely be omitted, especially the ones not included in the aforementioned tables. Thus, a low number of features is likely to be more effective. In Figure 6.3 the performances of both models with different number of features on the Metalzone data can be seen. For reference, the RMSE for both models when using all features is around 19mm. Thus, feature selection is directly responsible for a huge improvement in prediction quality. It can be seen, that

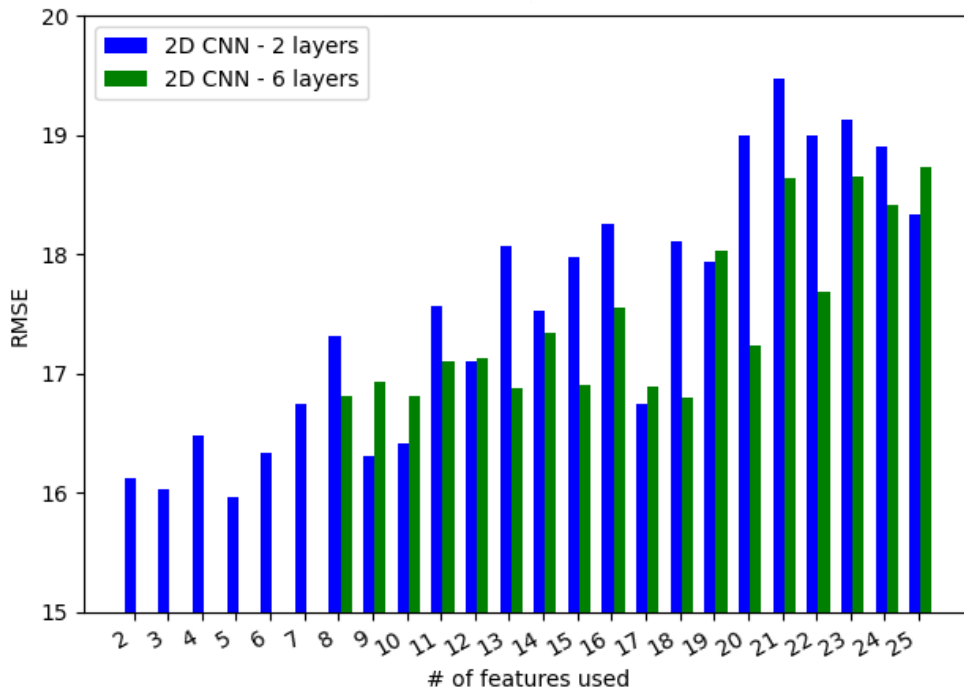


Figure 6.3: Comparison of model performances using different amount of features using the Metalzone data set.

lower numbers of features correlate with a better performance. However, if too many relevant features are omitted, the performance suffers. The same is the case when too many are included. Using approximately the top 5 features seems to be the sweet spot to achieve the best results. As already mentioned, due to the composition of the first models architecture, it is not possible to use less than 8 features since the kernel size is larger than one of the input dimensions, especially after the Pooling layers. Thus, only 8 or more features can be used with this architecture. Contrary to this, the smaller model can be used with even only two features. This allows it to use only the most important features and thus is able to produce the best results.

This finally leads us to the best performing configuration. As mentioned in earlier chapters, the amount of training data has a direct influence on the results. Thus, leave-one-out Cross Validation is used to maximize the amount of data for training. This was done for the final evaluation.

For the Metalzone data set an RMSE value of 15.70mm was achieved with the small model using 5 features. In Figure 6.4, the residuals between the target RBL values and the predictions can be seen. Most predictions are promisingly close to their target. Even worse predictions are not too far of considering that the RBL at the end of an campaign is roughly around 100mm.

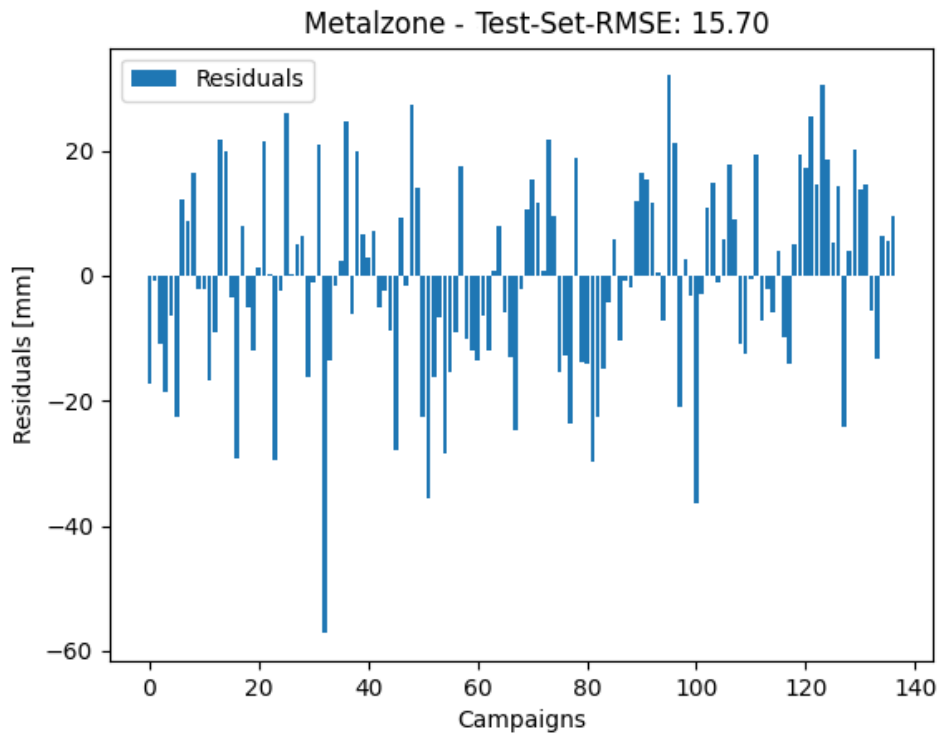


Figure 6.4: Residual plot of the results of the Metalzone data set using 5 features and a two layer 2D CNN.

For the Slagzone data set, the same methods produced the best results. Just as with the model described above, five features were used to achieve the top performance. Using the same 2 layer CNN an RMSE of 22.45mm was calculated. This result, specifically the obtained residuals, are visualized in Figure 6.5

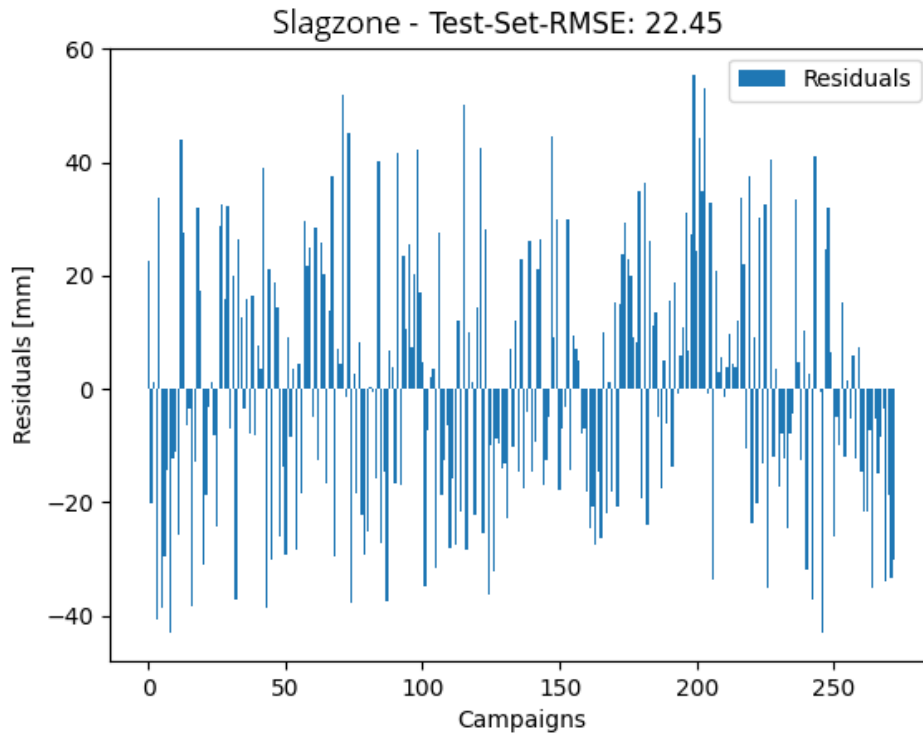


Figure 6.5: Residual plot of the results of the Slagzone data set using 5 features and a two layer 2D CNN.

For comparison, using all features in the data set leads to a performance of around 25mm. Overall, the performance of this set is fairly remarkable. Such a low RMSE for both sets shows that predicting the RBL is a viable alternative to manually measuring the remaining refractory lining.

## 6.2 Heart, Bottom Inlet, Bottom Outlet Data

Finding a model for this data set proved to be quite a challenge. The two dimensional convolutional neural network that produced the best results for the other data sets does not work at all for this data. It seems to be much harder for any neural network to capture meaningful relations between these measurements and the RBL. Nonetheless, the same process, that was introduced in Chapter 5.2, was applied here with the hope of finding the best alternative to the 2D CNN. The performances of all tested models were drastically worse when compared to Metalzone or Slagzone data. Recurrent neural networks seemed to perform better than the other approaches. A higher number of layers improved the performance slightly. However, the addition of a skip connection helped with balancing the drawbacks of a deeper model. After some fine tuning, the following model depicted in Figure 6.6 was used. It consists of GRU layers with 32 units each and a skip connection. In total, the architecture includes 5 GRU layers, 1 skip connection and a single fully connected layer before the output to calculated. Similar to models discussed in the previous chapter, this architecture is also rather light-weight and has around 30.000 trainable parameters. The exact number of parameters depends on the amount of features used.

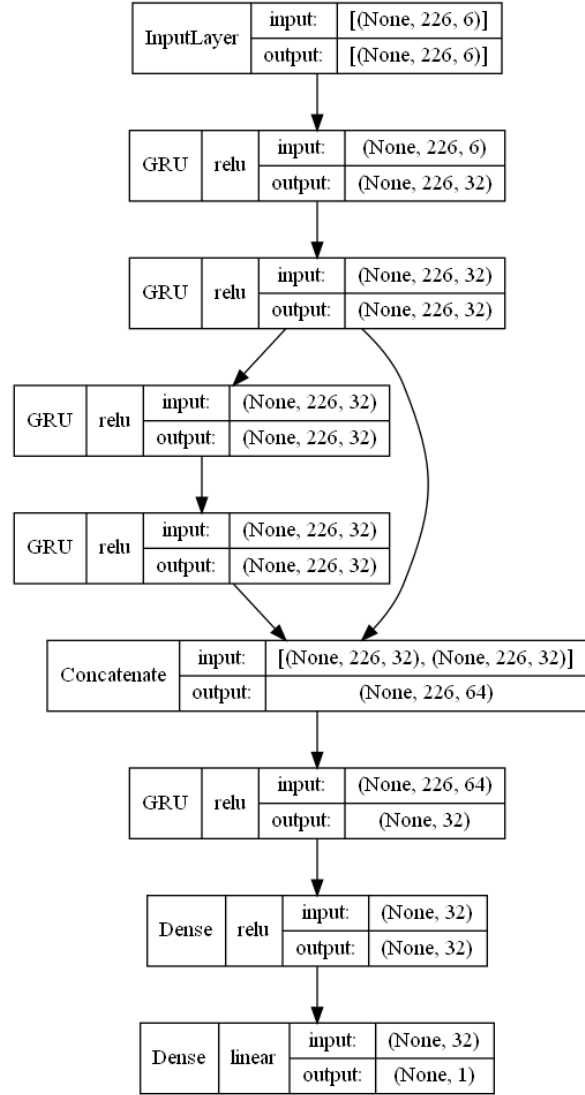


Figure 6.6: Architecture of the GRU model.

To potentially improve the results of this model, feature selection on the basis of SHAP values was performed. In Table 6.3 the 15 features with the most impact are ranked for the Bottom Inlet data. The same can be seen in Table 6.4 for the Bottom Outlet data set and in Table 6.5 for the Heart data set. When comparing these rankings, it becomes apparent that the same sensor measurements seem to be the most important, independent of the data set in question.

#	Feature	Shap val.
1	Start RH-Temp [C°]	0.0557
2	1.Pfa-Temp nach Abst. [C°]	0.0496
3	Ende RH-Temp [C°]	0.0410
4	O vor RH [ppm]	0.0105
5	Entschwefelung Kalk [kg]	0.0093
6	Abst.Zugabe Kalk [kg]	0.0074
7	O vor Al an RH [ppm]	0.0057
8	BR Erdgas Mng. vor BBeg [m3]	0.0052
9	O2 Menge vor Behand. [m3]	0.0038
10	Entschwefelung Tonerde [kg]	0.0022
11	Entschwefelung Turbokalk [kg]	0.0018
12	Entschwefelung CaC2 [kg]	0.0016
13	TR max. SP-Menge [m3/h]	0.0013
14	Abst.Zugabe Schlackeng. [kg]	0.0011
15	Abst.Zugabe Rohmagnesit [kg]	0.0008

Table 6.3: Top 15 Mean Absolute Shap Values  
- Bottom Inlet.

#	Feature	Shap val.
1	Ende RH-Temp [C°]	0.0736
2	1.Pfa-Temp nach Abst. [C°]	0.0662
3	Start RH-Temp [C°]	0.0613
4	Abst.Zugabe Kalk [kg]	0.0117
5	O vor RH [ppm]	0.0105
6	Entschwefelung Kalk [kg]	0.0103
7	O2 Menge vor Behand. [m3]	0.0070
8	O vor Al an RH [ppm]	0.0062
9	BR Erdgas Mng. vor BBeg [m3]	0.0044
10	Entschwefelung Tonerde [kg]	0.0029
11	Entschwefelung Turbokalk [kg]	0.0029
12	Entschwefelung CaC2 [kg]	0.0023
13	TR max. SP-Menge [m3/h]	0.0021
14	Abst.Zugabe Schlackeng. [kg]	0.0019
15	Abst.Zugabe Rohmagnesit [kg]	0.0011

Table 6.4: Top 15 Mean Absolute Shap Values  
- Bottom Outlet.

#	Feature	Shap val.
1	Start RH-Temp [C°]	0.0302
2	Ende RH-Temp [C°]	0.0298
3	1.Pfa-Temp nach Abst. [C°]	0.0268
4	O vor RH [ppm]	0.0145
5	O vor Al an RH [ppm]	0.0110
6	Entschwefelung Kalk [kg]	0.0086
7	Abst.Zugabe Kalk [kg]	0.0047
8	Entschwefelung Turbokalk [kg]	0.0025
9	BR Erdgas Mng. vor BBeg [m3]	0.0021
10	Entschwefelung Tonerde [kg]	0.0018
11	Entschwefelung CaC2 [kg]	0.0018
12	O2 Menge vor Behand. [m3]	0.0015
13	TR max. SP-Menge [m3/h]	0.0010
14	Abst.Zugabe Kohle [kg]	0.0009
15	Abst.Zugabe Schlackeng. [kg]	0.0008

- Heart.

This points towards the conclusion that the feature ranking based on SHAP values are not just random fluctuation in data, but actually represent that these sensor measurements are the most influential ones with regard to the remaining refractory lining. In Figure 6.7, the comparison of the impact of the number of selected features is shown. The Bottom Inlet set was used to demonstrate the difference. Again, it can be noticed that a smaller number of features increases the performance. The best RMSE values are achieved when using between four and eight of the most important features. Finally, we select the top six sensor measurements. For comparison, when using all features in these data set the performances are around 45mm to 50mm.

Overall, the results achieved for this data set are as follows. With the Bottom

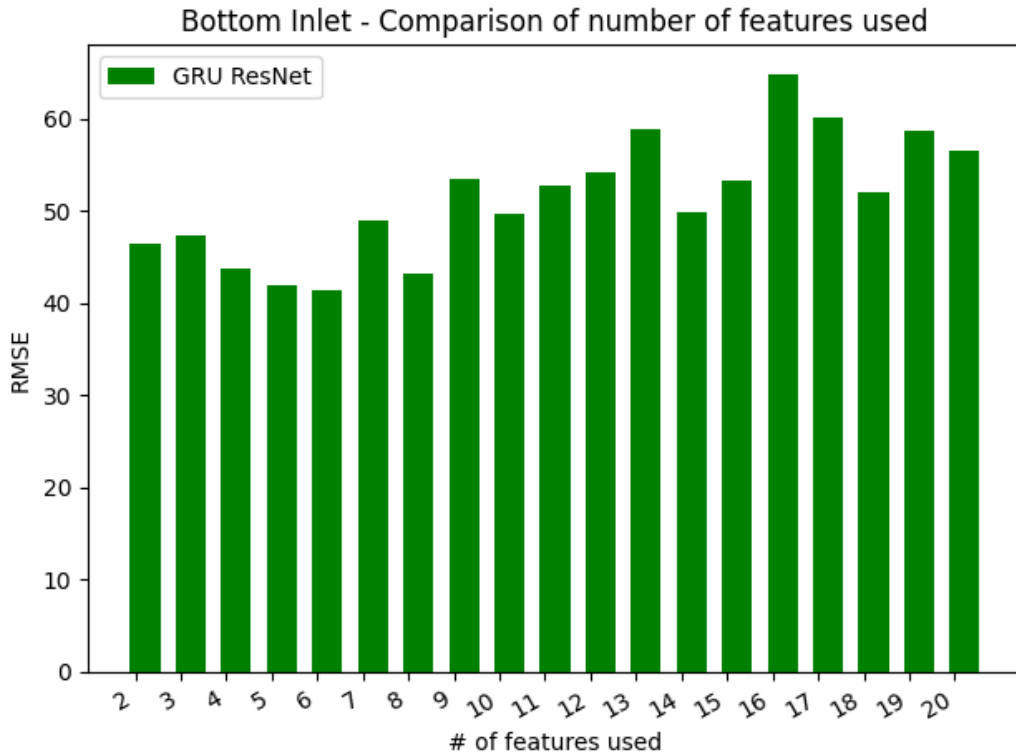


Figure 6.7: Comparison of model performances using different amount of features - Bottom Inlet data set.

Inlet data set an RMSE of 38.51mm was obtained. The residuals can be seen in Figure 6.8. Bottom Outlet achieved an RMSE of 39.72mm, while Heart reached 43.58mm. This is visualized in Figure 6.9 and Figure 6.10, respectively. Inspecting the residuals show that some predictions greatly differ from their respective targets. Nevertheless, the results for this data set are inferior to the ones achieved in Chapter 6.1. One explanation is that the steel processing of the RH degasser which produces the Bottom Inlet/Bottom Outlet/ Heart data is much more aggressive for the refractory lining than the steel treatment in the vessel. Furthermore, Chapter 6.3 provides results of other approaches which are better for this data set.

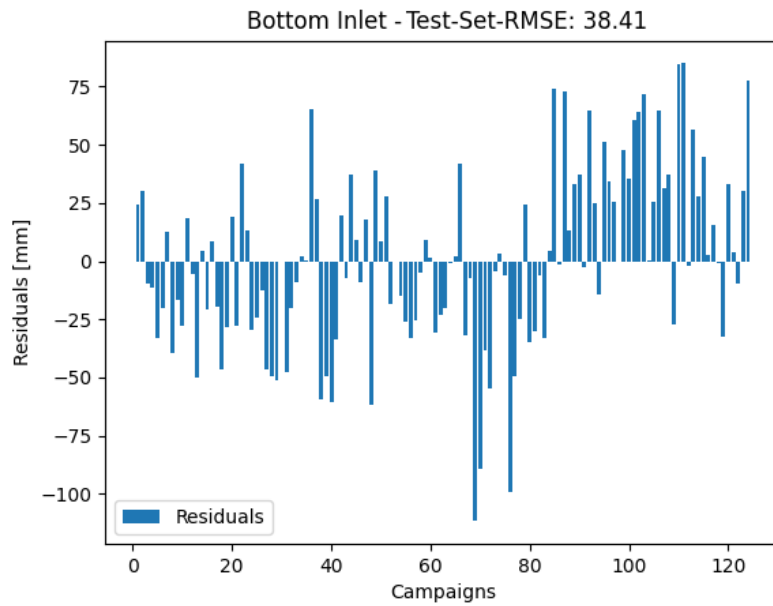


Figure 6.8: Residual plot of the results of the Bottom Inlet data set using 6 features and a GRU ResNet.

This leaves the conclusion that, contrary to the Metalzone and Slagzone set, neural networks might not be to most fitting method of finding the relationship between the sensor measurements and the remaining brick length in this data set.

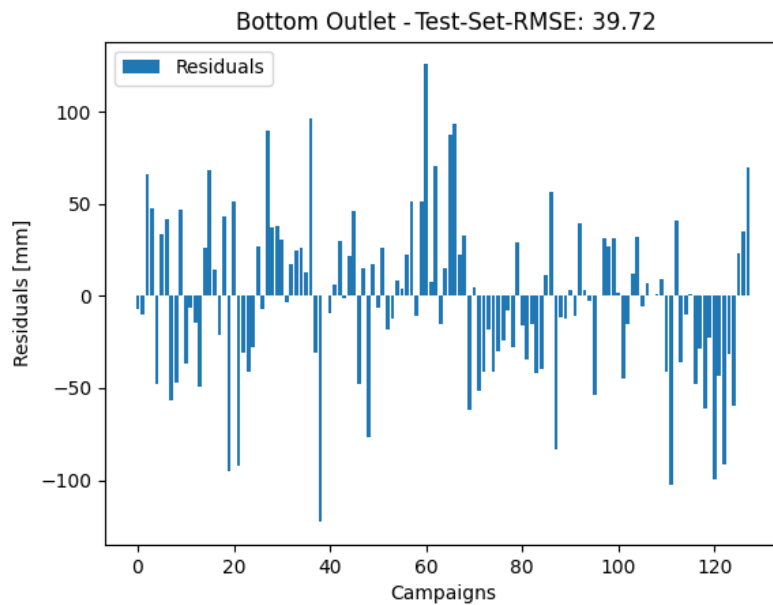


Figure 6.9: Residual plot of the results of the Bottom Outlet data set using 6 features and a GRU ResNet.





Figure 6.10: Residual plot of the results of the Heart data set using 6 features and a GRU ResNet.

### 6.3 Comparison

Mutsam and Pernkopf [2] used the same data set as in this thesis. However, they used a different methodology and thus arrived at other results. In this Section, these results will be compared to shown how valuable the results of this thesis are. The results in Table 6.6 are from an unpublished report where other approaches are compared to the iterative least squares (ILS) approach described in [2]. The methods used in their work will not

Dataset	Ra. For.	SVM	XGBoost	DNN	ILS	Avg. Wear	zD CNN	GRU ResNet
Metalzone	17.75	18.75	<b>16.74</b>	23.66	16.91	24.35	15.70	-
Slagzone	<b>22.52</b>	24.51	22.79	27.37	24.67	35.94	22.45	-
Heart	39.46	40.53	41.16	42.78	<b>36.42</b>	41.09	-	38.41
Bottom Inlet	33.14	33.59	34.33	36.88	<b>23.54</b>	39.00	-	39.72
Bottom Outlet	36.76	37.43	37.09	37.22	<b>28.23</b>	36.98	-	43.58

Table 6.6: Comparison of different approaches on all data sets.

be explored in detail, but it is important to note that the ILS achieve by far the best performance on the data set from the RH degasser which includes the Bottom Inlet, Bottom Outlet and Heart data sets. Nevertheless, other approaches like Random Forests, Support Vector Machines, XGBoost and deep neural networks were tested and worked better for the Metalzone and Slagzone data sets, which stem from a different application as detailed in

Chapter 3.

The average wear simply serves as sanity check for all approaches. Here, the average wear for all heats is calculated and used. This is done by normalizing the RBL with the number of heats in each respective campaign. If a method produces worse results than the average wear model, it is unable to effectively learn any connection between the input data and the target.

Even though the results displayed in Table 6.6 include neural networks, the models in this thesis achieve significantly better performances compared to these approaches due to the rigours amount of fine tuning done to find the best model, as detailed in Chapter 5.2.2. The results achieved in this thesis can be seen in the side of the table.

On one hand, the 2D CNN manages to produce better results for Metalzone and Slagzone compared to the Iterative Least Squares approach as well as compared to the other approaches.

On the other hand, the models tested for the second data set (Bottom Inlet, Bottom Outlet and Heart) could not reach an adequate performance. When looking at the results, it becomes apparent that it is significantly more difficult to achieve decent results with this data set as every approach delivers much higher RMSE values compared to the Metalzone/Slagzone data set. Even though a myriad of different architectures were tested, the results did not match the ones achieve with the ILS approach, which is likely the more suited method. However, a prediction error of around 40mm, after a campaign usually concluded with a remaining brick length in excess of 100mm, is still usable.

Overall, it is evident that the approach found in this thesis produces superior results for the Metalzone and Slagzone by using a two-dimensional CNN. However, the ILS approach seems to be more promising for the data set of the RH degasser.

## 6.4 Summary

In this chapter, the results of all data sets were presented. The models found with the process outlined in the previous chapters were further optimized by using feature selection based on calculating the SHAP values. It was shown that this significantly increased the performance. The final RMSE values for each part of these data sets was calculated using leave-one-out cross validation and resulted in respectable results for the Metalzone/Slagzone data set. However, in comparison with other approaches the results achieved when using the Bottom Inlet, Bottom Outlet and Heart data set were inferior.

## 7 Conclusion

In this thesis, the wear of the refractory lining of metallurgical vessels and processing machinery used in the steel industry is estimated. Since this is usually done with expensive laser measurements, the goal of this thesis is to use neural networks in order to predict the remaining brick length. The lining can be used several times during so-called heats (i.e. treatments) before needing to be replaced. This entire process is referred to as a campaign. At the end of a campaign the remaining brick length is measured. This serves as the target for this estimation problem. During each heat, several sensor measurements are made. These include duration, temperatures and other attributes. These measurements are the input to the model and are used as the basis to predict the RBL. The challenge of this data set is that every heat includes several sensor measurements, while each campaign has only a single target value.

The approaches evaluated during this work are based on several different kinds of neural networks. Convolutional neural networks, residuals neural networks, recurrent neural networks and other architectures were tested. Since it was unclear, which architecture would be the most effective in predicting this single value from dozens of different measurements, such a variety of models was assessed. Additionally, several different pre-processing steps were utilized to possibly improve the performance of the models. What proved to be most effective was a Gaussian filter that was applied to the input data as well as features selection on the basis of SHAP values. These values represent how impactful an input is to the model output. They were aggregated across all heats and campaigns and subsequently ranked. Different numbers of the most impactful features were used and their resulting performances compared. Finally, the best model was found manually by training multiple models with different architectures and modifications before evaluating and comparing them.

A two-dimensional CNN proved to be the best option for the Metalzone and Slagzone data set. Here, an RMSE of 15.70 and 22.45 was achieved, respectively. The results improved upon the ones achieved by other techniques. However, the data set for the RH degasser proved to be incompatible with the approaches tested in this thesis. Here, the results do not match the performances reached when using other methods. Here, an RMSE of 38.41 for the Bottom Inlet data set, an RMSE of 39.72 for the Bottom Inlet data set and an RMSE of 43.58 for the Heart data set were obtained. Nevertheless, this thesis shows that predicting the refractory wear in a metallurgical process with the usage of neural networks is a viable alternative.



## 8 Outlook and Future Work

Promising results were achieved by using the approaches outlined in this thesis. Other works [2] used different approaches that also provided respectable outcomes. Even though this thesis covered a myriad of different approaches and pre-processing methods, there are still plenty of other approaches that could provide better results. Further than that, a combination of different techniques and approaches might be the way to achieve the best performance.

The core issue with this regression problem is that the remaining RBL value is the result of the wear of a several dozen heats combined. Therefore, it is exceptionally difficult to learn the impact of a measurement made in one of these heats. This problem is reinforced by the little amount of data that is available. A model that estimates the wear of a single heat would be much more effective in capturing the impact of a single measurement. Such a model was actually built in the process of this thesis. However, the performance was not on par with the reported models. Even if the results were better, it would be rather unreliable. This is because, the only possible target for a single heat is the average wear of a campaign. Since such a campaign usually contains more than 100 heats, this average is not particularly meaningful, since any significantly different heat would most likely be averaged out.

In order to combat this problem, each heat would need a reliable measurement of how much wear occurred during the process to effectively learn the influence of each feature in the data set. The refractory lining is typically assessed every heat using laser measurements. If it is possible that these wear measurements are provided without too much noise a solid model could be built. This should be able to make better use of the individual measurements and predict the refractory wear each heat causes. Naturally, such a model would likely be significantly better at estimating the remaining brick length at the end of a campaign.

Another approach would be to classify the campaigns instead of predicting the exact RBL after several heats. If the campaigns are split into, for instance, high, low and medium wear, a classification model could be built. Of course this would not provide exact RBL predictions, but it might be easier to train. Further than that, if these classifications are precise, the campaign could be stopped whenever too many heats with predicted high wear occurred.

These are multiple different options that can be explored to further improve the prediction results. Due to the complexity of the problem, it is unclear

whether these approaches would likely perform better or worse. However, having refractory wear measurements after every heat should help in building better models. These measurements might be expensive, but would only be needed during the collection of training data. If the performance is as expected, the prediction of the created model might be able to remove the need for laser measurements.

# Bibliography

- [1] K. Väinämö and J. Röning, *An artificial neural network-based rotation correction method for 3d-measurement using a single perspective view*, 1997.
- [2] N. Mutsam and F. Pernkopf, *Regression with partially hidden targets for wear modeling*, 2022.
- [3] I. T. Administration, *Global steel trade monitor*, Sep. 2022. [Online]. Available: <https://www.trade.gov/data-visualization/global-steel-trade-monitor>.
- [4] A. Sharma and D. Singh, "Evolution of industrial revolutions: A review," *International Journal of Innovative Technology and Exploring Engineering*, pages 66–73, Sep. 2020.
- [5] S. Vaidya, P. Ambad and S. Bhosle, "Industry 4.0 – a glimpse," *Procedia Manufacturing*, pages 233–238, 2018, 2nd International Conference on Materials, Manufacturing and Design Engineering (iCMMD2017), 11-12 December 2017, MIT Aurangabad, Maharashtra, INDIA, ISSN: 2351-9789. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2351978918300672>.
- [6] M. Ghobakhloo, "The future of manufacturing industry: A strategic roadmap toward industry 4.0," *Journal of Manufacturing Technology Management*, number 6, pages 910–936, january 2018, ISSN: 1741-038X. [Online]. Available: <https://doi.org/10.1108/JMTM-02-2018-0057>.
- [7] B. Schmult, "Evolution of the hopewell furnace blast machinery," *IA. The Journal of the Society for Industrial Archeology*, number 2, pages 5–22, 2016, ISSN: 01601040, 23277858. [Online]. Available: <https://www.jstor.org/stable/26643088>.
- [8] *Basic oxygen steelmaking simulation*, 2021. [Online]. Available: [https://content.steeluniversity.org/simulators/sc13/bos/help/BOS\\_User\\_Guide\\_EN\\_ver02.pdf](https://content.steeluniversity.org/simulators/sc13/bos/help/BOS_User_Guide_EN_ver02.pdf).
- [9] M. Varga, "High temperature abrasion in sinter plants and their cost efficient wear protection," English, embargoed until null, Ph.D. dissertation, 2016.
- [10] M. Forrer, *Prediction of refractory wearwith machine learning methods*, 2012.

- [11] M. Shah, V. Vakharia, R. Chaudhari, J. Vora, D. Y. Pimenov and K. Giasin, "Tool wear prediction in face milling of stainless steel using singular generative adversarial network and lstm deep learning models," *The International Journal of Advanced Manufacturing Technology*, **number 1**, **pages** 723–736, Jul. 2022, ISSN: 1433-3015. [Online]. Available: <https://doi.org/10.1007/s00170-022-09356-0>.
- [12] "Refractory condition monitoring and lifetime prognosis for rh degasser," English, *in 2019 AISTech Conference Proceedings AISTech 2019 Iron and Steel Technology Conference and Exposition ; Conference date: 06-05-2019 Through 09-05-2019*, **january** 2019, **pages** 1081–1089.
- [13] P. Leray and P. Gallinari, "Feature selection with neural networks," *Behaviormetrika*, **number 1**, **pages** 145–166, **january** 1999.
- [14] J. Brownlee, *A gentle introduction to the rectified linear unit (relu)*, **august** 2020. [Online]. Available: <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>.
- [15] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," *in From Natural to Artificial Neural Computation* J. Mira and F. Sandoval, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, **pages** 195–201, ISBN: 978-3-540-49288-7.
- [16] R. Dastres and M. Soori, "Artificial neural network systems," *International Journal of Imaging and Robotics*, **pages** 13–25, **march** 2021.
- [17] E. Grossi and M. Buscema, "Introduction to artificial neural networks," *European journal of gastroenterology hepatology*, **pages** 1046–54, **january** 2008.
- [18] I. Santos, M. Castro Pena, N. Rodriguez-Fernandez, A. Torrente-Patiño and A. Carballal, "Artificial neural networks and deep learning in the visual arts: A review," *Neural Computing and Applications*, **pages** 1–37, **january** 2021.
- [19] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. [Online]. Available: <https://arxiv.org/abs/1412.6980>.
- [20] Y. Dauphin, H. Vries, J. Chung and Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization," *arXiv*, **february** 2015.



- [21] S. Kadam, *Neural networks part 3: Understanding back propagation and learning rates*, Jun. 2020. [Online]. Available: <https://medium.com/analytics-vidhya/neural-networks-part-3-understanding-back-propagation-learning-rates-3482a981a2f0>.
- [22] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. [Online]. Available: <https://arxiv.org/abs/1511.08458>.
- [23] T. Trnovszky, P. Kamencay, R. Orješek, M. Benco and P. Sykora, "Animal recognition system based on convolutional neural network," *Advances in Electrical and Electronic Engineering*, **october** 2017.
- [24] S. A. Alanazi, M. M. Kamruzzaman, M. N. Islam Sarker *et al.*, "Boosting breast cancer detection using convolutional neural network," *J Healthc Eng*, **page** 528622, **april** 2021.
- [25] IBM, *What are convolutional neural networks?* **october** 2020. [Online]. Available: <https://www.ibm.com/cloud/learn/convolutional-neural-networks>.
- [26] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **pages** 107–116, **april** 1998.
- [27] I. C. Education, *What are recurrent neural networks?* Sep. 2020. [Online]. Available: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>.
- [28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, **pages** 1735–80, **december** 1997.
- [29] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, 2014. [Online]. Available: <https://arxiv.org/abs/1412.3555>.
- [30] K. Cho, B. van Merriënboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, 2014. arXiv: 1409.1259. [Online]. Available: <http://arxiv.org/abs/1409.1259>.
- [31] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, 2015. [Online]. Available: <https://arxiv.org/abs/1512.03385>.
- [32] O. Russakovsky, J. Deng, H. Su *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, **number** 3, **pages** 211–252, 2015.

- [33] Keras, *Keras documentation: Keras applications*, 2022. [Online]. Available: <https://keras.io/api/applications/>.
- [34] F. H. K. d. S. Tanaka **and** C. Aranha, *Data augmentation using gans*, 2019. [Online]. Available: <https://arxiv.org/abs/1904.09135>.
- [35] C. Chadebec **and** S. Allasonnière, *Data augmentation with variational autoencoders and manifold sampling*, 2021. [Online]. Available: <https://arxiv.org/abs/2103.13751>.
- [36] H. Zhang, M. Cisse, Y. N. Dauphin **and** D. Lopez-Paz, *Mixup: Beyond empirical risk minimization*, 2017. [Online]. Available: <https://arxiv.org/abs/1710.09412>.
- [37] R. Fisher, S. Perkins, A. Walker **and** E. Wolfart, *Gaussian smoothing*, 2003. [Online]. Available: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm>.
- [38] M. McCombe, *Intro to feature selection methods for data science*, Jun. 2019. [Online]. Available: <https://towardsdatascience.com/intro-to-feature-selection-methods-for-data-science-4cae2178a00a>.
- [39] L. S. Shapley, "17. a value for n-person games," in *Contributions to the Theory of Games (AM-28), Volume II* H. W. Kuhn **and** A. W. Tucker, Eds. Princeton: Princeton University Press, 2016, **pages** 307–318. [Online]. Available: <https://doi.org/10.1515/9781400881970-018>.
- [40] S. Lundberg **and** S.-I. Lee, *A unified approach to interpreting model predictions*, 2017. [Online]. Available: <https://arxiv.org/abs/1705.07874>.
- [41] S. Lundberg, *Slundberg/shap: A game theoretic approach to explain the output of any machine learning model*. Jun. 2022. [Online]. Available: <https://github.com/slundberg/shap>.
- [42] A. G. Barnston, "Correspondence among the correlation, rmse, and heidke forecast verification measures; refinement of the heidke score," *Weather and Forecasting*, **number** 4, **pages** 699–709, 1992. [Online]. Available: [https://journals.ametsoc.org/view/journals/wefo/7/4/1520-0434\\_1992\\_007\\_0699\\_catcra\\_2\\_0\\_co\\_2.xml](https://journals.ametsoc.org/view/journals/wefo/7/4/1520-0434_1992_007_0699_catcra_2_0_co_2.xml).
- [43] A. Sarangam, *Epoch in machine learning: A simple introduction (2021)*, 2021. [Online]. Available: <https://www.jigsawacademy.com/blogs/ai-ml/epoch-in-machine-learning#:~:text=An%5C%20epoch%5C%20in%5C%20machine%5C%20learning,learning%5C%20process%5C%20of%5C%20the%5C%20algorithm..>
- [44] I. C. Education, *What is overfitting?* **march** 2021. [Online]. Available: <https://www.ibm.com/cloud/learn/overfitting>.

- [45] V. Suárez-Paniagu **and** I. Segura-Bedmar, "Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction," *BMC Bioinformatics*, **number 8**, **page** 209, Jun. 2018.
- [46] S. Ioffe **and** C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. [Online]. Available: <https://arxiv.org/abs/1502.03167>.
- [47] O. Demir-Kavuk, M. Kamada, T. Akutsu **and** E.-W. Knapp, "Prediction using step-wise l1, L2 regularization and feature selection for small data sets with large number of features," en, *BMC Bioinformatics*, **number 1**, **page** 412, **october** 2011.
- [48] Keras, *Keras documentation: Layer weight regularizers*, 2020. [Online]. Available: <https://keras.io/api/layers/regularizers/>.
- [49] F. Malik, *Neural networks bias and weights*, **march** 2021. [Online]. Available: <https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da>.
- [50] W. Boulila, M. Driss, M. Al-Sarem, F. Saeed **and** M. Krichen, *Weight initialization techniques for deep learning algorithms in remote sensing: Recent trends and future perspectives*, 2021. [Online]. Available: <https://arxiv.org/abs/2102.07004>.
- [51] K. Team, *Keras documentation: Conv2d layer*, 2022. [Online]. Available: [https://keras.io/api/layers/convolution\\_layers/convolution2d/](https://keras.io/api/layers/convolution_layers/convolution2d/).
- [52] A. Schindler, T. Lidy **and** A. Rauber, "Comparing shallow versus deep neural network architectures for automatic music genre classification," **november** 2016.
- [53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li **and** L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," **in** *2009 IEEE Conference on Computer Vision and Pattern Recognition* 2009, **pages** 248–255.

# Acronyms

<b>AdaGrad</b>	Adaptive Gradient Algorithm
<b>ANN</b>	Artificial Neural Network
<b>BOF</b>	Basic Oxygen Furnace Process
<b>CCD</b>	Charge-Coupled Device
<b>CNN</b>	Convolutional Neural Network
<b>DNN</b>	Deep Neural Network
<b>EAF</b>	Electric Arc Furnace
<b>GAN</b>	Generative Adversarial Networks
<b>GRU</b>	Gated Recurrent Unit
<b>ILS</b>	Iterative Least Squares
<b>ILSVRC</b>	ImageNet Large Scale Visual Recognition Challenge
<b>LSTM</b>	Long Short-Term Memory
<b>RBL</b>	Remaining Brick Length
<b>ReLU</b>	Rectified Linear Unit
<b>ResNet</b>	Residual Neural Network
<b>RH</b>	Ruhrstahl-Heraeus
<b>RMSE</b>	Root Mean Squared Error
<b>RMSEProp</b>	Root Mean Squared Propagation
<b>RNN</b>	Recurrent Neural Network
<b>SHAP</b>	Shapley Additive Explanations