Digital Twin of Torpedo Ladle Cars

MS53035

Jasmijn van Arnhem

Delft University of Technology

TU Delft

Digital Twin of Torpedo Ladle Cars

Student Name	Student number
Jasmijn van Arnhem	4648544

Instructor: Sid Kumar and Maria Gabriela Garcia Campos
Faculty: Faculty of Mechanical, Maritime and Materials Engineering, Delft

TU Delft

Abstract

Torpedo ladle cars play a role in transporting hot metal within steel plants, and optimizing their operation is crucial for reducing energy consumption and CO_2 emissions.This study investigates the feasibility of developing a digital twin for simulating the thermal management of torpedo ladle cars at Tata Steel by combining reduced-order models with machine learning techniques. Four distinct digital twin models were developed and evaluated: Random forest combined with singular value decomposition (SVD) (RF-SVDmodel), neural network combined with SVD (NN-SVDmodel), neural network without SVD (NN-model), and recurrent neural network without SVD (RNN-model). These models generalization abilities were evaluated using a learning curve or K-fold cross validation. The models accuracy was tested using validation datasets of a full cycle prediction of the torpedo ladle car process. The RMSE and R^2 of the validation prediction of each model were obtained and compared. Results show that RF-SVDmodel, NN-SVDmodel and RNN-model exhibit some challenges such as potential overfitting and inconsistency in performance. The NN-model is the most promising option with robust performance, high generalization abilities, and competitive accuracy to the AnsysTwinbuilder.

Contents

Abstract i

1 Introduction 1

2 State of the art 3
2.1 Torpedo ladle cars . . . 3
2.2 Finite element model . . . 4
2.3 Overview of Reduced Order Models . . . 5
2.3.1 Singular value decomposition . . . 5
2.3.2 Proper orthogonal decomposition . . . 7
2.3.3 Dynamic mode decomposition . . . 9
2.3.4 Principal component analysis . . . 10
2.4 Overview of Physics-based Machine Learning methods . . . 11
2.4.1 Artificial Neural Networks . . . 12
2.4.2 Decision tree regression model . . . 12
2.4.3 k-nearest neighbors regression model . . . 13
2.5 Digital twin applications . . . 13
2.6 Conclusion . . . 15
2.6.1 Research plan . . . 16

3 Methodology 17
3.1 Torpedo Ladle Car . . . 17
3.1.1 Physical description . . . 17
3.1.2 Finite element model . . . 19
3.2 Outlining of prediction model . . . 20
3.3 Data visualization and preprocessing . . . 22
3.4 Ansys Twinbuilder . . . 23
3.5 Model development . . . 24
3.5.1 Model evaluation and criteria . . . 25

4 Model 27
4.1 Singular value decomposition . . . 27
4.2 Random forest model (RF-SVDmodel) . . . 28
4.3 Neural network models . . . 30

5 Results and discussion 32
5.1 Model Generalization Metrics . . . 32
5.1.1 Random forest model (RF-SVDmodel) . . . 32
5.1.2 Neural network models, NN-svdmodel, NN-model and RNN-model . . . 33
5.2 Performance Evaluation on test and Validation Datasets . . . 35
5.2.1 RF-SVDmodel and NN-SVDmodel . . . 35
5.2.2 NN-model and RNN-model . . . 38
5.3 Prediction Performance Evaluation and Comparison of Models . . . 42
5.3.1 Random forest model (RF-SVDmodel) . . . 42
5.3.2 Neural network model with SVD (NN-SVDmodel) . . . 43
5.3.3 Neural network model (NN-model) . . . 45
5.3.4 Recurrent neural network model (RNN-model) . . . 46
5.3.5 Ansys Twinbuilder . . . 47

6 Conclusion 49
6.1 Future research . . . 50

References **51**

A Appendix **55**
A.1 Interpolation data . . . 55
A.2 R^2 . . . 56
A.3 Root Mean Square Error (RMSE) . . . 56
A.4 Learning curve . . . 57
A.5 k-Fold Cross-validation . . . 59
A.6 Recurrent Neural Network (RNN) . . . 60

B Appendix **61**
B.1 Singular Value Decomposition of CM and WL models . . . 61
B.2 K-fold Cross Validation of CM HM and WL . . . 62
B.3 Learning Curves of the WL and CM models for NN-SVDmodel and NN-model . . . 64
B.4 Test Evaluation of the WL and CM models for NN-SVDmodel and NN-model . . . 66
B.5 Learning Curves of the WL and CM models for RNN-model . . . 68
B.6 Test Evaluation of the WL and CM models for RF-SVDmodel . . . 70

C Appendix **71**
C.1 RF-Model . . . 71
C.2 NN-SVDmodel . . . 73
C.3 NN-model . . . 75
C.4 RNN-model . . . 77

D Appendix **80**
D.1 Tabs 1-3: Output Plots with Input Values . . . 80
D.2 Tabs 4 and 5: Input Distribution . . . 81
D.3 Tab 6: Temperature Change Analysis . . . 81

1 Introduction

Tata Steel manufactured around 30.15 million tonnes of crude steel in 2019 [1]. For most steelmaking processes, the primary energy source is coal because steelmaking is highly energy-intensive. Therefore, the steel industry emits large quantities of CO_2. CO_2 is a primary contributor to global warming. Therefore, it is essential to reduce the quantities of CO_2 emitted by the steel industry [2].

Steel production is a complex and energy-intensive process. One critical stage where significant energy losses occur is during hot metal transport in torpedo ladle cars. The hot metal loses heat due to radiation and convection to the environment. When the hot metal temperature decreases too much, it can negatively impact steelmaking. For instance, additional reheating steps or a reduction in the amount of mixed scraps added to the hot metal may be required.

During this production stage, various torpedo ladle cars are available for transportation, each with different features. Examples of these features include the initial temperature of the refractory inside the torpedo ladle cars, the weight of the hot metal inside, the temperature of the hot metal, or the lining age, which influences the heat loss of the hot metal inside. Simulating the torpedo car's thermal state and the hot metal's temperature during operation can provide insights into choosing the torpedo ladle car that results in the minimum amount of heat loss in the hot metal.[3]

Currently, a finite element model is used to simulate the thermal state of the torpedo ladle car at Tata Steel. However, the finite element model cannot efficiently simulate the thermal states of the torpedo ladle cars; it takes several hours and is therefore not applicable to real-time decision-making in the plant. Nevertheless, developing a digital twin of the torpedo car that can simulate the thermal state of the torpedo ladle in real time would be beneficial for minimizing heat loss in the hot metal during torpedo ladle car operations. Tata Steel currently utilizes Ansys Twinbuilder as a digital twin to simulate the thermal state of the torpedo ladle car. This is achieved by using data obtained from a finite element model. Key features influencing the thermal state, such as the initial refractory temperature, lining age, etc., are used to generate datasets of the thermal states of the torpedo ladle car for various scenarios. These datasets serve as the training data for supervised machine-learning techniques within the Ansys Twinbuilder model. Additionally, data reduction techniques are employed to enhance the efficiency of Ansys Twinbuilder.

While Ansys Twinbuilder offers a solution for digital twin development, we can overcome the limitations of Ansys Twinbuilder software by developing a digital twin ourselves. These limitations include restricted access to detailed model information, a black box approach that hinders customization, and limitations in flexibility and adaptability. Additionally, a self-developed solution would eliminate dependence on external support and streamline integration with existing systems.

This research explores alternative methods for creating digital twins of torpedo car operation, focusing on developing reduced order models and employing machine learning techniques without relying on ANSYS Twinbuilder software. Initially, a literature review is conducted, with an overview of digital twin technologies' current state of the art. Following, a chapter on the methodology behind the research,

where the physical description and the finite element model of the Torpedo car are explained. Furthermore, this chapter contains information on the Ansys twinbuilder, outlining the prediction model, data visualization and preprocessing, and model development. The next chapter concerns the singular value decomposition, the different models' architecture and hyper-parameters. Then, the results of the model's performance are shown and discussed, followed by the study's conclusion and some future research recommendations.

2

State of the art

2.1. Torpedo ladle cars

The torpedo ladle car has a capacity of 400 tons. The name of the torpedo ladle car comes from its shape; the torpedo-shaped ladle of the car, shown in figure 2.1, has a cylindrical barrel-shaped central body with a conical end. The torpedo ladle is built in a way that makes the mouth very small, so the heat loss is minimal. After the ladle is filled with hot metal, a synthetic slag can create a slag-metal reaction, which even helps remove more impurities and reduce heat loss [4].

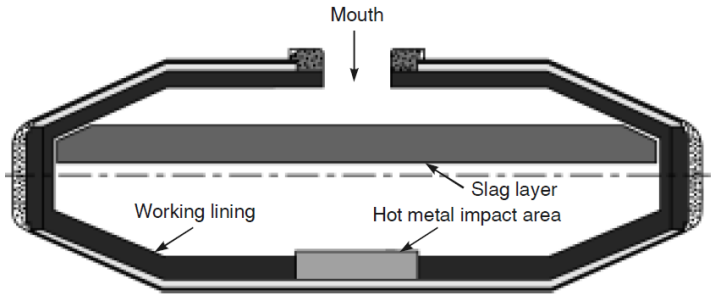


Figure 2.1: Schematic drawing of the torpedo ladle shape.[4]

Process

A torpedo ladle car transports hot metal from the blast furnace to the essential oxygen plant. The blast furnace produces hot metal continuously, but the basic oxygen furnace can only produce steel in batches. This results in the hot metal being stored in the torpedo car until it can be used in the primary oxygen plant. Furthermore, the torpedo car has a buffering function that removes impurities from the hot metal before they are processed in the primary oxygen plant. A schematic drawing of the torpedo ladle car operation is given in figure 2.2. The number of cycles of this operation determines the age of the torpedo ladle car. The age of the torpedo ladle car is essential because the refractory thickness inside the torpedo decreases with age, increasing the volume of hot metal. This will affect the heat losses and determine the torpedo's life. The total age of a torpedo car is around 1600 cycles, but every 400 cycles, a repair will be done and considered in the model as a new torpedo again. [3].

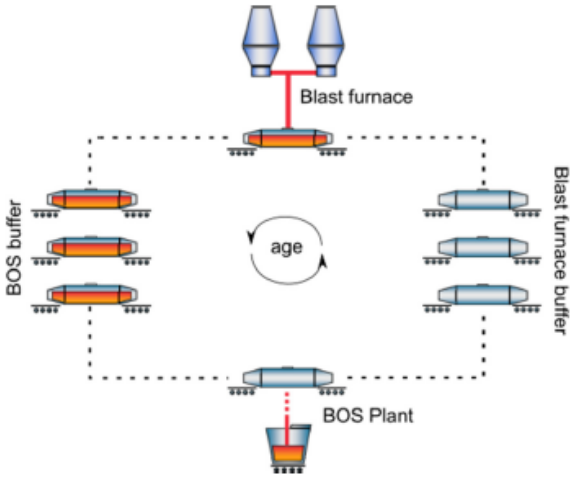


Figure 2.2: Schematic drawing of the torpedo ladle car process. [3]

2.2. Finite element model

The finite element model (FEM) is a numerical method that divides the problem domain into subdomains called finite elements. The model can give a solution for each element. The FEM can be used for a lot of different kinds of physical phenomena, including heat transfer, fluid flow, structural mechanics, etcetera. It is a potent tool that can solve complex problems that would be difficult to solve analytically. The accuracy of FEM depends on the amount of finite elements used; the accuracy of the model increases with increasing finite elements. Also, the computational complexity and time will increase with increasing finite elements used to simulate [5].

In paper [6], the finite element model for the torpedo car process is explained. Torpedo cars move hot metal from the Blast Furnaces 6 and 7 to the BOS2 plant. The heat loss should be minimized as much as possible during this process. Heat losses in the process are affected by the insulating properties of the torpedo ladle's refractory wall, transport and empty time of the torpedo ladle, the amount of hot metal inside the torpedo ladle, the age of the refractory lining and the degree of hot metal penetration into the refractory lining. Getting some insight into the effects of these aspects can help minimize heat loss. A thermal model based on the finite-element method was developed for the IJmuiden torpedo ladles. This finite-element model can calculate the temperature change over time for three temperature profiles inside the torpedo ladle car:

- Refractory temperature of the torpedo ladle car when it is empty
- Refractory temperature of the torpedo ladle car when it is filled with hot metal
- Temperature of the hot metal inside the torpedo ladle car

The paper not only explains that the FEM calculates these temperature profiles of the torpedo car, but it also demonstrates its accuracy by comparing the results to real-world data. Thermal couples placed at specific locations within the car recorded temperature changes, validating the model's effectiveness as a representation of the actual torpedo car used on the plant.[6]

The FEM can produce very accurate temperature-time profiles of the torpedo car process. Unfortunately, it cannot be used as a digital twin. The digital twin needs to be implemented in the plant and needs to be able to produce data that helps to make decisions in a couple of seconds or minutes. The FEM produces data very accurately, but it is very time demanding; creating one, for example, the time-temperature profile of the refractory temperature of the torpedo ladle car empty can take several hours, same for the other 2-time temperature cases. To implement the digital twin for decision-making, not only a one-time temperature profile is needed, but a significant amount is needed. When FEM is used for this decision, it takes days before it can decide which torpedo car would have the least amount of heat loss in the hot metal. Therefore, looking into reduced order models and machine learning is

needed to speed up the process so the decision will not take days or hours but only a couple of seconds. [3]

2.3. Overview of Reduced Order Models

Simulations play a crucial role in gaining deeper insights into complex physical problems. Currently, there are two main limitations in numerical simulation. First, with simulations, it is possible to obtain detailed data about the change of a system over time. However, this information is not always enough to better understand the system's underlying physics. Carefully analyzing this data is needed to make a simpler model to predict the system's behaviour. These data sets can be extensive therefore, this process can be difficult and time-consuming. High-fidelity numerical computations often generate these data sets on finite element models (FEM), which require many configurations. Second, numerical simulations of large systems are too computationally expensive to be used in multi-disciplinary settings. These limitations can be overcome by computed data converted into a lower-order model that can be a basis for digital twins, also known as a reduced-order model [7].

Multiple approaches exist for creating a reduced-order model for a high-dimensional data set. This Section will explain some of these approaches and how the reduced-order model has been established. The reduced-order models are Singular value decomposition in section 2.3.1, proper orthogonal decomposition in section 2.3.2, dynamic mode decomposition in section 2.3.3, and principal component analysis 2.3.4. Furthermore, the limitations, assumptions, strengths, and weaknesses are discussed.

2.3.1. Singular value decomposition

Singular value decomposition (SVD) is a tool that takes high-dimensional data and reduces the data set into only the critical feature, the essential correlation in that data, which can help to interpret, understand, and model that data. SVD is a data-driven tool that identifies patterns by looking at the data without additional system knowledge. It uses mathematical modes to map the system of interest into a new "simple" coordinate system.

To explain the math behind SVD, the large data set X is used. $X \in \mathbb{R}^{n \times m}$, SVD assumes that the matrix is in real space, and the matrix is defined as follows:

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & .... & x_m \\ | & | & & | \end{bmatrix} \tag{2.1}$$

The columns, $x_k \in \mathbb{R}^n$, can be measurements from experiments. For example, matrix X can be images reshaped as column vectors with as many elements as pixels in the image. Therefore, m is the number of images, and n is the number of pixels. In this case, $n \gg m$ because millions of pixels exist in one image. $n \gg m$ is for many systems therefore, the X matrix is often a tall-skinny matrix. SVD takes the X matrix from equation 2.1 and decomposes it as a product of three other matrices:

$$X = U\Sigma V^T \tag{2.2}$$

where U and V are both unitary matrices with orthonormal columns. where U is an $n \times n$ matrix and V is an $m \times m$ matrix. Σ is a matrix with non-negative entries $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m$ on the diagonal and zero off the diagonal. Values greater than σ_m are 0. Computing the whole Σ matrix is very computationally intensive. To decrease this intensity, there is an economical SVD constructed:

$$X = U\Sigma V^T = \begin{bmatrix} \hat{U} & \hat{U}^{\perp} \end{bmatrix} \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix} V^T = \hat{U}\hat{\Sigma}V^T \tag{2.3}$$

Figure 2.3 shows a schematic of the full and economy SVD matrices where $V^T = V^*$. No essential data will be lost using this system because the removed matrix is zero. The U matrix is called the left singular vectors of X, and the columns of V are called the right singular vectors. The diagonal elements of Σ are the singular values. σ is order from larger (σ_1) to smallest (σ_m). The Σ matrix is important due to its ordering, σ_1 is more important than σ_2, and so on [8, 9]. SVD provides a hierarchy of low-rank approximations, and it is possible to obtain a rank-r approximation. Where the leading

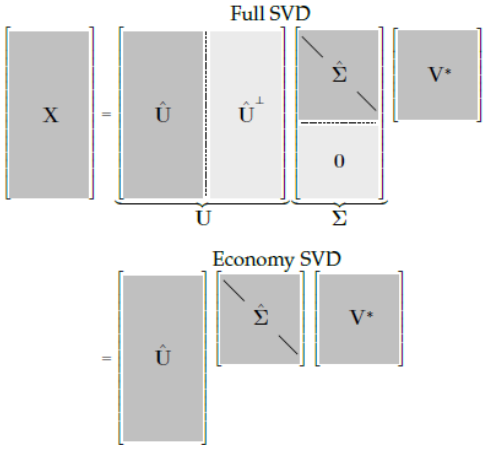


Figure 2.3: Schematic drawing of matrices in the full and economy SVD.[8]

r-singular values and vectors are kept, and the rest is discarded. The theorem of Eckart-Young helps to find the optimal rank-r approximation to X, in the least squares sense, is given by the rank-r SVD truncation $\tilde{X}$:

$$\underset{\tilde{X},s.t.rang(\tilde{X}=r)}{argmin} \left\| X - \tilde{X} \right\|_F = \tilde{U}\tilde{\Sigma}\tilde{V}^T \tag{2.4}$$

Here, $\tilde{U}$ and $\tilde{V}$ are the first r columns of U and V, and $\tilde{\Sigma}$ only contains the $r \gg r$ block. The $\|\|_F$ is the Frobenius norm. To simplify the equation, the truncated SVD basis is denoted by $\tilde{X} = \tilde{U}\tilde{\Sigma}\tilde{V}^T$. Σ is a diagonal matrix therefore the rank-r matrix can be written as followed:

$$\tilde{X} = \sum_{k=1}^{r} \sigma_k u_k v_k^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + ... + \sigma_r u_r v_r^T \tag{2.5}$$

For a given r, there is no better approximation for X than the truncated SVD approximation $\tilde{X}$. This results in that high dimensional data can be well described in a few dominant patterns of the columns $\tilde{U}$ and $\tilde{V}$ [8]. Some singular values in $\tilde{\Sigma}$ can be zero, and the truncated SVD can be exact. For values of r that are smaller than the number of nonzero singular values, the truncated SVD only approximates to X:[8]

$$X \approx \tilde{U}\tilde{\Sigma}\tilde{V}^T \tag{2.6}$$

To decide how many singular values to keep or where to truncate is a crucial decision when using SVD. Several factors play a part in the decision where to truncate, it depends on the desired rank of the system, the amount of noise, and the distribution of the singular values. This can be done using 'crude', identifying the 'elbows' or 'knees' in the singular value distribution, or using the hard-threshold algorithm. In the book [8], Brunton et al. explain in section 1.7 how to choose the truncation rank r while maintaining the most critical data using these techniques.

The literature suggests that using SVD for data reduction has several strengths. One of the SVD's strengths is the robustness to noise, Sadasivan et al. [10] explains how the SVD filters noise components from the starting data set. In addition, Martin et al. [11] explain that SVD is an efficient tool only for small input matrices or when the SVD rank-r is very small. Furthermore, SVD has a very high accuracy of the singular values calculations compared to the actual singular values, explained in [12, 13] by Demmel et al. However, several studies have found some weaknesses in the use of SVD. One of the weaknesses is that SVD can be computationally costly, according to paper [14] by Tzeng. The statement that SVD is computationally costly contradicts the fact that the SVD is computationally efficient. However, the efficiency decreases with increased computational complexity of the system or large matrices. This paper also finds a way to use SVD for larger matrices. The SVD without adjustment is limited to using only small matrices. Furthermore, the singular values and vector created by SVD are hard to interpret because they have no physical or semantic meaning, according to paper [15] by Xue et al. Machine learning algorithms like neural networks can be used to find patterns in these singular values and vectors.

2.3.2. Proper orthogonal decomposition

The proper orthogonal decomposition (POD) is a method that helps find essential patterns in a data set. It is a powerful tool due to its mathematical properties, which make it deployable in various situations. It is a reliable, linear tool that can be used for nonlinear data [16].

The POD applies the SVD method explained in section 2.3.1 to various partial differential equations (PDEs). This makes POD an essential method for the dimensionality reduction of complex, space-time systems. Deriving space and time relationships of complex systems often requires the knowledge of the underlying PDEs that limit complex systems. Solving these nonlinear PDEs analytically is almost impossible. First, the following system of nonlinear PDEs of a single space variable is represented as follows:

$$\begin{aligned} u_t &= N(u, u_x, u_{xx}, ..., x, t, \beta) \\ x &\in [-L, L] \end{aligned} \tag{2.7}$$

$N(.)$ represents the generically nonlinear evolution. β is a parameter that could represent various constants depending on the specific form of the equation and the physical system. For this explanation, $x \in [-L, L]$ will be used as the domain. Various computational techniques have led to an approximate numerical solution of equation 2.7. However, this approximation is highly dimensional and will, therefore, not be efficient. The aim is to reduce the PDEs to a set of ordinary differential equations (ODEs). For example, consider a standard space discretion equation of 2.7where the spatial variables x is evaluated on $n \gg 1$ points

$$u(x_k, t) \;\; for \;\; k = 1, 2, ..., n \tag{2.8}$$

where x is the space variable, with spacing $\Delta x = x_{k+1} - x_k = \frac{2l}{n}$. Using standard finite-difference formulas, space derivatives can be evaluated using neighbouring space points. Such space discretization transforms the PDE of equation 2.7 into a set of n ODEs :

$$\frac{du_k}{dt} = N(u(x_{k+1} + t), u(x_k, t), u(x_{k-1}, t), ..., x_k, t, \beta) \tag{2.9}$$

This process of discretization produces more manageable equations. However, if the accuracy requirements of the system are increased, the dimensions n of the system should increase as well, since $\Delta x = \frac{2l}{n}$. Therefore, the efficiency of the system decreases as well.

Another computational scheme can be considered for solving equation 2.7 by considering a commonly used technique for solving PDEs systems by separating variables. In this method, the assumption made is that the system is space and time-independent:

$$u(x, t) = a(t)\psi(x) \tag{2.10}$$

variable $a(t)$ includes all the time dependencies of equation 2.7 and $\psi(x)$ is the space dependence and is orthogonal. Separation of variables only works analytically if equation 2.7 is linear. In that case, two differential equations can be derived: the space-time dependencies of the complex system. For the general form of equation 2.7, separating variables can create a computation algorithm that can produce accurate solutions. The space solutions are not known beforehand. Therefore, the set of basis modes is assumed to construct $\psi(x)$. These assumptions on basis modes underlie the critical ideas of an eigenfunction expansion method. This produces a separation of variable solutions of the following form:

$$u(x, t) = \sum_{k=1}^{n} a_k(t)\psi_k(x) \tag{2.11}$$

Here $\psi_k(x)$ forms a set of $n \gg 1$ basis modes. The orthogonal properties of $\psi_k(x)$ make it possible to solve equation 2.11. Therefore, a scalar version of equation 2.7 with scalar separable solution $u(x, t) = \sum_{k=1}^{n} a_k(t)\psi_k(x)$ is used to obtain the following equation:

$$\sum_{k=1}^{n} \psi_k \frac{da_k}{dt} = N(\sum_{k=1}^{n} a_k\psi_k, \sum_{k=1}^{n} a_k(\psi_k)_x, \sum_{k=1}^{n} a_k(\psi_k)_{xx}, ..., x, t, \beta) \tag{2.12}$$

These basis functions are orthogonal, which implies the following:

$$\langle \psi_k, \psi_j \rangle = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases} \tag{2.13}$$

The inner product of $\langle \psi_k, \psi_j \rangle$ can be defined as:

$$\langle \psi_k, \psi_j \rangle = \int_{-L}^{L} \psi_k \psi_j^* dx \tag{2.14}$$

Where ψ_j^* is the complex conjugation of ψ_j. Obtaining $a_k(t)$ can be done by using equation 2.11. This can ————————be accomplished by multiplying the equation 2.11 by $\psi_j(x)$ and integrating it from $x \in [-L, L]$. This results in a Galerkin projection of POD:

$$\frac{da_k}{dt} = \langle N(\sum_{j=1}^{n} a_j \psi_j, \sum_{j=1}^{n} a_j (\psi_j)_x, \sum_{j=1}^{n} a_j (\psi_j)_{xx}, ..., x, t, \beta), \psi_k \rangle \quad k = 1, 2, ..., n \tag{2.15}$$

This Galerkin projection is essential because it is often used to perform simulations based on the entire system of equation 2.7. This new system can be created to accommodate the proper solution by choosing the correct modal basis elements ψ_k and the number of modes n. This separation of variables, which is usually for solving linear PDEs, can work for nonlinear and non-constant coefficient PDE if there are enough model basis functions chosen to accommodate all nonlinear mode that occurs in equation 2.15.

The computational efficiency and accuracy of solving nonlinear PDEs shown in equation 2.7 relies on the chosen basis modes. A method that allows for maximum computer efficiency must be constructed by reducing the dimensionality of the data set. Most algorithms produce large systems of size n. Therefore, it is important to construct a data strategy that selects a minimal number of optimal modes (POD modes) that still accurately describes the system of equation 2.7. In order to create the most effective POD modes, the dynamics described by Equation 2.7 are sampled at predefined time intervals. A snapshot u_k consists of samples of complex system, where k indicates sampling at time t_k: $u_k := [u(x_1, t_k)\ u(x_2, t_k)\ ...\ u(x_n, t_k)]^T$. The continuous function and modes will be evaluated at n discrete location in space. This results in a high-dimensional vector representation. There is a large data set X that needs to be analyzed :

$$X = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & ... & u_m \\ | & | & & | \end{bmatrix} \tag{2.16}$$

the columns $u_k = u(t_k) \in \mathbb{C}^n$ can be measurement from experiments or simulations. X consists of a time series of data, with m distinct measurements over time. In the matrix $X \in \mathbb{C}^{n \times m}$ is very large state dimension n. As discussed in section 2.3.1, the equation 2.2 can provide a unique matrix composed from a complex-valued matrix like X. When applying SVD, U will provide the best modes to approximate X in an l_2 sense. The matrix V gives the time history of each of the modal elements. The matrix Σ has a hierarchic weighting of the dominance of the mode. The number of modes generated is determined by the number of snapshots m taken to construct X. To make the system as efficient as possible, the snapshots should still be as low as possible to get an accurate result. Therefore, the r-rank should be approximated by the explained equation 2.6. The low-rank truncation can help construct the modes of interest ψ_k from the columns of truncated matrix $\tilde{U}$. The optimal basis mode will be as follows:

$$\tilde{U} = \Psi = \begin{bmatrix} | & | & & | \\ \psi_1 & \psi_2 & ... & \psi_r \\ | & | & & | \end{bmatrix} \tag{2.17}$$

due to the trunaction only the r most dominant modes are used in equation 2.11. This snapshot-based method is a data-driven computational architecture. It provides an equation-free method, i.e. basis equation 2.7 may be unknown.

The state u of the PDE can be approximated by using a Galerkin expansion

$$u(t) \approx \Psi a(t) \tag{2.18}$$

where $a(t) \in \mathbb{R}^r$ and is a coefficient vector that is time dependent. The plugin in the equation of complex systems according to the Strum-Liouville theory. Multiplying it by Ψ^T gives the dimensional reduction evolution shown in the following equation:

$$\frac{da(t)}{dt} = \Psi^T L \Psi a(t) + \Psi^T N(\Psi a(t), \beta) \tag{2.19}$$

By solving this system of smaller dimensions, the solution of the high dimensional nonlinear dynamical system can be obtained [17, 18].

According to the literature, using POD for data reduction has several strengths. One of its strengths is its high computational efficiency and accuracy, even for large data sets. Mortara et al. [19] explain how the POD method shows high computational efficiency and accuracy when used for nonlinear panel flutter problems. Furthermore, POD is not sensitive to perturbations in data. Rathinam et al. [20] have done a sensitivity analysis for different applications of the POD technique and state in most cases that POD is not sensitive to perturbations. However, the POD cannot be performed when the data set has too much noise. Rathinam et al. also explained another weakness within the POD technique: the computational complexity of POD. In addition, Schmid [21] explains that PODs are limited to normally distributed data with a relatively simple covariance structure. The POD will not give accurate results when data is not normally distributed or has a complex covariance structure.

2.3.3. Dynamic mode decomposition

Dynamic mode decomposition (DMD) was first developed for the dynamics community to identify Space-time coherent structures from high dimensional data. It can be applied to different data sets with dominant or quasi-periodic behaviour. DMD is based on POD and utilizes computational SVD. DMD differs due to modal decomposition, where each mode consists of a spatially correlated structure with the same linear behaviour over time. So, DMD provides dimensional reduction of the data set but can also provide predictions of how these modes will evolve in time.

First, data must be collected and organized if DMD wants to be applied to the system. The data is collected in matrix X and X' where both matrices contain columns that are the vectors of the data, and these vectors change over time, but each column in matrix X' contains one step further in the future than the columns of matrix X:

$$X = \begin{bmatrix} | & | & & | \\ x(t_1) & x(t_2) & \dots & x(t_{m-1}) \\ | & | & & | \end{bmatrix} \tag{2.20}$$

$$X' = \begin{bmatrix} | & | & & | \\ x(t_2) & x(t_3) & \dots & x(t_m) \\ | & | & & | \end{bmatrix} \tag{2.21}$$

The eigenvalues and eigenvectors of the best-fit linear operator A that relates to matrix $X' \in \mathbb{R}^{n \times m}$ and $X \in \mathbb{R}^{n \times m}$ with $n \gg m$ explained in section 2.3.1.

$$X' \approx AX \tag{2.22}$$

The reason for wanting to find the eigenvalues and eigenvectors of the best fit linear operator of A instead of matrix $A \in \mathbb{R}^{n \times n}$ itself is that the matrix is massive and therefore is to computational intensive to calculate. The following formulas will provide a step-by-step procedure for calculating the eigenvalues and eigenvectors of matrix A using the equations of the SVD. The first step is to calculate the values of the equation 2.6. where $\tilde{U} \in \mathbb{C}^{n \times r}$, $\tilde{\Sigma} \in \mathbb{C}^{r \times r}$ and $\tilde{V} \in \mathbb{C}^{m \times r}$. Choosing the rank r is important, as the principled hard-thresholding algorithm can help determine r from noisy data. Combining equation 2.6 and 2.22 the following equations are obtained:

$$A = X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}^T \tag{2.23}$$

$$\tilde{A} = \tilde{U}^T A \tilde{U} = \tilde{U}^T X' \tilde{V} \tilde{\Sigma}^{-1} \tag{2.24}$$

The reduced $\tilde{A}$ has the same nonzero eigenvalues as the full A matrix. Therefore, $\tilde{A}$ only needs to be computed to obtain all eigenvalues of A. From equation 2.24 $\tilde{A}$ can be obtained. Using the SVD explained in section 2.3.1, $\tilde{U}^T$, $\tilde{V}$ and $\tilde{\Sigma}^{-1}$ can be calculated, X' can be obtained by reconstructing the starting data set X. Knowing the $\tilde{A}$ the following equation can be used:

$$\tilde{A}W = W\Lambda \tag{2.25}$$

The diagonal matrix of Λ are the eigenvalues of matrix A, and the columns of W are the eigenvectors of $\tilde{A}$. The eigenvectors W of the reduced system and time-shifted matrix X' can be used to reconstruct the high-dimensional DMD modes Φ [22]:

$$\Phi = X' \tilde{V} \tilde{\Sigma}^{-1} W \tag{2.26}$$

The DMD modes Φ are the eigenvectors of matrix A, proven by:

$$\begin{aligned} A\Phi &= (X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}^T)(X' \tilde{V} \tilde{\Sigma}^{-1} W) \\ &= X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{A} W \\ &= X' \tilde{V} \tilde{\Sigma}^{-1} W \Lambda \\ &= \Phi \Lambda \end{aligned} \tag{2.27}$$

Literature shows that DMD has various strengths. One of its strengths is that DMD is a purely data-driven technique. Schmid [23] explains that DMD can be implemented without knowing the specific mathematical equation determined by the system's dynamics. Schmid et al.[23] also explain the focus of DMD on subdomains within a more extensive complex system, this can be useful if only specific regions of the system need to be analyzed. However, DMD has weaknesses and limitations when applied to specific data. Erichson et al.[24] explain that DMD fails when applied to non-stationary data or intermittent phenomena or broadband. Erichson et al.[24] also explain that DMD is quite noise-sensitive. However, according to Schmid et al., [25], DMD can identify prevalent frequencies and coherent structures in noise.

2.3.4. Principal component analysis

The PCA is used to simplify a data set by reducing a high dimensional data set to a lower dimensional data set. PCA transforms the data to a new coordinate system such that the data with the most significant variance of the projected data will lie in the first principal component (first coordinate), the second most significant variance will be the second principal component, etcetera. Using PCA as a reduced order model technique will help remove most characteristics of the data set that will contribute most to the variance of the data set, doing this by keeping lower-order principal components and ignoring higher-order ones, just like SVD has PCA, a hierarchic system. The only difference in a hierarchic system is that the lower order components are not necessarily the most critical data. This depends on the data set [26].

The principle composition matrix (T) can be computed using the following steps. Matrix X has dimension $n \times p$. First the mean vector $\overline{x_j}$ with $j = 1, .., p$ is determent as follows:

$$\overline{x}_j = \frac{1}{n} \sum_{i=1}^{n} X_{ij} \tag{2.28}$$

The mean matrix will be as follows:

$$\overline{X} = \begin{bmatrix} 1 \\ . \\ . \\ . \\ 1 \end{bmatrix} \overline{x} \tag{2.29}$$

Then, mean-subtraction of the data is performed to find a principal component basis with a minimum mean squared error when approximating the data. This subtracted data is stored in the $n \times p$ matrix B.

$$B = X - \overline{X} \tag{2.30}$$

Then, the covariance matrix of the rows of B is calculated as follows:

$$C = \frac{1}{n-1} B^T B \tag{2.31}$$

It is possible to obtain the principal components by computing the decomposition of C:

$$CV = VD \tag{2.32}$$

Here, the V matrix is equivalent to the V matrix from SVD of x. D is the eigenvalues and C can be computed as the eigenvectors [27].

According to the literature, PCA has several strengths and weaknesses. As noted by Statheropoulos et al. [28], one strength of PCA is that it effectively reduces noise in datasets. Additionally, Gewers et al. [29] explain that PCA is a powerful method for visualising high-dimensional data. However, Karamizadeh et al. [30] explain that when using PCA, the covariance matrix is complex to evaluate accurately. Furthermore, Karamizadeh et al. explain that the PCA can not capture the simplest invariance without additional information from the dataset.

2.4. Overview of Physics-based Machine Learning methods

Machine learning is a subset of artificial intelligence. It can teach computers to learn patterns from data without being explicitly told what to look for. Applying machine learning can improve the performance of specific tasks by learning more about the data they process. Choosing the proper machine-learning application requires choosing a suitable learning algorithm and understanding the problem that needs to be solved. There are several machine learning algorithms, for example, supervised, unsupervised, and semi-supervised [31].

Supervised machine learning generates a function that maps an input to an output value from a given input-output pair data set. The function concludes with labelled training data from the input-output pair data set. The supervised machine learning algorithms need external assistance. The input data set is divided into test and training data sets. The training data set consists of input values with corresponding output variables that must be predicted or classified. The algorithms learn from the patterns from the training data set and apply them to the test data set to predict the output of a particular input value. Here are some critical supervised learning algorithms: k-nearest Neighbors, Decision Tree, and Neural Networks [32].

When using unsupervised machine learning, there is no input-output paired data set; it is only a data set with input values. During this approach, a function will be used to recognize unidentified existing patterns from the input data set. Therefore, the training data set is not labelled, and a statistical learning approach is used to find hidden structures in this data. Some important unsupervised learning algorithms are K-Means, Hierarchical Cluster Analysis, Isolation Forest and Principal Component Analysis [33].

Semi-supervised machine learning is a combination of supervised and unsupervised machine learning. The data set consists of data with input-output pairs and only input data. In other words, the data set contains labelled and unlabeled data. Therefore, most semi-supervised learning algorithms are combinations of supervised and unsupervised algorithms. An example is Deep Belief Networks (DBNs), where unsupervised components are stacked on top of one another. These components are trained unsupervised, and the system is fine-tuned using the supervised learning technique [34].

For this literature review, the focus is on a starting data set that is created by FEM simulation. Here, the data set contains input-output pairs. Therefore, the rest of this section will explain different supervised machine-learning algorithms. Each algorithm's advantages, disadvantages, limitations, and assumptions will be discussed.

2.4.1. Artificial Neural Networks

Artificial Neural Networks (ANN) are computational processing systems that mimic the biological nervous systems. ANNs consist of a large amount of interconnected computational nodes. These nodes are distributed so that artificial neural networks can learn from the input data to optimize the final output data. Figure 2.4 shows a schematic drawing of an artificial neural network [35].

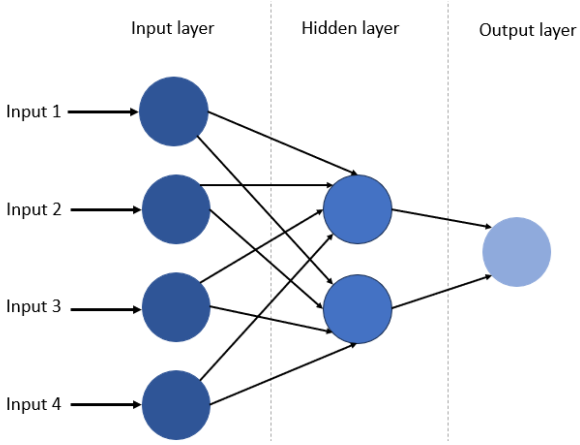


Figure 2.4: Schematic drawing of a Feed-forward Neural Network, consisting of an input, hidden, and output layer. The structure can be the basis for several common Artificial Neural Networks.

The output data $\alpha(p)$ at an input p is estimated using weights and biases that adapt to the data during training. As shown in figure 2.4, the ANN consists of three layers. The input layer is the part where the ANN receives the input to the model. In the figure, the number of nodes is the number of inputs to the model. Then, the hidden layer,l, is not the input or output. There are ℓ_{l} nodes. The ith node in layer l is referred to as η_l^i for $i \in \{1, ..., \ell_{\text{l}}\}$. Each node takes an input, then evaluates the input by an activation function, g, and calculates an intermediate output, o_i^l. This output is then used in the next layer, $l+1$. As shown in the figure 2.4, each input node is connected to each node in the hidden layer or following layer, and information passes through the network in forward motion. The arrow gives the connection in the figure and is a certain weight w_l^i, and a bias, b_i^l, the weight and bias define the importance and effect of a certain input to the output. The activation function at each node is given by $h_i^l = w_i^l o_i^l - 1 + b_i^l$ and therefore depends on the input of the node, the weight given to the connections of the nodes and the bias. The output for node η_l^i is determined by the activation function $o_l^i = g(h_l^i)$. Then, from figure 2.4, the last layer is the network's final output, where the number of output values is equal to the number of nodes. Training inputs with corresponding training outputs train ANN. During the training process, the data iterates back and forth over the network; in doing so, the weights and biases are adjusted so that the output becomes more accurate and similar to the actual output [36].

According to the literature, the use of neural networks has several strengths. One of the strengths of using artificial neural networks is its ability to work with nonlinear complex models that do not need to be specified, as explained by Bourquin et al.[37]. In addition, Livingstone et al.[38] explain that ANN reduces high dimensional data. However, when needed, a decoder can visualize the original data to understand the underlying patterns in the data set. However, ANN also has some weaknesses and limitations. According to Bilbao et al., [39], overfitting is one of the main problems. ANN has a minimal error when working with the training data set. However, when the test set is introduced, the ANN model has a high error rate.

2.4.2. Decision tree regression model

The decision tree regression model divides the input data into several smaller segments. Then, these segments estimate an output value based on the data within a specific segment. Just like the ANN, the Decision tree consists of nodes. However, unlike the ANN, the decision tree has a family tree-like structure. Figure 2.5 shows a schematic drawing of a decision tree. The tree consists of a root node, the starting node, representing the entire input data set. This can also be split up. The interior nodes

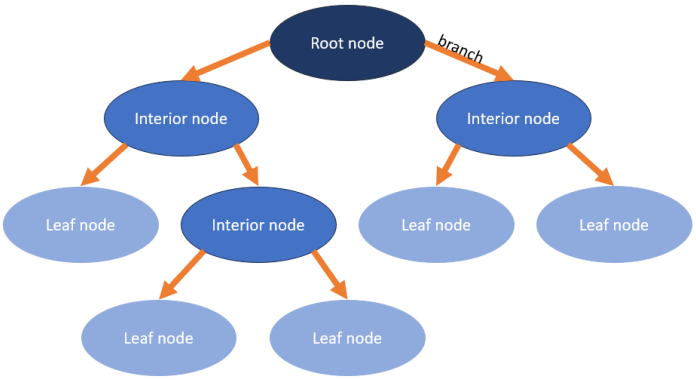


Figure 2.5: Schematic drawing of a decision tree structure

represent some features of the data set, and the arrows are the branches and represent the decision rule. The leaf node represents the outcome.[40].

The literature suggests that using the decision tree model has several advantages. According to Song et al., [41] decision tree is easy to understand and interpret and can be applied to heavily skewed data without resorting to data transformation. The decision tree model uses a non-parametric approach and is robust to outliers in data. However, the decision tree model also has some limitations; Yuvaraj et al. [42] explain that when using a small data set, the decision tree tends to overfit and suggests avoiding this overfitting.

2.4.3. k-nearest neighbors regression model

K-nearest neighbour methods regression is a non-parametric regression model that labels k-nearest patterns in data space. The output value for the new data point is calculated based on the k most similar training data points and averaging their output values. Data points that are closest to the new data points are defined as the k-nearest neighbours.

A standard regression model maps patterns to $\mathbb{R}^d$ subspace. In a regression, labels **y'** $\in \mathbb{R}^d$ for new patterns **x'** $\in \mathbb{R}^q$ are predicted using a set of N observations. The goal of the model is to learn a regression function **f** : $\mathbb{R}^q \rightarrow \mathbb{R}^d$. When **x'** is unknown, the k-nearest neighbour regression model computes the mean of the function using its k-nearest neighbour. The k-nearest neighbour regression is computed using equation 2.33.

$$f_{KNN}(\mathbf{x'}) = \frac{1}{K} \sum_{i \in N\kappa(\mathbf{x'})} y_i \tag{2.33}$$

with $N_\kappa(\mathbf{x'})$ containing the indices of the k-nearest neighbors of the **x'**. Due to the assumption of the locality of equation 2.33 in data and label space, an average of the k-nearest neighbours [43].

The paper by Bansal et al.[44] discusses various strengths and weaknesses of the k-nearest neighbours regression model. One strength of the k-nearest neighbour regression model is its tolerance and resistance to noise, its fast and easy interpretation, and its algorithm, which is easy to apply to problems. However, the k-nearest neighbour regression model also exhibits specific weaknesses; when applying this technique, finding a suitable value for K is hard and has a high computational cost [44].

2.5. Digital twin applications

A Digital Twin consists of several computer-generated models representing a physical object. The models must adapt to operational changes based on real-life data collected from the physical object. In the end, a Digital Twin should be able to predict how the state of an object evolves by exchanging data from the physical to virtual domains [45].

A machine learning-based approach for a mathematical digital twin model that simulates input-output data is constructed by Min et al.[46]. The paper explains a method for this digital twin, taking the following five steps.

1. **Preparation and data collection**. Different simulations like finite element models and real-life measurements are the starting dataset.
2. **Data feature engineering**. This step aims to prepare the dataset by washing and transforming it. ROM can do this to reduce the dimensionality of the data set and filter noise.
3. **Model training and validation**. The collected and transformed data must be split into the validating and training datasets. The training dataset is used in a machine learning tool to create an accurate mapping relationship based on the available data and algorithms; this can be done using different machine learning algorithms. The validation data set validates these training results. The reference accuracy, error of fitting, and coefficient of determination are essential indicators for evaluating the model.
4. **Tryout and optimization**. The training model is tested in a real-time production environment with the latest data to verify its effectiveness and security. After the tryout, the system will be optimized according to the test results and feedback from the model.
5. **Model online deployment**. Here, the final digital twin model is deployed online, and the system will continuously update the optimal control parameters of the product/process based on the digital twin model with real-time data.[46]

The following sections discuss a selection of papers that successfully implemented the concept of creating a digital twin. The literature review focuses on a digital twin for the Torpedo Car Process. Therefore, most papers that are discussed in this section are related to thermomechanical systems.

Digital twin for manufacturing of thermoplastic composites

In [47], Hürkamp et al. developed a digital twin of thermoplastic manufacturing. The interface temperature of the organic sheet used for moulting thermoplastic composites is predicted by creating a digital twin using the POD reduced order model and decision tree machine learning technique. The paper does an error analysis of the digital twin, and the digital twin results are accurate according to the mean approximated error. However, the results can exhibit outliers When looking at the convergence of maximum errors. The reason for this behaviour is that the digital twin does not perfectly capture the behaviour of the real injection moulding machine. According to the paper, this can be overcome using a more suitable machine learning method and parametrization. Also, they assume that the digital twin will become more accurate when they collect more data and learn more about the injection moulding process.[47]

Digital twin for nuclear reactor physics

In the paper [48], Gong et al. succeeded in developing a digital twin to predict the power distribution over the core during the operation stage in the nuclear reactor. They implemented an SVD auto encoder reduced order model to build a non-intrusive forward model with machine learning methods such as KNN and DT. SVD autoencoder is a type of neural network that uses SVD to reduce the system dimensions. This paper compares different digital twins that are developed for this case. Figure 2.6 shows that different ROMs POD and SVD autoencoder combinations and the KNN and DT machine learning techniques are compared. The relative error for the KNN SVD autoencoder is the lowest [48].

Digital twin for flow field monitoring of push-plate Kiln

The digital twin model developed in the paper [49] by Wu et al. is used to predict and generate data on the flow field of the push-plate Kiln. POD is used as ROM to reduce high-dimensional Computational Fluid Dynamics data to low-dimensional features. Then, convolutional neural networks (MCNNs) construct models predicting low-dimensional features for fast flow field prediction. MCNNs are multiple parallel Convolutional neural network (CNNs) architectures or branches within the network. CNNs can be seen as a class of ANN. Yamashita et al. [50] explain how CNN works and its application. Wu et al. compare digital twin models, where the ROM stays the same, but different machine learning techniques are used. In the paper, the mean square error of the different models is compared and shown in figure 2.7. MCNN has the lowest mean square error; according to the paper, the model runs in 0.2s [49].

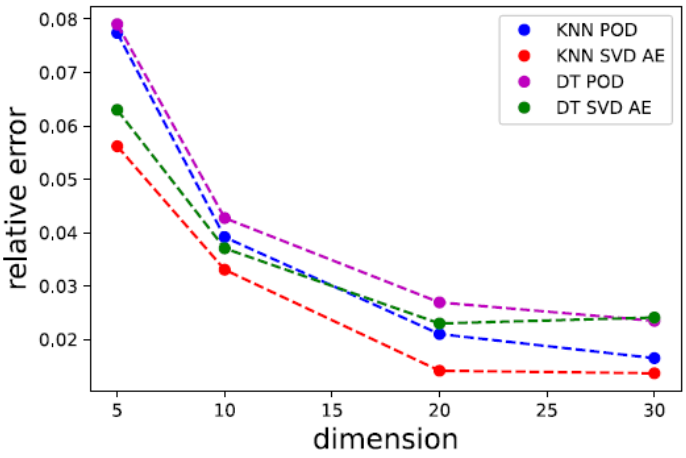


Figure 2.6: Prediction error of different digital twins that are developed. [48]

Network Model	Number of Parameters	Mean Square Error
FNN	8 k	0.00343
CNN	6 k	0.00264
MPCNN	6 k	0.00306
MCNN	**6 k**	**0.00223**

Figure 2.7: Performance comparison of the digital twin using different machine learning techniques [49]

Digital twin for the prediction of the performance of polylactic acid parts

Butt et al. [51] realized the development of a digital twin of polylactic acid parts. To develop this digital twin, three simulations were done of the conjugate convective heat transfer, the multiphase material melting, and the non-Newtonian microchannel. Butt et al. use a commercialized program to reduce the high dimensional simulated data that is used for the digital twin to reduce the computational time. This program uses the POD-Galerkin projection method as ROM. This will be the database required for the machine-learning algorithms of conventional neural networks and random forests. Butt et al. compare the two techniques on accuracy. Random forest is a machine-learning technique that consists of multiple decision trees. For this case, the conventional neural network has an accuracy range of 74% to 88%, and the random forest has an accuracy range of 88% to 95% for the testing data of the output parameters [51].

2.6. Conclusion

This literature review aims to explore alternative methodologies for creating a digital twin of the torpedo car process, focusing on using reduced order models on finite element model data and machine learning techniques.

First, several reduced-order models are discussed in Section 2.3. Choosing the correct reduced order model depends on the starting data set. For example, the dataset's sizes, amount of noise, or outliners are essential factors for choosing the reduced-order model. DMD fails when using non-stationary data. The FEM data consist of stationary data and non-stationary data. Therefore, it is likely that using DMD as ROM may not be the most suitable choice for this digital twin. PCA, SVD and POD are robust, accurate, and efficient methods. The SVD is limited to the use of a small-sized beginning dataset. This can be resolved by adjusting the model.

Second, various machine learning techniques are discussed in Section 2.4. The choice of a machine learning technique for a digital twin depends on the system's characteristics being modelled. ANN and SVM are suitable methods for complex and nonlinear problems. A decision tree is a valuable tool for interpretable models. At the same time, a k-nearest neighbour is more suitable for similarity-based

predictions in data sets and is easy to use and interpret. The machine learning technique analyses the data and finds accurate outcomes.

Four digital twin applications are discussed in Section 2.5. Where different combinations of machine learning techniques with reduced order models are integrated. The digital twin for manufacturing thermoplastics was created using POD as a reduced order model and a decision tree as a machine learning technique. The digital twin for nuclear reactor physics compared different ROM with ML techniques, where the combination of KNN with SVD gave the lowest relative error. The combination of POD with KNN, POD with DT, and SVD with DT gave a higher relative error but still gave accurate results. Digital twin for push-plate kiln flow field monitoring uses POD with CNN and has a low mean square error. Digital twin for the prediction of the performance of polylactic acid parts uses a POD-Galerkin projection method as ROM with CNN and random forest, where the random forest gives the highest accuracy of 88% to 95%, but using CNN still gives accuracy in the range of 74% to 88%.

Different methodologies exist for creating a digital twin of the torpedo car process that can be constructed by looking at the different reduced-order models, machine-learning techniques, and digital twins. PCA, SVD and POD look promising as ROM for the digital twin due to their robustness, accuracy, and efficiency. SVD and POD were implemented in all the digital twins of Section 2.5. Furthermore, various machine learning techniques can be used for the digital twin, but it depends on the relationship of the dataset; therefore, depending on whether the system is complex, linear, nonlinear, or polynomial, a suitable machine learning technique can be chosen. Also, the accuracy can depend more or less on the system, so trying different machine learning techniques in the digital twin to find the most accurate would be factual.

2.6.1. Research plan

Based on the literature review and existing examples of digital twin creation, the following research question is proposed:

Can a reduced-order model combined with a machine learning technique be used to develop a feasible and accurate digital twin for simulating the thermal management of torpedo ladle cars, and if so, which combination provides the best performance?

The following steps provide a framework for investigating the research question.

- **Data Acquisition and Preprocessing** Collect and understand the FEM data that consist of various temperature-time profiles of the torpedo ladle car. Clean this data and order the data so the input and output datasets are clear and ordered.
- **Reduced-Order Model Development** Apply a suitable ROM (e.g., SVD, POD) to the data to decrease the dimensions of the output data to decrease computational cost for the digital twin.
- **Machine learning** Employs different machine learning techniques to find patterns between the input and the reduced output datasets to make predictions.
- **Model validation and evaluation** Asses the model's ability to predict the output data accurately without the model overfitting. Compare the accuracy of this model with the existing ANSYS twin builder model of TATA Steel.

3 Methodology

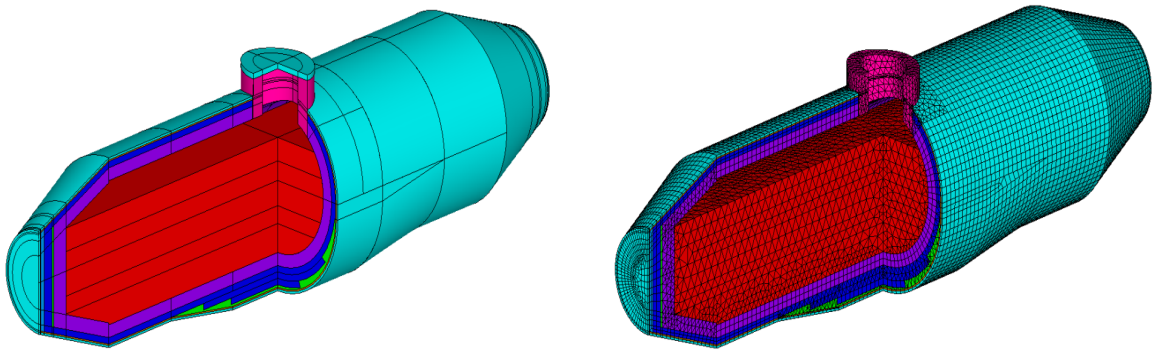
In section 2.6, a framework for investigating the research question has been formulated. As a result, this chapter is organized into distinct sections. The initial part will explain the physical object and the finite element model. Then, information is provided about the current digital twin used by Tata Steel, Ansys Twinbuilder. Following this, we will provide an overview of the prediction model's framework, the data visualization and preprocessing part, and finally, the model development process will be discussed.

3.1. Torpedo Ladle Car

3.1.1. Physical description

For the FEM, a full 3D model is constructed to get a proper description of the radiation of the object. The geometry design is based on the technical drawing of the torpedo ladle car used in the plant. Figure 3.1a shows a $\frac{3}{4}$ view of the torpedo ladles model geometry.[6]

The element mesh of the torpedo ladle car for the model is shown in Figure 3.1b. Eight node elements (SOLID70) are used. These can be increased to 20 elements (SOLID90) to increase the accuracy of the finite element model, but the computational costs will also increase. The element sizes can vary between 2.5 to 15 cm for the refractory linings and up to 30 cm for the hot metal bath.[6]



(a) Geometry of the torpedo ladle

(b) Mesh of the Torpedo ladle

Figure 3.1: The constructed 3D model of the Torpedo Ladle

Preheating

When the preheating is started, all the hot metal bath elements will not be present in the torpedo ladle car, so they are not included in the calculation step. They will not affect the thermal behaviour of the torpedo ladle elements. On the outside of the torpedo ladle, convection heat transfer is applied, which describes the heat transfer due to radiation and convection to the environment. Wind and rain will also affect the heat transfer from the outside but are not considered for the model. The only thermal load

applied to the inside is the convective and radiation heat transfer of the burner for preheating. This is shown in Figure 3.2. The temperature-dependent values of the burners heat transfer coefficient and the corresponding temperature loads are determined by fitting the modelled refractory lining temperatures to the measured temperatures.[6]

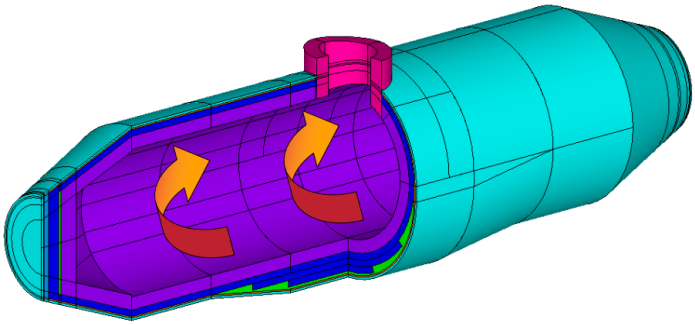


Figure 3.2: During the preheating stage, an effective convective heat transfer is used to mimic the burner that is applied to the inside of the lining of the torpedo ladle. The red arrows show the direction of effective convective heat transfer inside the ladle.

Filled torpedo ladle

During the filling of the torpedo ladle car with hot metal, the hot metal bath elements are assigned an initial temperature. Inside the torpedo ladle car, convective load and radiation load are applied on the refractory wear lining and the bath surface level, shown in Figure 3.3. The heat transfer to the surroundings by convection on the inside of the torpedo car is obtained from computational fluid dynamics calculation. For the optimization of the effective thermal conductivity of the hot metal bath, the heat transfer by convection and conduction is imitated.[6]

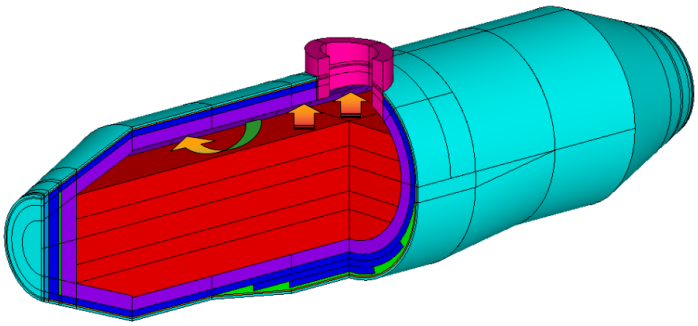


Figure 3.3: Filled torpedo ladle. Convection and radiation heat transfer direction.

Empty torpedo ladle

The preheating model description is similar to the description of the empty torpedo ladle model. The difference is the internal thermal loads: convection and radiation are applied instead of the burner. See Figure 3.4. The convection load is not just natural convection to the surroundings. When the torpedo ladle car empties its contents at the BOS2 plant, it still contains a few tonnes of hot metal when it returns to the blast furnace. The presence of these hot metal remains is not considered in the model. Instead, an effective convection load is applied.[6]

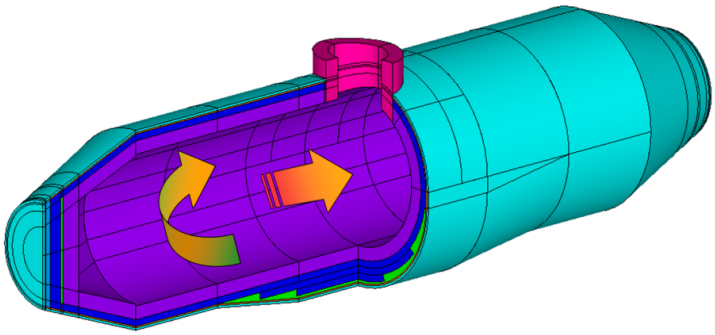


Figure 3.4: An empty torpedo ladle's convection and radiation heat transfer are illustrated. [6]

3.1.2. Finite element model

The finite element model generates data for the digital twin models. The finite element model is validated with thermocouple measurements of the torpedo ladle car. There are 30 thermocouples were used, and the temperature condition of the torpedo ladle car was monitored for nearly eight months. The torpedo ladle was modelled in full 3D using the finite-element method (FEM) in ANSYS. The model contains heat transfer to the environment by radiation and convection based on empirical natural convection relation and computational fluid dynamics calculations from precious research at Tata Steel.

Figure 3.5 show the thermocouple measurements and the FEM simulations results. There is a good agreement between the thermocouple measurements and the FEM simulation results. The only exception is the measured temperature plateau at 100 degrees Celsius during heating up, the evaporation of water causes this and has not been included in the FEM model. The FEM model cover 95% of the scenarios from the past year regarding full and empty times, casting and tapping times, hot metal temperature, hot metal volume, and wear of the refractory lining. This sets the application limits of the thermal-logistic model and provides the temperature data for detailing the thermal-logistics model.[52]

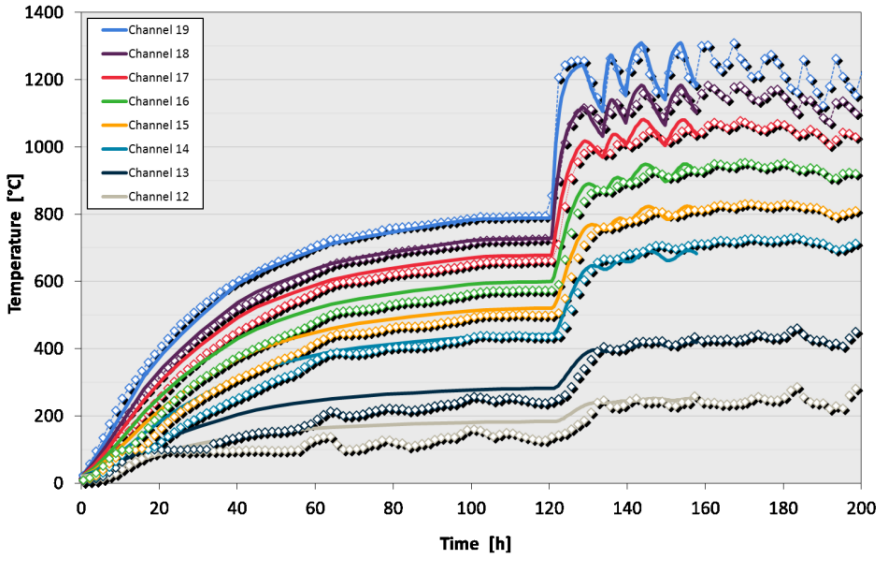


Figure 3.5: The diamonds are the thermocouple measurements, and the lines are the FEM calculated values.[52]

3.2. Outlining of prediction model

The finite-element model can calculate the temperature change over time for three temperature profiles (WL, CM and HM) inside the torpedo ladle car:

- Refractory temperature of the torpedo ladle car when it is empty (CM)
- Refractory temperature of the torpedo ladle car when it is filled with hot metal(WL)
- Temperature of the hot metal inside the torpedo ladle car (HM)

Each temperature profile is essential for creating the digital twin of the torpedo car process. Therefore, to predict the temperature behaviour in the hot metal and the refractory of the torpedo ladle car, it is necessary to develop three separate prediction models: CM, WL, and HM.

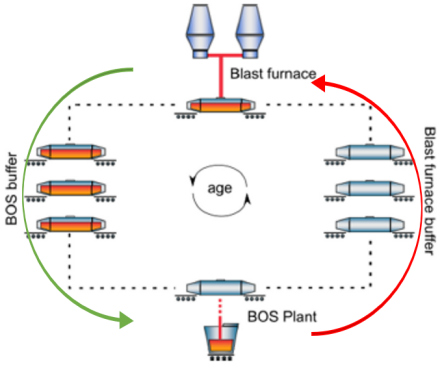


Figure 3.6: The torpedo ladle cars process, moving hot metal from the blast furnace to the BOS plant.

Figure 3.6 shows an illustration of the torpedo ladle car operation. These ladle cars transfer hot metal from the blast furnace to the BOS plant. Throughout this journey, the temperature of the refractory of the torpedo ladle cars undergoes variation. The green arrow in Figure 3.6 indicates the stage where the temperature profile of WL and HM is observed, increasing the refractory temperature as the hot metal temperature decreases. The red arrow indicates the stage where the temperature profile of CM is monitored, leading to a decrease in the refractory temperature during this operation phase.

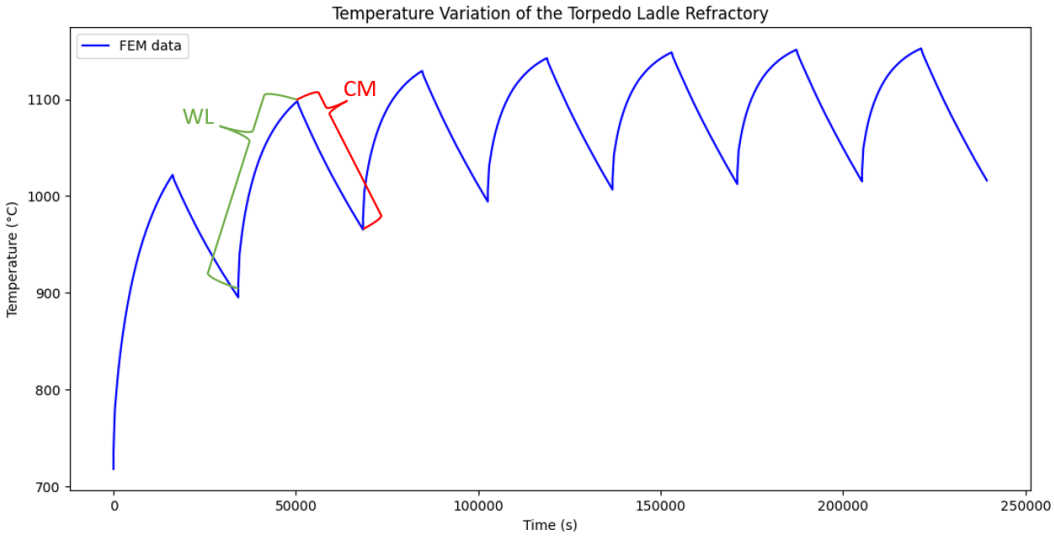


Figure 3.7: Temperature variation of the refractory temperature during several operation cycles.

Figure 3.7 illustrates the temperature variation of the refractory of a single torpedo ladle car. Each cycle of the torpedo car process occurs following a WL and CM prediction, with the number of peaks in

the graph representing the number of cycles the car has undergone. Ultimately, the prediction model aims to predict these temperature variation curves for WL and CM, which are then used to predict the temperature variation of the HM model.

These temperature predictions of each model depend on several parameters, which are the input values for the model, where the temperature is the output. For the WL and HM prediction models, these input parameters include lining age, time, initial temperature of the refractory, initial temperature of the hot metal, and weight of the hot metal. For the CM prediction model, the input parameters consist of lining age, time, and initial temperature of the refractory.

Only the initial input parameters are known during real-life torpedo operations prediction, which occurs at time 0. The time, lining age, hot metal weight, and hot metal initial temperature are known for each cycle. The prediction process begins with the initial input parameters, with the WL model predicting the starting temperature values. After the WL model prediction, the CM model prediction begins with known time and lining age, with the last temperature value of the WL model serving as the initial refractory temperature input value. The CM model then predicts the temperature variation over a specific time, and the last temperature value is used as the initial refractory temperature input for the following WL model prediction. This iterative process can continue for as many cycles as necessary. Furthermore, the initial refractory temperature obtained for the WL model serves as the same initial refractory temperature for the HM model prediction.

As a result, the temperature predictions of CM and WL must be accurate, especially the end temperature value of the prediction. Therefore, examining the difference between the predicted and actual temperatures of the last prediction value is essential. When this difference is minimized, the overall accuracy of the prediction of real-life torpedo operations is enhanced. Because each prediction of WL and CM depends on the previous prediction's last temperature value. When the accuracy of this value increases, the prediction input becomes closer to the correct input, resulting in a better real-life torpedo operations prediction. However, when this value is not close, the error propagates with each cycle, reducing the accuracy of the overall prediction. Therefore, calculating the difference in the end temperature between the predicted values and the actual end temperature values would be a valuable metric for evaluating the prediction model's performance.

3.3. Data visualization and preprocessing

The finite element model is utilized to compute temperature-time profiles for each prediction model: CM, WL, and HM to gather sufficient data for the prediction models. However, each calculation employs different input values that influence the temperature-time profile. For the WL and HM prediction models, these input parameters include lining age, time, initial temperature of the refractory, initial temperature of the hot metal, and weight of the hot metal. For the CM prediction model, the input parameters consist of lining age, time, and initial temperature of the refractory. This results in that the WL and HM model contains 252 datasets with between 13 and 31 temperature values as outputs and five input values. Additionally, for the CM 282 datasets, temperature values were between 13-31 as outputs and three input values.

A plotly package within Python is used to create a dashboard that serves as a tool for visualizing the datasets associated with the CM, WL, and HM models. The dashboard was structured with multiple tabs, each showing different visual-izations for specific aspects of the data analysis. The dashboard visualized the output temperature-time profiles alongside their corresponding input values. This helped us understand how different factors affected temperature changes over time. Furthermore, the visualization can help us identify outliers in the datasets. Additionally, the dashboard featured a tab displaying the distribution of input values, allowing for insights into the diversity and range of inputs the model was trained on.

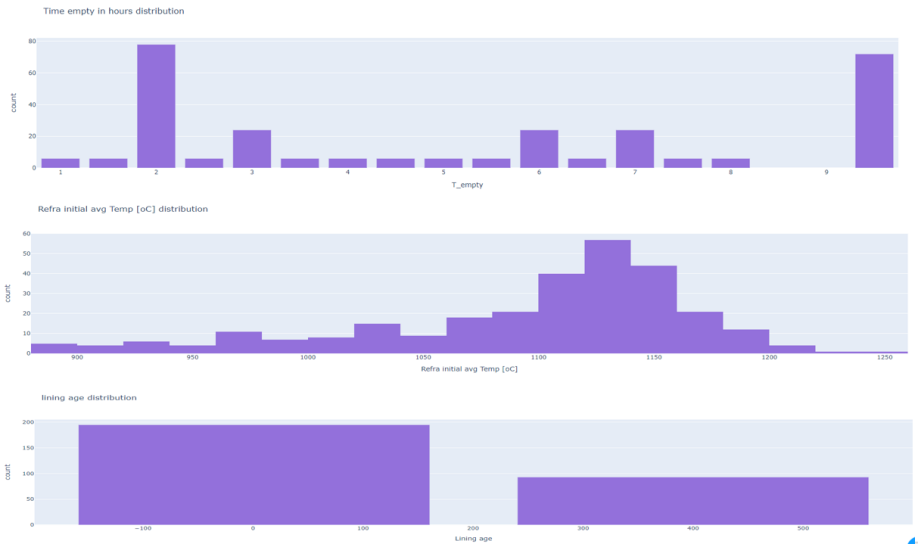


Figure 3.8: Histogram of the input distribution of the CM model

Figure 3.9: Input distribution of the time and lining age for the WL and HM model

Figure 3.10: Input distribution of the hot metal weight, initial refractory temperature and the initial temperature of the hot metal for the WL and HM model.

In Figure 3.8, the histogram illustrates the distribution of input values across the dataset of the CM model. The input distribution of the HM and WL model are shown in Figures 3.9 and 3.10. These input histograms provide valuable insights into the range and diversity of input parameters used for model training. Figures 3.8 and 3.9, we noticed a higher concentration of input values around the 2-hour and 9-hour marks for time. These values represent the minimum and maximum time intervals initially used to construct the ANSYSTwinbuilder model. However, relying solely on these extreme time values led to inaccuracies in predicting using input time values lying between 2 and 9 hours. Additional input values were introduced at intervals between the minimum and maximum times to address this issue. This expansion increased the accuracy of the AnsysTwinbuilder model and helped capture underlying patterns of the data across various time values. Furthermore, appendix D contains several dashboard snapshots, highlighting the visualizations of output temperature-time profiles and corresponding input parameters for the CM, WL, and HM models.

The output values must be uniform in size to employ data for reduced order model (ROM) and machine learning (ML) applications. However, the original data contains variability, with data values ranging between 13 and 31 values. To address this issue, interpolation techniques can create datasets of a uniform size of 31 values. Alternatively, the dataset can be interpolated to a higher number of values. The advantage of interpolation to a higher number of values is that the data becomes more complete due to smoothing out irregularities in the dataset. This results in a more complete representation of the data, and the underlying physical meaning can be captured better. This can lead to more robust and accurate predictions when training the data using machine learning. Therefore, the dataset is going to be interpolated to 4002 values. Appendix A.1 provides a detailed explanation of how data interpolation is performed. However, it is essential to notice that excessive interpolation might introduce artificial patterns or inaccuracies in the data, possibly leading to incorrect interpretations of the model or inaccurate predictions.

3.4. Ansys Twinbuilder

Tata Steel IJmuiden currently uses a commercialized software named Ansys Twinbuilder as a digital twin for the torpedo ladle cars. The Ansys Twinbuilder creates a digital twin using the following steps. First, they interpolate the data, then use an SVD to reduce the dimensions of the data, and then apply a Neural Network for predictions. This software uses Python for the programming, and the Ansys Twinbuilder gives accurate predictions. For several reasons, we need to create the digital twin of the torpedo ladle car in Python rather than relying on the commercialized software of Ansys Twinbuilder.

Since Ansys Twinbuilder is a commercialized software, we lack access to every model detail. Ansys Twinbuilder contains different components, but their specific employment remains unknown. Information such as the number of vectors the data is reduced to, the hyperparameter configuration of the neural network, and the performance evaluation on generalizations are unrevealed. Therefore, this model operates as a black box. Given the traditional nature of the steel industry, ensuring the model is explainable is crucial. Furthermore, Ansys twinbuilder software can be less flexible when adapting to changes in the torpedo ladle car or the process it undergoes. Incorporating new features or integrating extra data into the digital twin becomes challenging. When using Ansys Twinbuilder, we lose the ability to make these adjustments ourselves, which can result in significant time constraints and reliance on external support. Additionally, using Ansys Twinbuilder in the existing system can be more challenging. When creating a digital twin ourselves, it is easier to integrate it into existing software systems and data sources within Tata Steel.

The following bullet points summarize the drawbacks of using the commercialized Ansys Twinbuilder.

- Limited access to detailed model information
- Operates as a black box
- Restricted flexibility and adaptability
- Dependency on external support
- Integration challenges with existing systems

3.5. Model development

From chapter 2, several reduced order models and machine learning techniques are discussed. The goal is to establish a framework for the prediction models by selecting the most appropriate reduced-order model and machine learning techniques.

Tata Steel currently utilizes ANSYS TwinBuilder, a commercial software that has created an accurate prediction model for CM, WL, and HM. This software employs Singular Value Decomposition (SVD) as a reduced-order model and uses feedforward neural networks as the machine learning technique for prediction tasks. The effectiveness of SVD in TwinBuilder, along with the specific characteristics of the data, results in SVD being a suitable technique for reducing the dimensions of the output data.

The effectiveness of the feedforward neural network model in TwinBuilder makes it a suitable choice for a machine learning tool, especially given the complexity of the model. However, considering the challenges associated with hyperparameter tuning, the time-intensive nature of neural network training, and the complexity of the model's architecture, we have chosen to utilize Random Forest in the initial structure of the prediction model due to its simplicity. Random Forest offers benefits such as more straightforward hyperparameter tuning, providing insights into feature importance, and less sensitivity to overfitting than decision trees. Therefore, using a random forest to build the initial prediction model is a suitable choice. Once this model is established, we can explore more advanced machine learning techniques like feedforward or recurrent neural networks.

This thesis employs various frameworks for developing prediction models; each of these models is compared to the commercialized ANSYS TwinBuilder model and the finite element model data. The objective is to identify the most accurate prediction model for the given task. The models and their respective structures are outlined below:

- **Random forest model (RF-SVDmodel)**: Utilizes Singular Value Decomposition (SVD) for data reduction and employs a random forest algorithm for prediction. Figure 3.11 shows a schematic drawing of the model.

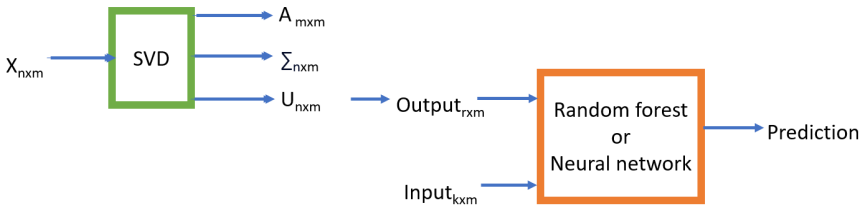


Figure 3.11: A flow chart of the RF-SVDmodel and the NN-model is shown. Where n are the temperature values from the FEM model, m is the number of datasets, k is the input values, and r is the r-rank chosen for the SVD.

- **Neural network model (NN-SVDmodel)**: Also, employs SVD for data reduction and uses a neural network for prediction. Figure 3.11 shows a schematic drawing of the model.
- **Neural network model without SVD (NN-model)**: Does not employ SVD; instead, the data is interpolated to 31 values from the original 4002. A neural network is employed for prediction. Figure 3.12 shows a schematic drawing of the model.

Figure 3.12: A flow chart of the NN-model is shown. Where n are the temperature values from the FEM model, m is the number of datasets and k is the input values.

- **Recurrent neural network model (RNN-model)**: Similar to the NN-model, this model does not use SVD; the data is interpolated to 31 values. However, it uses a recurrent neural network for prediction.Figure 3.13 shows a schematic drawing of the model.

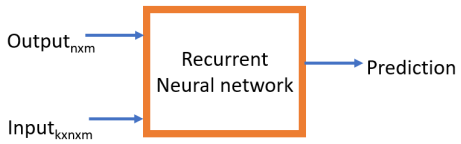


Figure 3.13: A flow chart of the RNN-model is shown. Where n are the temperature values from the FEM model, m is the number of datasets and k is the input values.

3.5.1. Model evaluation and criteria

The performance of the four different models is evaluated by using specific metrics. The following steps will be taken to evaluate the models, and certain criteria will be tested to determine the best model.

- **Model Generalization Metrics: k-fold cross-validation and Learning Curves**

  First, we want to evaluate the model on its generalization capabilities, where for the RF-model, the k-fold cross-validation is performed, k-fold cross-validation is explained in appendix A.5. For the NN-SVDmodel, NN-model, and RNN-model a learning curve is obtained during the training of these models, where the analysis of these curves is explained in appendix A.4. Both metrics are used to identify the models that are overfitting or underfitting.

  Criteria

For k-fold cross-validation, the variance of the model's performance across various data subsets should be stable, as this indicates that the model is generalizing well. Therefore, the R^2 error and RMSE on the test data for different folds should either be similar or should not exhibit significant variance. The learning curve evaluates the model by comparing its performance on training and validation data, aiming for the two curves to align. This alignment indicates that the model effectively generalises to unseen data, as the training and validation sets exhibit similar performance trends. A small generalization gap of up to 5% in loss between the training and validation sets indicates that the model is effectively learning the underlying patterns in the data without overfitting or underfitting and, therefore, still acceptable for models with good generalization capabilities.

- **Performance Evaluation on test and Validation Datasets**

  The datasets for each model are split into 80% for training and 20% for testing. This section evaluates the performance of the models by calculating the R^2 and RMSE on test data predictions, as explained in Appendix A.2 and A.3, respectively. In addition, a second validation was also made to verify the generalization performance for WL, HM, and CM on each cycle in the torpedo operation. An unseen input dataset and a lining age of 100 were considered. The model was not trained at this age, which provides extra confirmation if the model captures the underlying patterns in the dataset. There are three validation datasets named lining age 1, lining age 100, and lining age 400, which are temperature profiles of the refractory during multiple cycles and the hot metal temperature inside. Furthermore, for the real-life application of the model, the end temperature of the WL prediction is used as the initial refractory temperature input for the CM model. Therefore, Δ(C°) is calculated, representing the difference between the end temperature of the prediction and the actual end temperature.

  Criteria

  The R^2 and RMSE across different test data should be similar. Therefore, the model does not over or under-perform on specific inputs. The validation datasets can confirm that the model is not overfitting and understands the underlying patterns of the model when the model is predicting well. In this case, the R^2 should be close to 1. Also, a R^2 close to 1 means that the predicted data patterns closely match the actual values. Besides the model's accuracy on these predictions,$\Delta(C°)$ should also be as low as possible to minimise error prediction during real-life operation prediction.

- **Prediction Performance Evaluation and Comparison of Models**

  We only know the first input value when predicting real-life torpedo operations. The model prediction on the real-life torpedo operation uses the first input value to calculate the refractory temperature using the WL model and the hot metal temperature using the HM model. The end temperature of the WL model prediction will be used as an input value for the CM model prediction, and so on. These predictions are made for the different models. The performance of this real-life operation prediction is evaluated using RMSE and R^2. These models are also compared to the existing AnsysTwinbuilder model.

  Criteria

  Aim for the lowest possible RMSE in the HM model to ensure optimal performance across various digital twin models. Even a one-degree variance in hot metal temperature can have significant implications, potentially saving up to one million euros. Furthermore, a R^2 close to 1 means that the predicted data patterns closely match the actual values. Therefore, the RMSE values and R^2 of this prediction will be used to find the model with the best performance, for R^2 close to one and RMSE ideally close to zero.

4

Model

This chapter explains the process of obtaining the r-rank within the Singular Value Decomposition for both the RF-SVD and NN-SVD models. We also explore the various hyperparameters of these models, detailing the strategies employed to optimize them for increasing model performance. Finally, we provide the specific hyperparameters utilized in our models, offering insights into their architecture and configuration.

4.1. Singular value decomposition

In section 3.5, both the Random Forest model (RF-SVDmodel) and the Neural Network model (NN-svdmodel) employ data interpolation and singular value decomposition (SVD) techniques to reduce the data dimensions. In section 2.3.1, a more detailed explanation of the mathematics behind SVD can be found. This section will explain how the SVD is applied to the HM prediction model and how the r-rank of the SVD is found.

The scikit-learn package is utilized to implement SVD on the output data. Initially, the HM model's matrix X, has dimensions of 4002×282, where 282 represents the number of datasets, each containing 4002 temperature values. Through SVD, we can decrease the data size from having one singular vector per dataset to 4002 singular vectors per dataset, resulting in a truncated left singular value matrix denoted as U. Determining the minimum number of vectors required for the matrix is essential to maintain a reliable representation of the original data.

In scikit-learn, the truncatedSVD command is employed to reduce the dimensionality of the matrix. The $n\_components$ parameter can be adjusted to specify each dataset's desired number of vectors. The following steps are undertaken to determine the optimal $n\_components$ value that accurately represents the original data.

We start by employing truncatedSVD to iteratively reduce the matrix X to lower dimensions, ranging from $n\_components$ 1 to 4002. This results in a series of left singular vector matrices, denoted as U_1, U_2, U_3, all the way up to U_{4002}. Each of these matrices captures a different amount of information from the original dataset. For instance, U_1 represents the least amount of information, as it consists of one vector representing 4002 values, while U_{4002} contains the most information.

Next, we utilize these matrices, U_1, U_2, U_3, ..., U_{4002}, to reconstruct the original matrix X. We obtain a series of re-constructed matrices, denoted as X_1, X_2, X_3, ..., X_{4002}. These matrices offer insights into how well the dimensionality reduction process preserves the original data.

To evaluate the dimensionality reduction, we calculate the relative error between each of the obtained matrices, U_1, U_2, ..., X_{4002}, and the original matrix U. This comparison provides valuable feedback on how well each dimensionality-reduced matrix represents the original data. The resulting relative error values are then plotted against the $n\_components$ for the HM model in Figure 4.1.

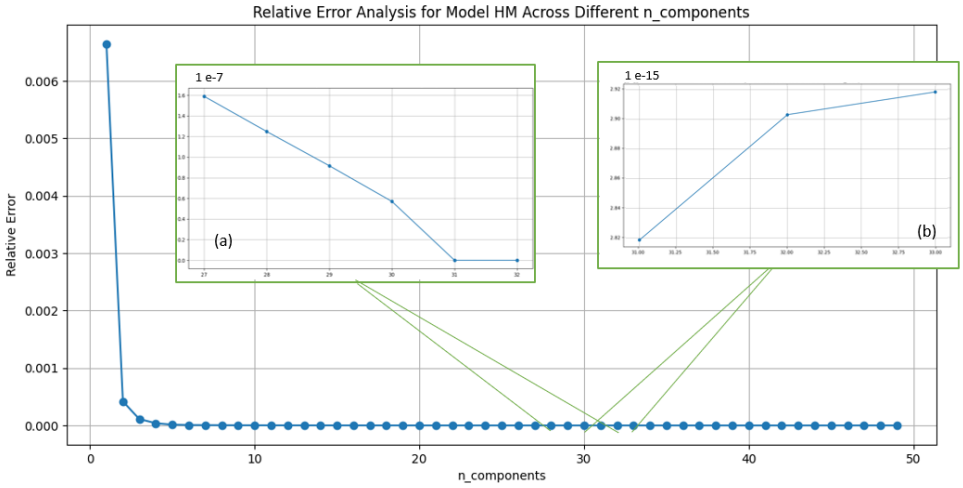


Figure 4.1: Relative error of U_m ranging from 1 to 50. Subfigures (a) and (b) are zoomed views showing the relative error for U_m 28 to 32 and 31 to 33, respectively.

Upon closer examination, we observe a large reduction in relative error between matrices U_{30} and U_{31} when looking at Figure 4.1 (a). Where after $n\_components$ 31, the relative error increases according to Figure 4.1 (b). These two figures demonstrate that the relative error in U_{30} is on the order of 10^{-7}, while in U_{31}, it is on the order of 10^{-15}. For U_{31} and higher, the relative error stays in the magnitude of 10^{-15}. Therefore, choosing $n\_components = 31$ is the most optimal amount of $n\_components$ due to its accuracy and the low dimensionality.

The way of employing the SVD for the HM model is identical to that of the WL and CM models. Therefore, how the SVD is applied and how the r-rank is found for the CM and WL models can be found in the appendix B.1. For models CM and WL, the optimal amount of $n\_components$ is 31.

4.2. Random forest model (RF-SVDmodel)

In section 4.1, the dimensionality reduction process is performed, and the original output data is reduced from 4002 temperature values per dataset to 31 vectors per dataset. In the RF-SVD model, in addition to applying SVD, random forest is integrated as a machine learning tool for prediction. Random forest combines the multiple decision trees to make predictions by combining their outputs. Section 2.4.2 elaborates on the functionality of decision trees in data prediction.

For the CM model, every output dataset is associated with three input values and for the WL and HM model, every output dataset is associated with five input values. To facilitate model training and evaluation, the input and output values are partitioned into an 80% segment for training and the remaining 20% for testing. To create a randomforest model we use Python packages scikit learn. From this package, the RandomForestRegressor tool is used for the random forest prediction. This module features hyperparameters that need to be adjusted to optimize model accuracy. In Table 4.1, the hyperparameters of the RandomForestRegressor are listed along with their purposes, including how they contribute to tuning and managing overfitting.

Table 4.1: RandomForesRegressor hyperparameters and their purpose

Hyperparameters	Purpose
$n\_estimators$	Number of trees in the random forest
max_depth	Maximum depth of the trees inside the randomforest. If None, nodes are expanded until all leaves are pure or until all leaves contain less than $min\_samples\_split$ samples. Deeper trees may capture more complex patterns in the data but can also lead to overfitting
min_sample_split	Minimum number of samples required to split an internal node. It helps prevent overfitting by controlling how finely the trees are partitioning the input space.
min_sample_leaf	Minimum number of samples required at the leaf node. It also prevents overfitting by imposing constraints on the tree structure.
$max\_features$	The number of features to consider when looking for the best split. Limiting this number can prevent the model from focusing too heavily on a subset of features and improves generalization
$Bootstrap$	Bootstrap samples are used when building trees. This parameter can be turned on or off. Turning this off may lead to less correlated trees and potentially increase the perfomance.

To optimize the performance of the RandomForestRegressor model, we had to adjust its hyperparameters. The process began with a grid search. Grid search was enabled to explore hyperparameter combinations by systematically sampling from specified ranges. This approach provided insight into the effect of different hyperparameter settings on model performance and helped us establish the range of the hyperparameters to use.

Once the range of the hyperparameters was established, k-fold cross-validation was performed to assess the model's performance across multiple subsets of the dataset. Through this process, we verified that the model performance remained stable across different folds, which could give insight into the model's generalization abilities. The model was fixed to the random state of the data splitting during hyperparameter tuning.

The R^2 score of the test datasets was calculated for each combination of hyperparameters within the identified range. This evaluation allowed for the comparison of model performance under different hyperparameter settings, and the selection of the most effective configuration could be found. The most effective configurations were further evaluated on an unseen validation set to confirm their generalization capability and identify potential signs of overfitting. This validation step was essential for selecting a model with solid performance on new, unseen data.

Finally, the selected models underwent fine-tuning by adjusting the hyperparameters slightly. This iterative process aimed to optimize model performance further and identify the optimal combination of hyperparameters that yielded the best results. The following hyperparameters were found for the best model:

Table 4.2: Hyperparameters of the RF-Model for the CM, WL and HM model

Hyperparameters	CM Model	WL Model	HM Model
$n\_estimators$	100	200	100
max_depth	None	None	none
min_sample_split	2	2	2
min_sample_leaf	1	1	1
$max\_features$	sqrt	sqrt	1.0
$Bootstrap$	True	True	True

4.3. Neural network models

In section 4.1, the dimensionality reduction process is performed, and the original output data is reduced from 4002 temperature values per dataset to 31 vectors per dataset. SVD and neural networks were used for the Neural network model (NN-SVDmodel), only the neural network model was used in the Neural network model without SVD (NN-model) and only the Recurrent neural network was used in the Recurrent neural network model (RNN-model). The hyperparameter tuning for all of these models is similar.

To create a neural network model, we used the Python package TensorFlow. From this package, Keras was used to create neural networks for prediction. This model features several hyperparameters that need to be adjusted to create a model with the best performance. In table 4.3, the hyperparameters of the NeuralNetwork are listed with their corresponding purpose.

Table 4.3: Neural network hyperparameters and their purpose

Hyperparameters	Purpose
Number of layers	Neural networks with more layers can learn more complex patterns but are also more sensitive to overfitting.
The number of neurons per layer	Affects the capacity and expressiveness of the model. Increasing the number of neurons enables the model to capture more complex relationships in the data but becomes more sensitive to overfitting.
Activation functions	Activation functions introduce non-linearity in neural networks, enabling them to capture complex patterns. It directly influences the model's ability to learn effectively and generalize to unseen data.
Learning Rate	The learning rate determines the size of steps taken during training. Higher rates may lead to faster convergence but could cause instability or overshooting, while lower rates result in slower convergence but more stable training.
Batch size	The number of training samples used in one training. Smaller batch sizes lead to noisier updates but are less prone to overfitting. Larger batch sizes can lead to faster training times but a higher risk of overfitting.
Regularization techniques	Such as L1,L2 regularization, dropout, and batch normalization can help prevent overfitting by imposing constraints on the model's parameters or by introducing noise during training
Optimizer	The optimization algorithm is used to update the model's parameters during training.

The hyperparameters needed to be adjusted to optimise the Neural network model's performance. We began the hyperparameter tuning process by creating an early stopping during the neural network model training. Where the validation loss of the model is being monitored. The model stops automatically when the validation loss curve does not decrease any more. This prevents the model from training for too long. When the model trains for too long, the model may start to overfit the training data.

For the hyperparameter tuning, we set up a baseline neural network model with a single layer and a small number of nodes. This baseline evaluated different activation functions by examining the learning curve and an R^2 score RMSE of the test data. This gave insight into which activation function would be optimal for integration into the neural network architecture.

When the activation functions that yielded the best results were established, we can now increase the complexity of the model. This is done by adding layers, increasing the number of neurons, changing the learning rate and batch size. Here the performance of the model is monitored by the learning curve, R^2 score and RMSE of the test data, and the results of the validation data to see which neural network architecture would give the optimal model. Furthermore, some L2 regulations are added to the layers. This resulted in a decrease in the generalization gap in the learning curve.

This hyperparameter fine-tuning resulted in an optimized model. The NN-SVDmodel and the NN-

model have different architectures. The hyperparameters of the NN- model and NN-SVDmodel are given in Table 4.4. The hyperparameters for RNN-model are given in Table 4.5. For the NN-SVDmodel and NN-model the dense layers in Keras are used to create layers in the neural network, for the RNN-model the SimpleRNN layers are used to create the Recurrent neural network.

Table 4.4: NN-SVDModel and NN-Model Architecture and Hyperparameters. All models have a batch size of 16 and a learning rate of 0.001. R is the regularization technique; regularizer L2 was used in this case.

NN-SVDmodel	**Layer 1**	**Nodes**	**R**	**Layer 2**	**Nodes**	**R**	**Layer 3**	**Nodes**	**R**
CM model	mish	32	1e-3	swish	32	1e-3	relu	32	1e-3
WL model	gelu	64	1e-3	mish	64	1e-3	linear	32	1e-3
HM model	gelu	64	1e-3	linear	64	1e-3	x	x	x
NN-model	**Layer 1**	**Nodes**	**R**	**Layer 2**	**Nodes**	**R**	**Layer 3**	**Nodes**	**R**
CM model	relu	64	1e-3	linear	64	1e-3	linear	64	1e-3
WL model	relu	128	1e-3	linea	128	1e-3	linear	128	1e-3
HM model	gelu	64	1e-3	linear	64	1e-3	x	x	x

Table 4.5: RNN-Model Architecture and Hyperparameters. All models have a batch size of 16 and a learning rate of 0.0001. R is the regularization technique; regularizer L2 was used in this case. The model's CM, WL, and HM have the same number of nodes for each model in all layers, which is given in the nodes column.

RNN-model	**Layer 1**	**R**	**Layer 2**	**R**	**Layer 3**	**R**	**Layer 4**	**R**	**Nodes**
CM model	gelu	1e-2	mish	1e-2	elu	1e-3	linear	1e-3	128
WL model	gelu	1e-2	swish	1e-2	elu	1e-3	linear	1e-3	256
HM model	elu	1e-2	mish	1e-2	mish	1e-2	linear	1e-3	128

5

Results and discussion

The models were evaluated using various metrics. Initially, their generalization capabilities were tested through k-fold cross-validation and Learning Curves. Furthermore, a performance evaluation of the test and validation datasets was performed. Finally, the prediction performance and comparison of the models were evaluated in real-life operation prediction scenario. Section 3.5.1 provides a detailed explanation of the evaluation metrics and their criteria.

5.1. Model Generalization Metrics

Initially, we examine the RF-SVD model for indications of overfitting through k-fold cross-validation on the test data. This step is essential for evaluating the model's ability to generalize effectively while reducing the risk of overfitting the training data. For the NN-SVD model, NN-model and RNN-model, the learning curve of the models are evaluated to understand the model's training dynamics and potential overfitting tendencies.

5.1.1. Random forest model (RF-SVDmodel)

On the RF-SVD model, a k-fold cross-validation was performed, as explained in the appendix A.5. The K-fold cross-validation provides insights into the model's tendency to overfitting. In the case of the RF-SVDmodel the dataset is divided into five equal-sized "folds". The model was trained and tested five times, using a different fold as the test set.

Table 5.1 represents the results of k-fold cross-validation, where the test data and predicted values by the RF-SVDmodel are evaluated for each fold. For the CM, WL, and HM models, the mean R^2 and RMSE of the predicted values of each fold are calculated. Appendix B.2 gives some visuals where the predicted values of each fold are plotted against the test data.

Table 5.1: k-fold cross validation results of the CM, WL and HM model where the mean R^2 and the RMSE of each fold is shown.

Model	Fold 1		Fold 2		Fold 3		Fold 4		Fold 5	
	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
WL Model	0.84	11.02	0.73	13.67	0.86	11.60	0.6	12.94	0.72	14.37
CM model	0.96	6.09	0.94	6.95	0.81	7.06	0.81	7.95	0.89	7.44
HM model	0.96	5.05	0.9	4.9	0.94	6.40	0.96	4.3	0.97	5.98

The results of the k-fold cross-validation show a high mean R^2 across the folds for HM, CM and WL models. The mean R^2 values for the CM model range from 0.81 to 0.96, whereas for the WL model, they range from 0.6 to 0.86, and for the HM model, they range from 0.94 to 0.96. The models CM and HM have a minimal variance of mean R^2, which indicates that these models have accurate prediction

abilities and capture the underlying data patterns. Additionally, the minimal variance observed in the mean R^2 across the folds suggests overall stability in the models. However, due to the slight variance, especially in the WL and CM models, the potential for overfitting cannot be ruled out entirely.

Furthermore, the CM and HM models consistently have low RMSE values across all folds; CM ranges from 6 to 8 degrees Celsius, and HM ranges from 4 to 6 degrees Celsius. However, the RSME values of the WL model are relatively higher, ranging from 11 to 14 degrees Celsius. CM and HM models have accurate predictions according to the low RMSE. The higher RMSE in model WL may indicate a lower predictive accuracy compared to CM and HM. Nevertheless, all models demonstrate a consistent performance across all folds. This stability in RMSE values further reinforces the reliability of the models.

5.1.2. Neural network models, NN-svdmodel, NN-model and RNN-model

The learning curve is used to evaluate the generalization capabilites of the NN-SVDmodel, NN-model and RNN-model. The learning curves of HM for all three models are presented in this section and show the progression of training and validation loss over training iterations.

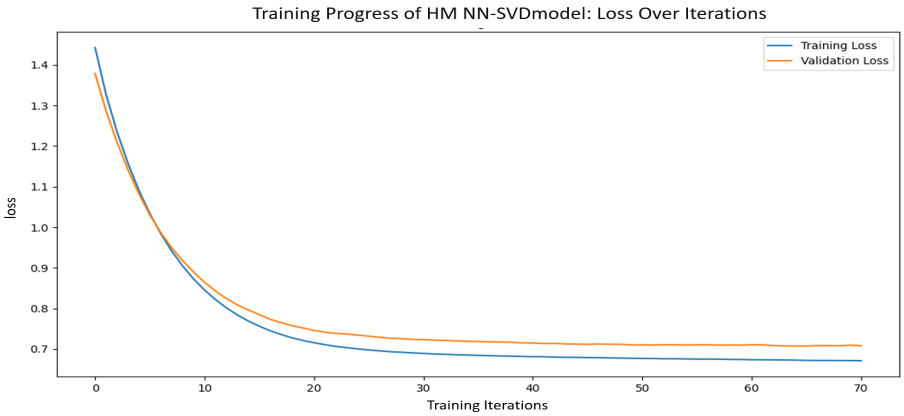


Figure 5.1: Learning curve of the HM model of NN-model

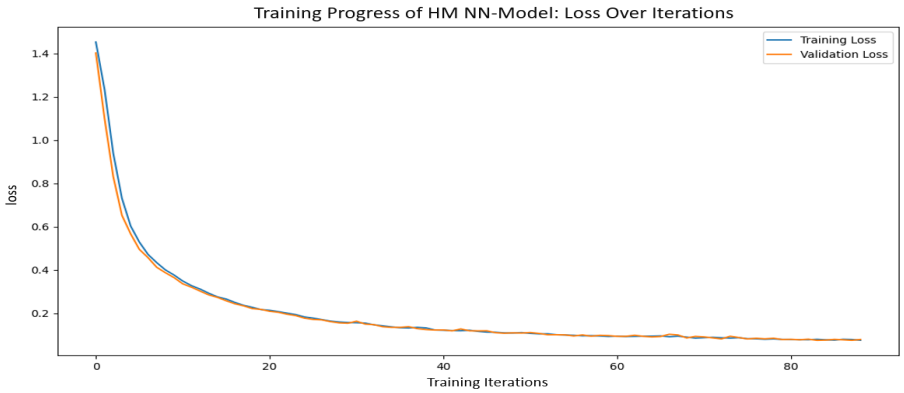


Figure 5.2: Learning curve of HM model of NN-SVDmodel

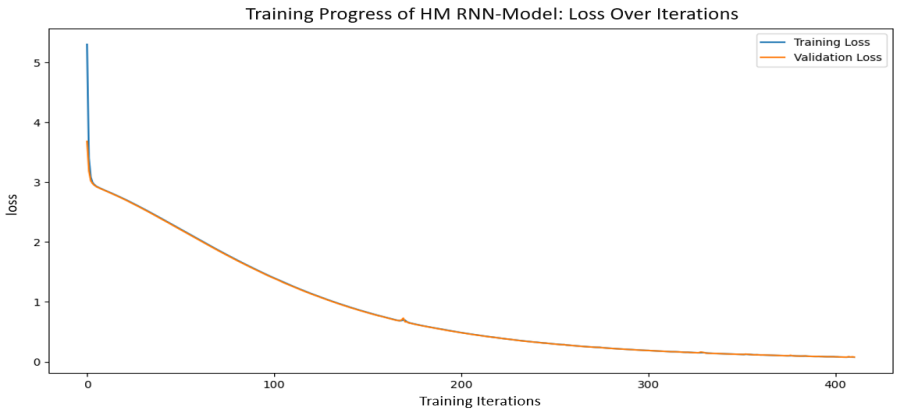


Figure 5.3: Learning curve of the HM model of RNN-model

The results of the learning curves for models WL and CM can be found in appendix B.3 and B.5. Due to their similarity to the WL and CM model, the discussion of the learning curve of WL and CM is similar to the HM model. In Figure 5.1, representing the learning curve of NN-model, Figure 5.2, shows the learning curve of NN-SVDmodel and Figure 5.3, representing the learning curve of RNN-model. We see that the validation and training loss curves overlap for all three models. This overlap indicates a good learning behaviour of the models during training, suggesting effective learning without significant overfitting or underfitting.

5.2. Performance Evaluation on test and Validation Datasets

This section will evaluate the models' performance by calculating the R^2 and RMSE on test data predictions. Furthermore, we want to assess the model's accuracy using a validation dataset with input data on which the model is not trained. This serves to validate the models' performance and generalization capabilities.

5.2.1. RF-SVDmodel and NN-SVDmodel

The R^2 curves of the test data results for the HM model of RF-SVDmodel and NN-SVDmodel are shown in Figure 5.4 and Figure 5.5, respectively. For RF-SVDmodel, R^2 values ranging between 0.65 and 0.999 is obtained. Additionally, the RMSE of the test indices ranges from 0.040 to 18.65 degrees Celcius. For NN-SVDmodel, R^2 values ranging between 0.95 and 0.999 is achieved. The RMSE of the test indices is ranging between 0.137 to 7.873 degrees Celsius.

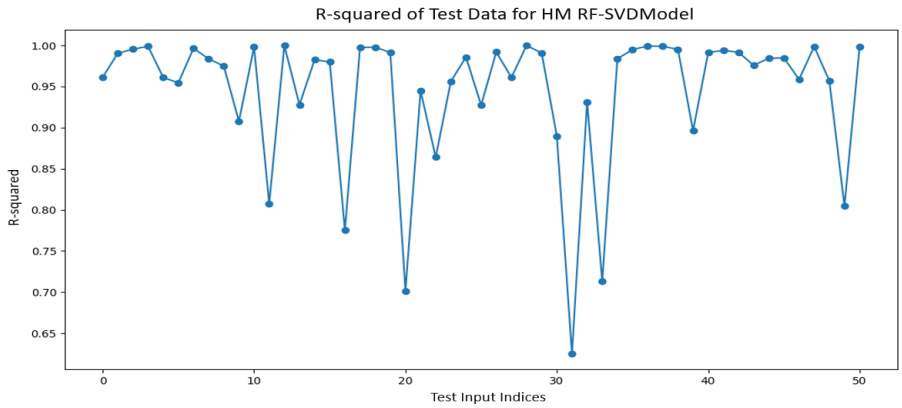


Figure 5.4: R^2 of test data for HM RF-SVDmodel. RMSE=4.017 of the whole test dataset, where RMSE of the test indices ranges from 0.040 to 18.65.

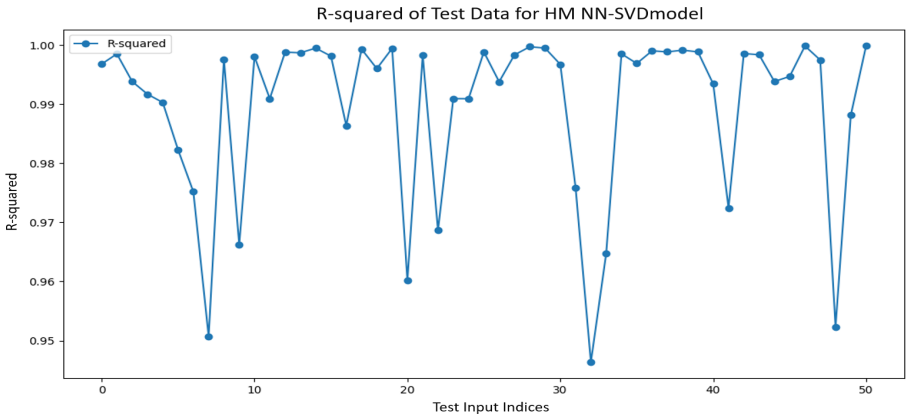
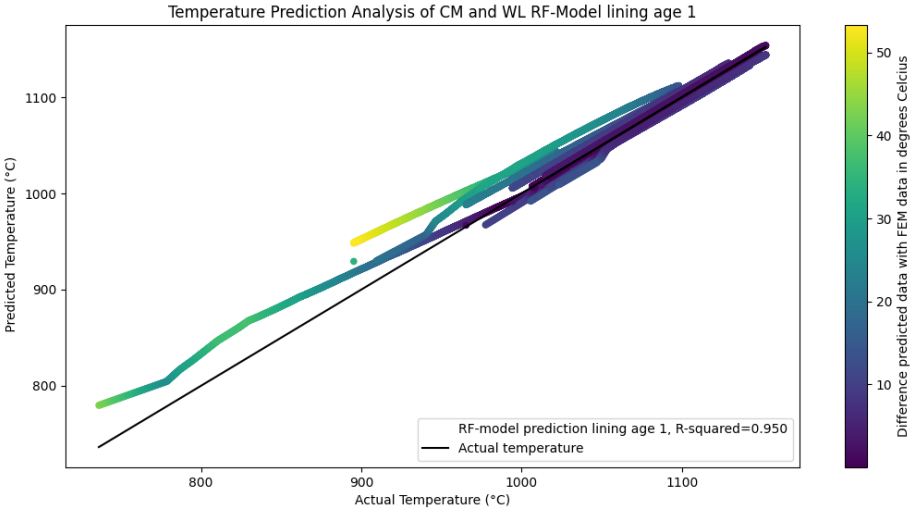


Figure 5.5: R^2 of test data for HM NN-SVDmodel. RMSE= 2.439 of the whole test dataset, where RMSE of the test indices ranges from 0.137 to 7.873.

For the HM model of RF-SVDmodel, the R^2 values range from 0.65 to 0.99, indicating a moderate to strong degree of correlation between the predicted and actual values. However, several values do not meet the criteria that R^2 should be close to 1 for all test datasets, suggesting that there are instances where the model struggles to accurately capture the underlying relationships within the data. On the other hand, the R^2 values achieved by the HM model NN-SVDmodel range from 0.95 to 0.999, reflecting a consistently high degree of correlation between the predicted and actual values. This range indicates that the model accurately captures the underlying patterns and relationships within the dataset.

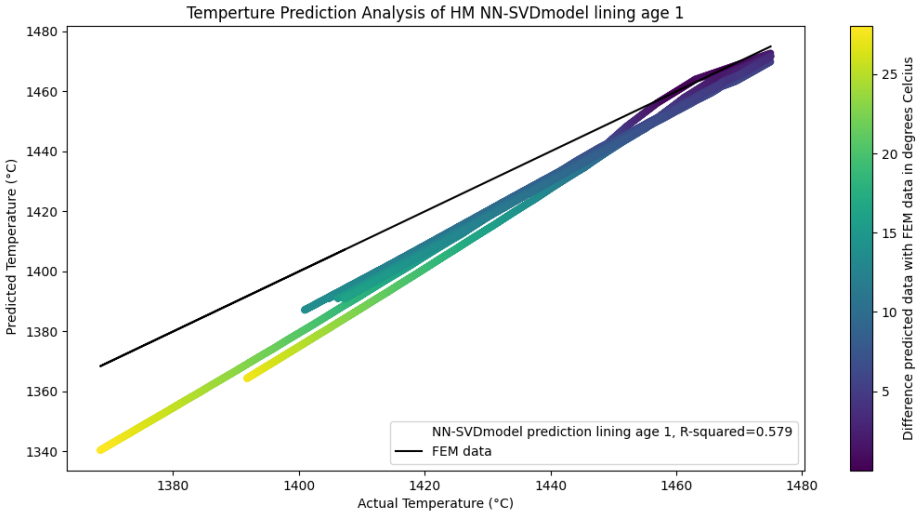
When examining the RMSE metrics for both models, the HM model of RF-SVDmodel demonstrates

an overall RMSE of 4.017 degrees Celsius for the entire test dataset. The range of RMSE values across different test indices varies from 0.040 to 18.65 degrees Celsius. In contrast, the HM NN-SVDmodel achieves a lower overall RMSE of 2.439 degrees Celsius for the entire test dataset, with the RMSE of the test indices ranging from 0.137 to 7.873 degrees Celsius. This considerable difference in RMSE values suggests that the HM model of NN-SVDmodel outperforms the HM model of RF-SVDmodel regarding predictive accuracy, with significantly lower errors in temperature predictions across various scenarios. The R^2 results of the test data for the CM and WL model of RF-SVDmodel and NN-SVDmodel can be found in Appendix B.6 and B.4



#	Δ ($C\circ$)
1	-61.599
2	8.390
3	-53.284
4	-13.909
5	-22.988
6	6.593
7	-11.276
8	8.876
9	-0.565
10	4.494
11	-0.392
12	7.735
13	-3.411
14	8.393
15	-3.472

Figure 5.6: The results of the validation dataset of lining age 1 for WL and CM model of RF-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in real-life torpedo operation prediction.



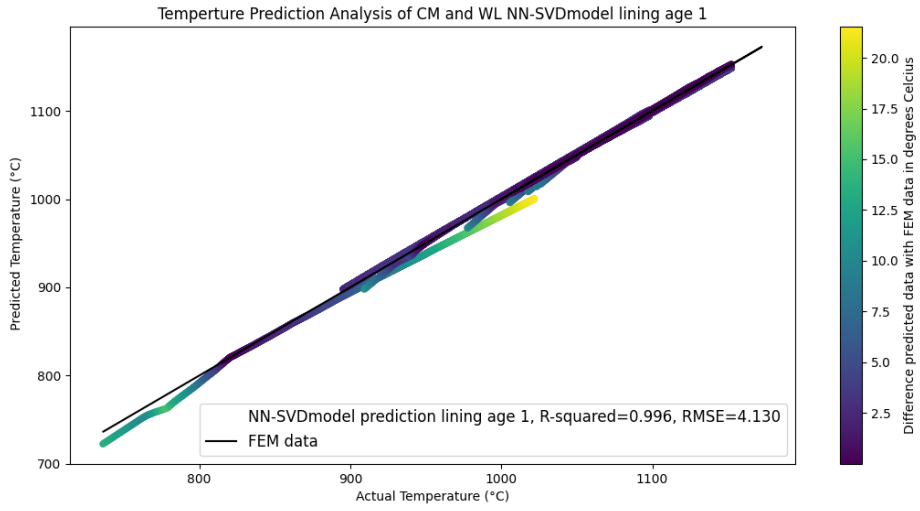
#	Δ ($C\circ$)
1	28.014
2	27.375
3	13.779
4	13.675
5	15.134
6	15.437
7	16.285

Figure 5.7: The results of the validation dataset of lining age 1 for HM model of RF-model.

Figure 5.6 and 5.7 shows the results from the validation dataset for the RF-SVDmodel. These figures compare the predicted temperature values with the actual ones, represented by the solid black line. The legend of the graph shows the R-squared error of the prediction. Additionally, a color bar in

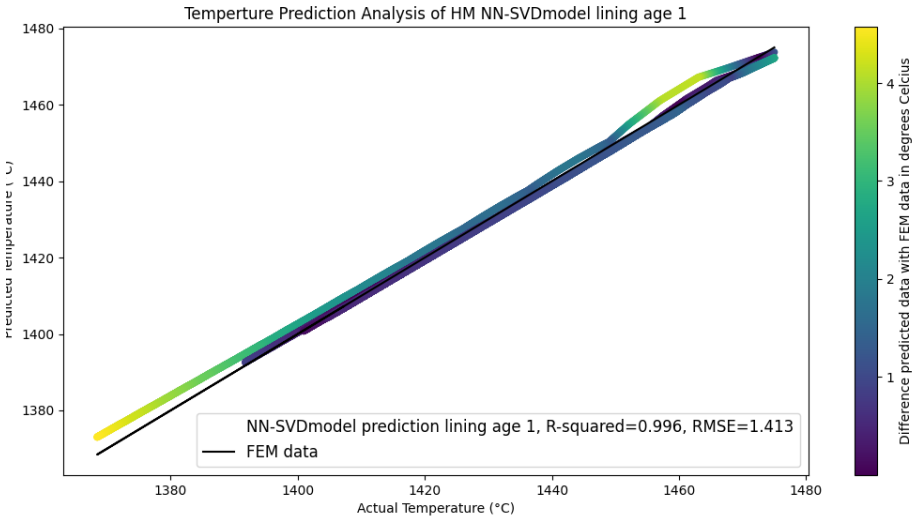
the graph indicates the temperature differences between the predicted and actual values. Furthermore, a table next to the graph summarizes the temperature differences for the last temperature value of each prediction of the WL and CM models.

The results in Figure 5.6 show a high R-squared for this validation set, indicating a good overall fit of the model to the data. However, despite the high R-squared, temperature differences can be significant, reaching almost 50 degrees in some cases. The table next to the graph illustrates the variability in temperature differences, particularly at the end of each prediction cycle. This end temperature serves as a new input value for the prediction cycle when doing a real-life operation prediction. Therefore, if this value varies significantly from the actual temperature, the error will propagate throughout the entire prediction cycle. When this happens, the model can break, and the accuracy of the real-life torpedo operation prediction can be way off.



#	$\Delta\ (C\circ)$
1	-4.48
2	21.517
3	-2.642
4	3.717
5	-2.296
6	2.273
7	-1.295
8	2.19
9	-0.445
10	-1.37
11	0.158
12	1.02
13	0.436
14	0.701
15	0.331

Figure 5.8: The results of the validation dataset of lining age 1 for WL and CM model of NN-SVDmodel. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in real-life torpedo operation prediction.



#	$\Delta\ (C\circ)$
1	-4.571
2	-0.840
3	0.156
4	0.411
5	0.479
6	0.505
7	0.507

Figure 5.9: The results of the validation dataset of lining age 1 for HM model of NN-SVDmodel.

Figures 5.8 and 5.9 show the results obtained from the validation dataset for the NN-SVD model. In this evaluation, the model was tested with the validation dataset of lining age 1, which was not part of its training data. The graphs compare the predicted temperature values with the actual ones, represented by the solid black line. The legend within each graph provides information on the R^2 error of the predictions. Additionally, a colour bar on the right side of each graph indicates the temperature difference between the predicted and actual values. Furthermore, a table alongside the graphs summarizes the temperature differences for the last temperature value of each prediction cycle generated by the NN-SVD model.

Figures 5.8 and 5.9 display a high R^2 value for this validation set. Even when examining the table on the right, the predicted end temperatures closely match the actual temperatures very similar to Figures 5.8 and 5.9. The predictions closely follow the linear FEM data line. Therefore, the real-life torpedo operation prediction could be promising due to the models' high accuracy. However, Figures 5.8 and 5.9 have the same issue as Figures 5.8 and 5.9 due to the lower accuracy in the first prediction in the cycle.

5.2.2. NN-model and RNN-model

The R^2 curves of the test data results of the HM model can be used to evaluate the performance of both models on unseen data. Figure 5.10 illustrates the R^2 values of the test data for the NN-model, and Figure 5.11, shows the R^2 of the test data for the RNN-model. For NN-model, R^2 values ranging between 0.95 and 0.999 is obtained. Additionally, the RMSE of the test indices ranges from 0.249 to 4.626 degrees Celcius. For RNN-model, R^2 values ranging between 0.825 and 0.999 is achieved. The RMSE of the test indices is ranging between 0.327 to 9.465 degrees Celsius.

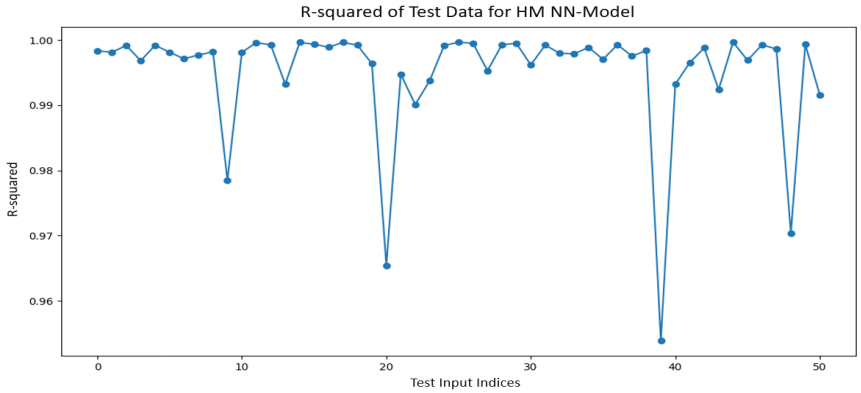


Figure 5.10: R^2 of test data for HM NN-model. RMSE= 1.417 of the whole test dataset, where RMSE of the test indices ranges from 0.249 to 4.626.

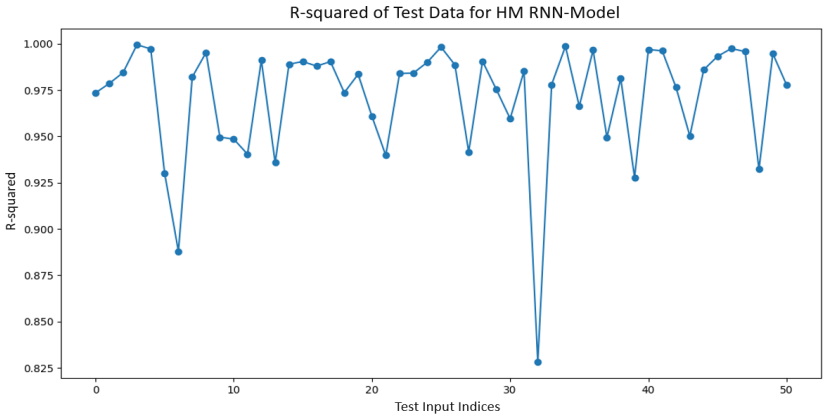
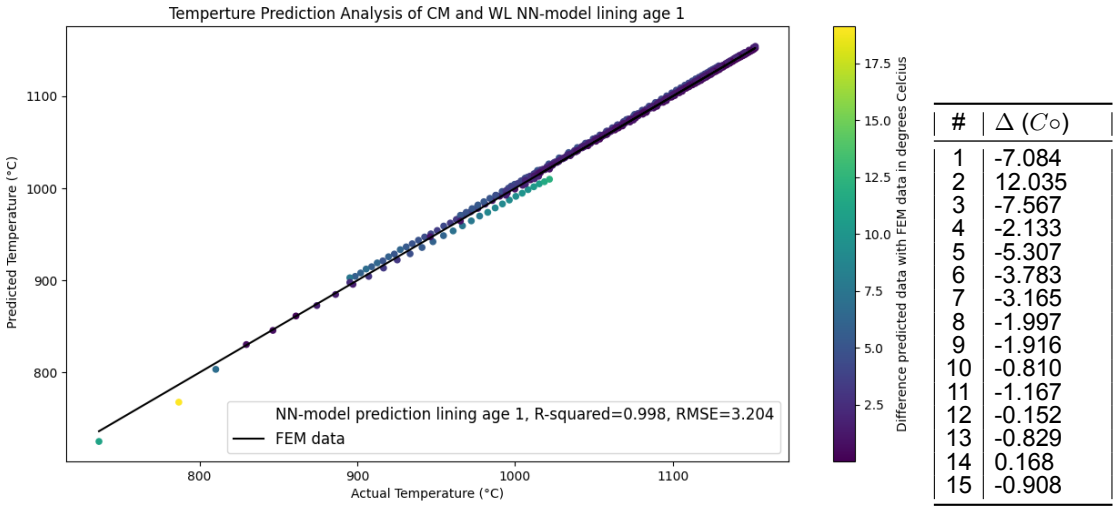


Figure 5.11: R^2 of test data for HM RNN-model. RMSE= 3.677 degrees Celsius of the whole test dataset, where RMSE of the test indices ranges from 0.327 to 9.465 degrees Celsius.

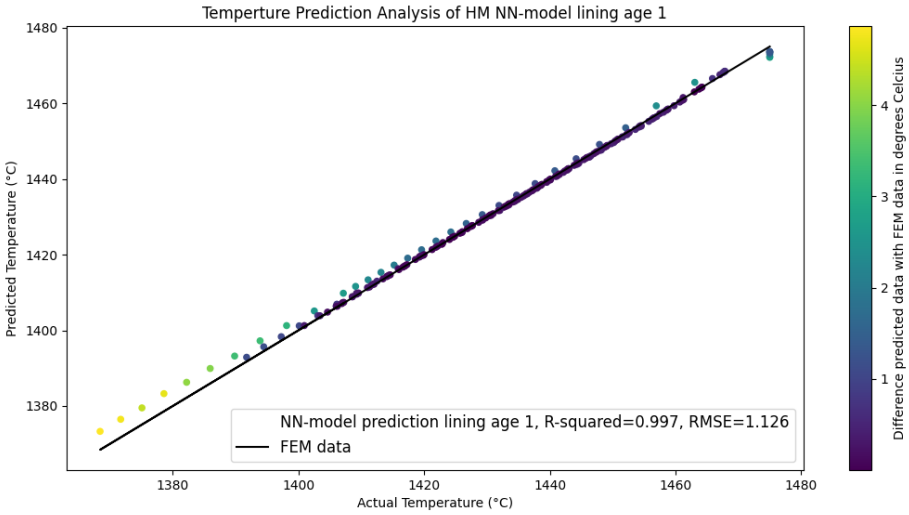
The R^2 values achieved by the NN-model range between 0.95 and 0.999. This high R^2 range indicates that there is a high degree of correlation between the predicted and the actual values. These findings suggest that the model accurately captures the underlying relationships within the data. Furthermore, the RNN-model also achieved a high R^2 value range, indicating that the model is also accurately capturing the relationships of the underlying patterns.

In addition to R^2, the RMSE metrics of the test data were obtained for both models to provide further insights into their performance. For the RNN-model the RMSE of some test predictions can get as low as 0.33 degrees Celsius, whereas the overall RMSE of the test data is around 3.7 degrees Celsius, indicating a satisfactory level of predictive accuracy. However, instances where the RMSE reaches 9.5 degrees Celsius suggest the model may struggle with accurate predictions in specific scenarios. Compared to the NN-model, the overall RMSE of the test data is around 1.417, representing a difference of nearly 2 degrees Celsius compared to the RNN-model. Although the minimum RMSE of both models is similar, the NN-model outperforms the RNN-model significantly in terms of maximum RMSE, with a difference of almost 5 degrees Celsius. The R^2 results of the test data for the CM and WL model of RNN-Dmodel and NN-model can be found in Appendix B.5 and B.4



#	Δ ($C\circ$)
1	-7.084
2	12.035
3	-7.567
4	-2.133
5	-5.307
6	-3.783
7	-3.165
8	-1.997
9	-1.916
10	-0.810
11	-1.167
12	-0.152
13	-0.829
14	0.168
15	-0.908

Figure 5.12: The results of the validation dataset of lining age 1 for WL and CM model of NN-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in real-life torpedo operation prediction.

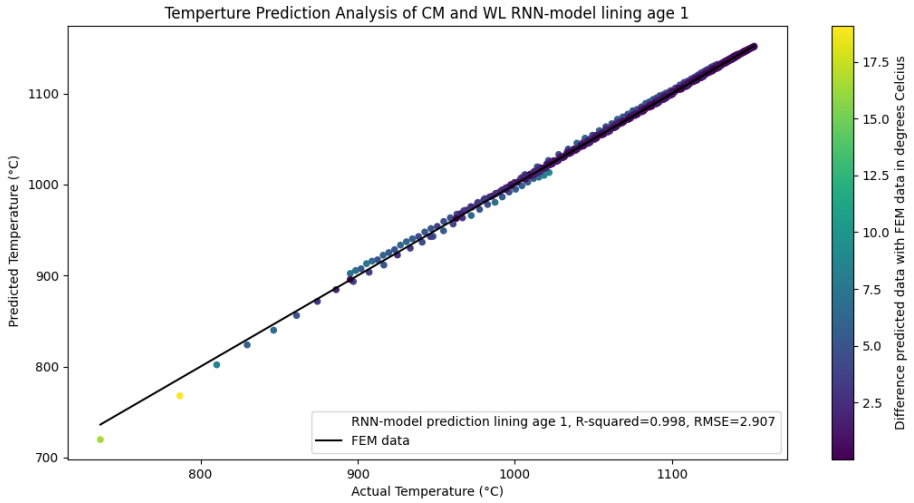


#	Δ (C∘)
1	-4.864
2	-1.133
3	-0.330
4	-0.189
5	-0.177
6	-0.178
7	-0.189

Figure 5.13: The results of the validation dataset of lining age 1 for HM model of NN-model.

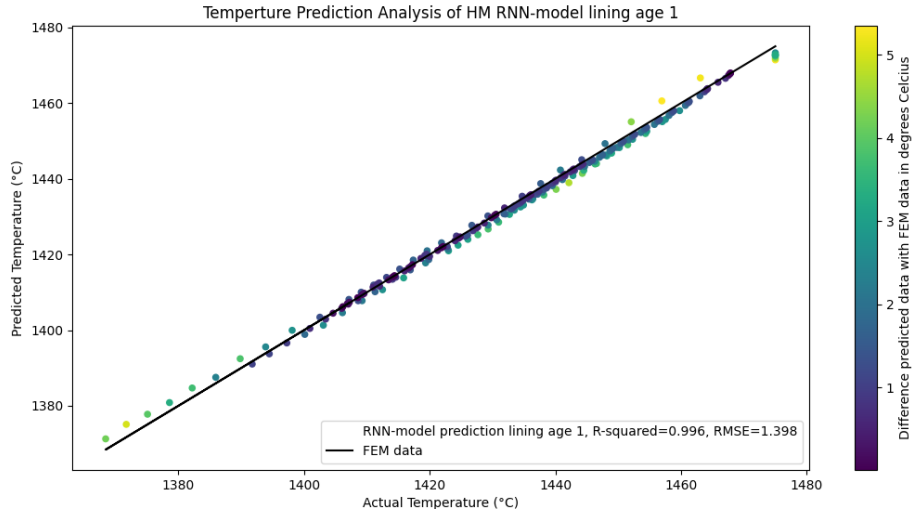
Figure 5.12 and Figure 5.13 presents the results obtained from the validation dataset for the NN-model. During this evaluation, the model was tested with the validation dataset of lining age 1, which was not part of its training data. The graph in Figure 5.12 and 5.13 compares the predicted temperature values with the actual ones, represented by the solid black line. The legend within the graph provides insight into the R^2 error of the prediction. Additionally, a colour bar on the right of the graph indicates the temperature difference between the predicted and actual values. Furthermore, a table alongside the graph summarizes the temperature differences for the last temperature value of each prediction cycle generated by the HM, WL and CM models.

Figure 5.12 and Figure 5.13 display a high R^2 value for this validation set. Even when examining the table on the right, the predicted end temperatures closely match the actual temperatures. Furthermore, the scatter points of the predictions match the FEM data line indicating that the model makes accurate predictions and has learned the underlining patterns of the data to a high degree. This high accuracy is promising for real-life torpedo operation predictions. However, the initial six predictions in figure 5.12 and the initial two predictions in Figure 5.13 exhibit slightly lower accuracy compared to subsequent predictions, due to the model's training process. The model is less trained on the initial temperature values, thus explaining the inaccuracy observed in these initial predictions.



#	$\Delta\ (C\circ)$
1	-1.694
2	8.689
3	-7.037
4	-3.420
5	-2.567
6	-2.901
7	-0.424
8	-1.021
9	0.740
10	0.107
11	1.446
12	0.708
13	1.764
14	1.004
15	1.676

Figure 5.14: The results of the validation dataset of lining age 1 for WL and CM model of RNN-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in real-life torpedo operation prediction.



#	$\Delta\ (C\circ)$
1	-2.844
2	0.716
3	0.479
4	0.150
5	-0.020
6	-0.099
7	-0.146

Figure 5.15: The results of the validation dataset of lining age 1 for HM model of RNN-model.

Figures 5.14 and 5.15 show the results obtained from the validation dataset of lining age 1 for the RNN-model. This validation dataset was not part of the training data. The graphs compare the predicted temperature with the actual ones, represented by the solid black line. The legend within both graphs provides information on the R^2 error and the RMSE of the predictions. Furthermore, a color bar represents the temperature difference between the predicted and actual values. A table on the right of the graphs summarizes the temperature differences for the last temperature value of each prediction cycle generated by the RNN-model.

Figures 5.14 and 5.15 display a high R^2 value and a relatively low RMSE for this validations set. The tables in both figures indicate that the final temperature of each prediction does not match as closely to the actual temperature in the first predictions of the cycle as they do in the final ones. The model is trained less on these initial temperature values, which could explain the observed inaccuracy.

5.3. Prediction Performance Evaluation and Comparison of Models

We only know the first input value during the real-life torpedo operation prediction. This section evaluates the results of the real-life torpedo operation prediction of the RF-SVDmodel, NN-SVDmodel, NN-model and RNN-model. The real-life torpedo operation prediction is done on the three separate validation datasets. Different lining ages distinguish the validation sets: 1, 100, and 400. Even the lining age 100 is a new input value for the model because the models are only trained on lining ages 1 and 400 as input.

5.3.1. Random forest model (RF-SVDmodel)

In this section, the results of real-life torpedo operation prediction for the RF-SVDmodel are evaluated. The model is evaluated on three different validation sets where the results are plotted and evaluation metrics RMSE and mean R^2 are calculated.

Table 5.2: The R^2 and RMSE on the validation datasets for WL, CM and HM model of RF-SVDmodel

RF-SVDmodel		
WL and CM model validation	R^2	RMSE [°C]
Lining age 1	0.935	17.265
Lining age 400	0.920	19.101
Lining age 100	0.949	15.046
HM model validation	R^2	RMSE [°C]
Lining age 1	0.713	1.840
Lining age 400	0.749	19.101
Lining age 100	0.781	10.283

Figure 5.16 illustrates the results for the WL and CM models on these validation datasets for RF-SVDmodel and the Ansys Twinbuilder. Figure 5.17 presents the same results for the HM model. In both graphs, green points represent the predictions of the RF-SVDmodel, while red points represent the prediction of the Ansys Twinbuilder. The graphs compare the predicted temperature values with the actual ones, represented by the solid black line. Table 5.2 gives the RMSE and R^2 values of the validations dataset prediction of the RF-SVDmodel.

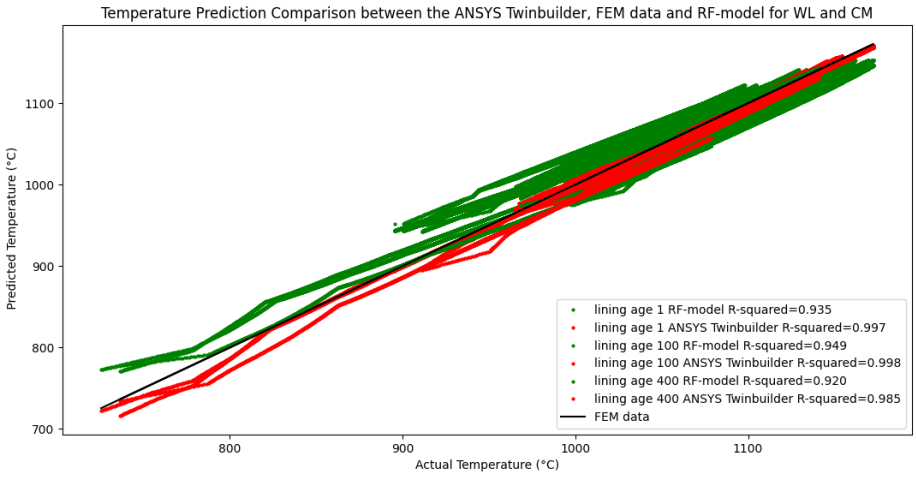


Figure 5.16: Real-life torpedo operation prediction of WL and CM model of RF-model. Green is the RF-model prediction, and red is the AnsysTwinbuilder.

The R^2 values of the Random Forest model across all three validation sets and for WL and CM are relatively high, indicating reasonable predictive accuracy. However, the CM and WL model performance

in terms of RMSE is much higher, indicating some predictive inaccuracy within these models. However, for HM, the R^2 value is 0.715, which, although not drastically low, is not close to one and, therefore, not the desired accuracy level. This difference is not desired, in particularly when considering the importance of the HM model in the final decision-making process for selecting the torpedo car, which aims to minimize heat loss in the hot metal. Furthermore, the RMSE of the HM model is relatively good for the lining age 1 validation, but for the lining ages 100 and 400 validations, the RMSE is significantly high, which indicates an inconsistency in predictive accuracy and an overall inaccurate predictive capability of this model.

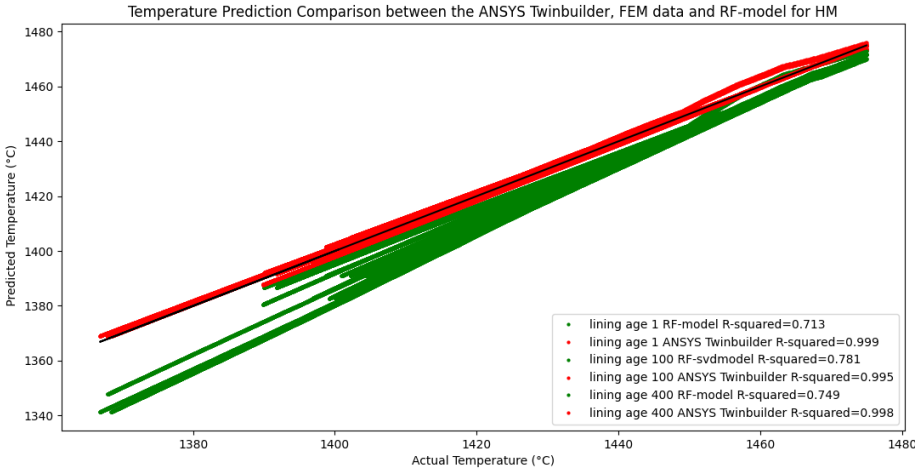


Figure 5.17: Real-life torpedo operation prediction of HM model of RF-model. Green is the RF-model prediction, and red is the AnsysTwinbuilder.

5.3.2. Neural network model with SVD (NN-SVDmodel)

In this section, the results of real-life torpedo operation prediction for the NN-SVDmodel are evaluated. The model is evaluated on three different validation sets where the results are plotted and evaluation metrics RMSE and mean R^2 are calculated.

Table 5.3: The R^2 and RMSE on the validation datasets for WL, CM, and HM models of NN-SVDmodel

NN-SVDmodel		
WL and CM model validation	R^2	RMSE [°C]
Lining age 1	0.983	8.731
Lining age 400	-0.762	89.418
Lining age 100	0.337	54.225
HM model validation	R^2	RMSE [°C]
Lining age 1	0.993	8.731
Lining age 400	0.970	3.827
Lining age 100	0.918	4.697

Figure 5.18 and Figure 5.19 show the results of the comparison of NN-SVDmodel with Ansys Twinbuilder and the FEM data. In both graphs, green points represent the predictions of the NN-model, while red points represent the prediction of the Ansys Twinbuilder. The graphs compare the predicted temperature values with the actual ones, represented by the solid black line. The R^2 values for each validation set can be found in the legend of the respective graph. Table 5.3 gives the RMSE and R^2 values of the validations dataset prediction of the RF-SVDmodel.

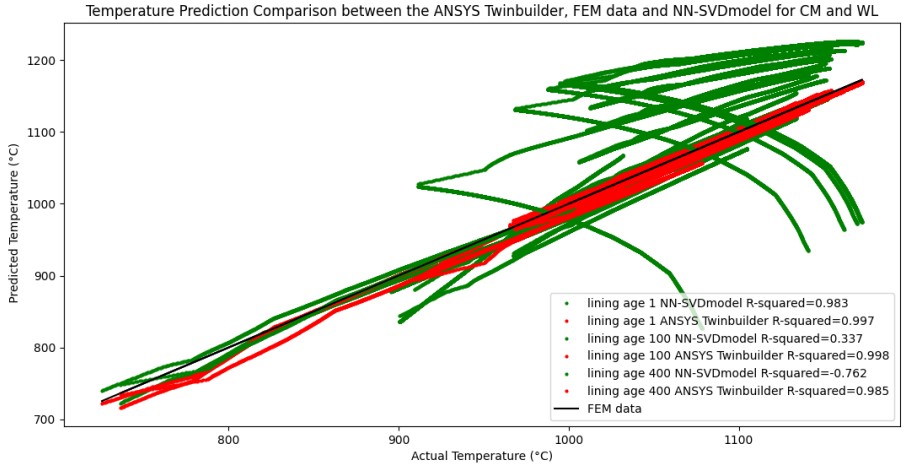


Figure 5.18: Real-life torpedo operation prediction of WL and CM model of NN-SVDmodel. Green is the NN-SVDmodel prediction and red is the AnsysTwinbuilder.

NN-SVDmodel has a high R^2 value of 0.935 for validation dataset of lining age 1. However, its performance in terms of R^2 is significantly low for the validation datasets of lining age 100 and 400, which were -0.762 and 0.33, respectively. Furthermore, for the validation dataset of lining age 100 and 400, the RMSE values are also significantly higher. NN-SVDmodel does not give accurate CM and WL model predictions when running a real-life torpedo operation prediction. In Figure 5.18, we can see the NN-SVDmodel's predictions deviate drastically from the linear line representing the FEM data. This confirms the model's inaccuracy and even looks like a breakdown within the model's predictive capabilities. This breakdown may be caused by the predictions serving as inputs for the subsequent data points being too far off. The model may have difficulties recognizing these initial temperatures of the refractory. In those cases, the model can create inaccurate predictions, and this propagates through the whole cycle.

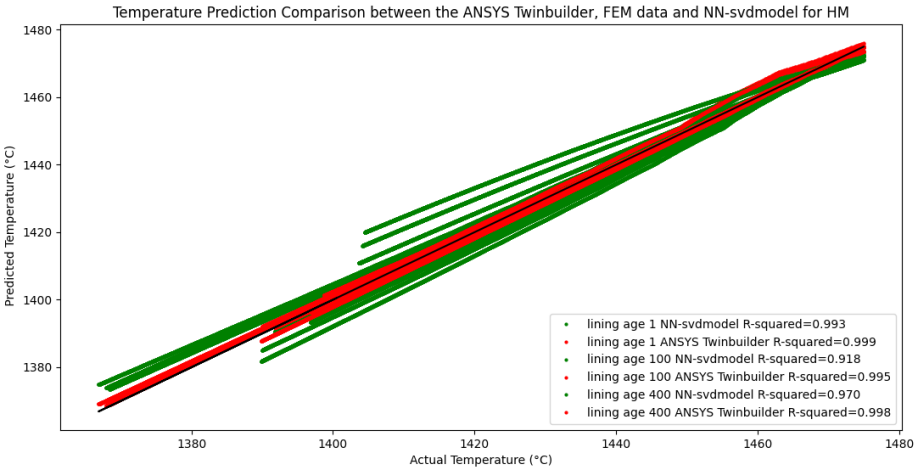


Figure 5.19: Real-life torpedo operation prediction of HM model of NN-SVDmodel. Green is the NN-SVDmodel prediction and red is the AnsysTwinbuilder.

Despite the NN-SVDmodel breaking and providing inaccurate input values, the HM model of the NN-SVDmodel has excellent R^2 values across all validation set higher than 0.918. The RMSE has mixed values across the validation datasets of the HM model. Meanwhile, for lining ages 1 and 100, the RMSE is relatively high. This indicates an inconsistency in the predictive accuracy of the HM model of the NN-SVDmodel. The high R^2 values in the HM models can be explained by the fact that changes in temperature within the refractory material do not significantly influence the temperature of the hot metal

as much. They are resulting, that an inaccuracies in the WL and CM models do not impact the HM model to the same extent. Furthermore, while the NN-SVDmodel becomes inaccurate for lining ages 100 and 400, the NN model does not. The NN-SVDmodel predicts using Singular Value Decomposition (SVD) vectors instead of actual temperature values, unlike the NN-model. This distinction could explain why the NN-SVDmodel struggles to generalize well in all validation datasets. In addition to the potential loss of information, the model could face challenges in capturing the nonlinear relationships in temperature data when using SVD vectors. Actual temperature values contain nonlinear patterns and fluctuations that directly inform the model about the system's behavior. However, when these values are transformed into SVD vectors, the model may encounter difficulty learning these nonlinearities.

5.3.3. Neural network model (NN-model)

In this section, the results of real-life torpedo operation prediction for the NN-model are evaluated. The model is evaluated on three different validation sets where the results are plotted and evaluation metrics RMSE and mean R^2 are calculated.

Table 5.4: The R^2 and RMSE on the validation datasets for WL, CM, and HM models of NN-model

NN-model		
WL and CM model validation	R^2	RMSE [°C]
Lining age 1	0.994	5.271
Lining age 400	0.991	6.698
Lining age 100	0.995	4.697
HM model validation	R^2	RMSE [°C]
Lining age 1	0.996	1.494
Lining age 400	0.997	1.271
Lining age 100	0.993	1.886

Figure 5.20 illustrates the results for the WL and CM models on these validation datasets for NN-model and the Ansys Twinbuilder. Figure 5.21 presents the same results for the HM model. In both graphs, green points represent the predictions of the NN-model, while red points represent the prediction of the Ansys Twinbuilder. The graphs compare the predicted temperature values with the actual ones, represented by the solid black line. The R^2 values for each validation set can be found in the legend of the respective graph. Table 5.4 gives the RMSE and R^2 values of the validations dataset prediction of the RF-SVDmodel.

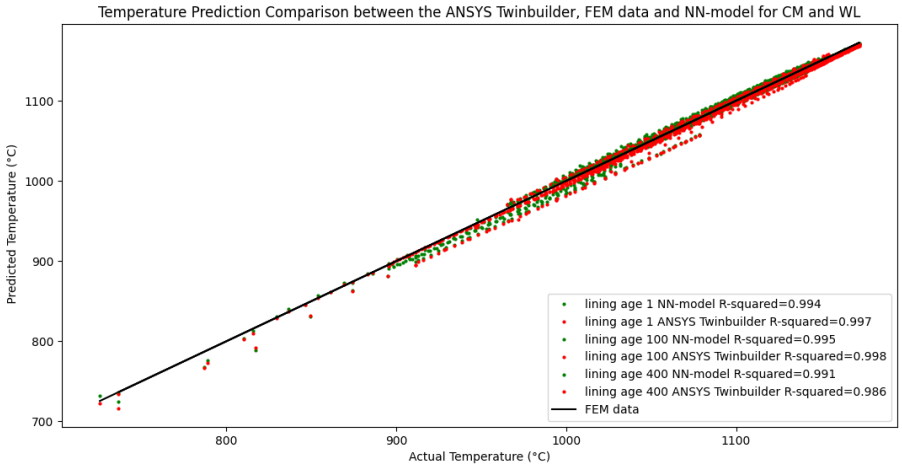


Figure 5.20: Real-life torpedo operation prediction of WL and CM model of NN-model. Green is the NN-model prediction, and red is the AnsysTwinbuilder.

In Table 5.4, we see the performance of the NN-model for the validation set of lining age 1, lining age 100 and lining age 400. As can be seen for all three validation sets the model has a high R^2 error ranging from 0.993 to 0.997 for the WL, CM and HM model. Furthermore, the RMSE values of each validation set of all models are also relatively low, which means that the prediction of this model is accurate even in the case of real-life torpedo operation prediction. Additionally, the R^2 and RMSE values of the NN-model and Ansys Twinbuilder are closely related, indicating that the NN-model performs comparably to Ansys Twinbuilder.

Additionally, the NN-model's has a consistent high R^2 values across different validations. This means the model can accurately predict outcomes across various scenarios. This confirms the model's effectiveness in capturing complex relationships.

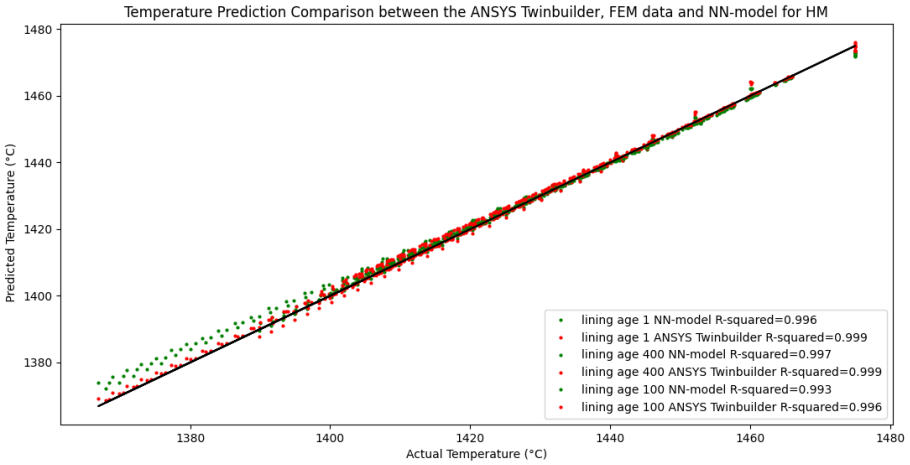


Figure 5.21: Real-life torpedo operation prediction of HM model of NN-model. Green is the NN-model prediction, and red is the AnsysTwinbuilder.

5.3.4. Recurrent neural network model (RNN-model)

In this section the results of real-life torpedo operation prediction for the RNN-model are evaluated. The model is evaluated on three different validation set where the results are plotted and evaluation metrices RMSE and mean R^2 are calculated.

Table 5.5: The R^2 and RMSE on the validation datasets for WL, CM, and HM models of RNN-model

RNN-model		
WL and CM model validation	R^2	RMSE [°C]
Lining age 1	0.921	19.412
Lining age 400	0.935	17.797
Lining age 100	0.944	16.11
HM model validation	R^2	RMSE [°C]
Lining age 1	0.89	2.308
Lining age 400	0.992	2.016
Lining age 100	0.993	1.829

Figure 5.22 presents the results for the WL and CM models on all three validation datasets for the RNN-model. Figure 5.23 illustrates the results of the validation datasets for the HM model of the RNN-model. These figures, the green points represent the predictions of the RNN-model and the red points represent the predictions of the Ansys Twinbuilder. The solid black line represents the actual temperature. The R^2 value for each validation dataset can be found in the legend of the figure. Table 5.5 gives the RMSE and R^2 values of the validations dataset prediction of the RF-SVDmodel.

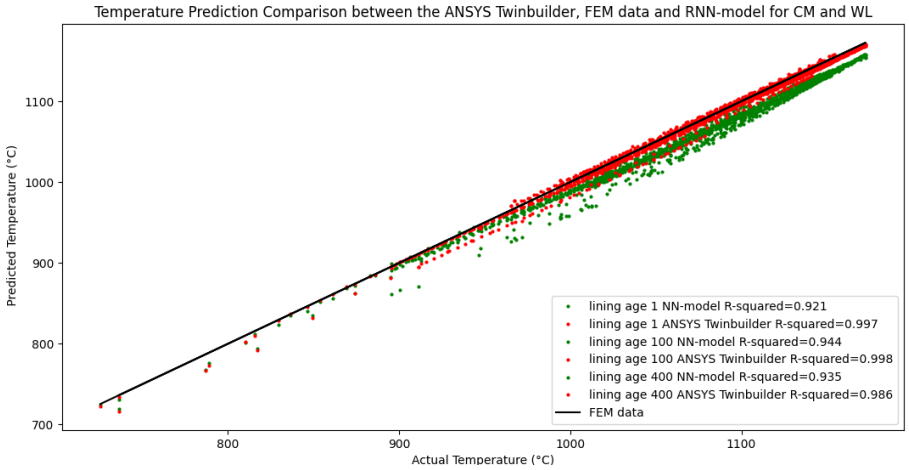


Figure 5.22: Real-life torpedo operation prediction of WL and CM model of RNN-model. Green is the RNN-model prediction, and red is the AnsysTwinbuilder.

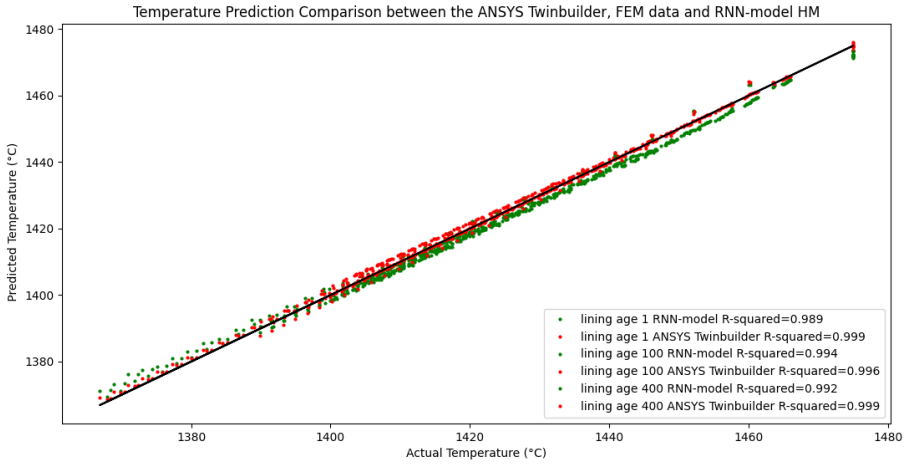


Figure 5.23: Real-life torpedo operation prediction of HM model of RNN-model. Green is the RNN-model prediction and red is the AnsysTwinbuilder.

RNN-model shows consistently good R^2 values between 0.89 and 0.993 across all validation sets and for all models. However, the RMSE values of the CM and WL models are around 18 degrees Celsius, which is relatively high and indicates a much lower predictive accuracy across all validation datasets. Even though the HM model has high R^2 values, it contains low RMSE values of around 2 degrees Celsius across different validation datasets. They indicate that the RNN-model has relatively high predictive accuracy on the HM model.

5.3.5. Ansys Twinbuilder

The performance of the Ansys Twinbuilder based on the three validation sets is presented in the table below. These values are used to compare the new models with the existing digital twin and compare the performance of the new models to the existing one.

Table 5.6: The R^2 and RMSE on the validation datasets for WL, CM, and HM models of AnsysTwinbuilder

AnsysTwinbuilder		
WL and CM model validation	R^2	RMSE [°C]
Lining age 1	0.997	3.654
Lining age 400	0.985	8.308
Lining age 100	0.998	3.213
HM model validation	R^2	RMSE [°C]
Lining age 1	0.999	0.571
Lining age 400	0.998	0.876
Lining age 100	0.995	1.502

6

Conclusion

The objective of this research was to investigate the following question: Can a reduced-order model combined with a machine learning technique be used to develop a feasible and accurate digital twin for simulating the thermal management of torpedo ladle cars? This investigation was conducted within the specific context of thermal management for torpedo ladle cars at Tata Steel. This research successfully developed and evaluated four distinct digital twin models: RF-SVDmodel, NN-SVDmodel, NN-model, and RNN-model, to assess their performance. The performance of the models was tested on their generalization capabilities, their performance on the test and validation dataset, and their predictive performances on real-life torpedo operations.

Currently, Tata Steel IJmuiden has a digital twin of the torpedo ladle car created by commercialized software named Ansys Twinbuilder. The commercialized Ansys Twinbuilder has several drawbacks. This includes limited access to detailed model information, operating as a black box, restricted flexibility and adaptability, dependency on external support, and integration challenges with existing systems.

The digital twins that are developed consist of three prediction models: one predicting the refractory temperature of the torpedo ladle car when it is empty (CM model), another for when it is filled with hot metal (WL model), and a third for estimating the temperature of the hot metal inside the torpedo ladle car (HM model). Four digital twin models were constructed to perform these predictions. The RF-SVDmodel and NN-SVDmodel employ singular value decomposition to reduce the data to vectors. In the RF-SVDmodel, the prediction is done by the random forest machine learning technique, while in the NN-SVDmodel, the Neural Network technique is applied. In the NN-model and RNN-model, the raw data is used without data reduction. In the NN-model, the neural network machine learning technique is used for predictions, while in the RNN-model the recurrent neural network technique is used.

First, the generalization capabilities of the models were evaluated. For the RF-model a k-fold cross-validation was used, where the HM model showed good generalizing capabilities based on its low variance of R^2 across different folds. However, the CM and WL model has a slightly higher variance in the R^2 over the different folds and a higher RMSE, indicating that it does not generalize as well as the HM model and gives less accurate predictions. For the NN-SVDmodel, NN-model and RNN-model, the generalization abilities were evaluated using a learning curve. For the NN-SVDmodel, NN-model, and RNN-model, the training and validation curves in the learning curve align perfectly across the CM, WL, and HM models, suggesting that the model is generalizing the data to a high degree without overfitting.

Secondly, the evaluation of the RF-SVDmodel, NN-SVDmodel, NN-model, and RNN-model revealed varying models' performances on the test dataset and validation datasets. Where the NN-SVDmodel and the NN-model performed best on the test dataset according to the performance criteria. Where NN-SVDmodel exhibited consistently higher accuracy, with R^2 values ranging from 0.95 to 0.999 and a RMSE of 2.439 degrees Celsius ($^\circ$C). NN-model exhibits a similar accuracy with R^2 values ranging from 0.96 to 0.999 and an even better RMSE of 1.417 degrees Celsius.

On the validation dataset, all models showcased a R^2, indicating a good overall fit of the predicted

temperature to the actual temperature. However, looking at δ°C of the end temperature of each prediction, some models perform better than others. Where for the RNN-model, NN-model and NN-SVDmodel these differences where for most prediction under one degree Celsius for the WL, CM and HM model. For the RF-SVDmodel these were significantly higher.

Thirdly, the RF-SVDmodel, NN-SVDmodel, NN-model, and RNN-model were evaluated on their performance of real-life torpedo operation predictions for three different cases (lining age 1, lining age 100, and lining age 400). Among these models, the NN-model demonstrates consistently high R^2 values ranging from 0.993 to 0.997 across all validation sets for WL, CM, and HM models. Its low RMSE values indicate accurate predictions even in real-life torpedo operation prediction. In contrast, the RF-SVDmodel and NN-SVDmodel show mixed results, exhibiting relatively close to one R^2 values but inconsistent performance in RMSE, particularly in the HM model. The NN-SVDmodel faces breakdowns in predictive capabilities, which is especially evident in CM and WL models during full cycle predictions. Meanwhile, the RNN-model exhibits a moderate performance, with relatively high R^2 values but higher RMSE values for CM and WL models, indicating less robust predictions. Comparing these models to the existing Ansys Twinbuilder, it is evident that the NN-model closely rivals its performance, with similar R^2 values and RMSE values.

In conclusion, this research demonstrates the potential of combining reduced-order models with machine-learning techniques to develop a feasible and accurate digital twin for the thermal management of torpedo ladle cars. Through the evaluation of four distinct digital twin models, it is clear that the NN-model would be the best model to use as digital twin for the torpedo car, exhibiting robust performances, high generalization abilities, and competitive accuracy compared to the existing Ansys Twinbuilder model. The model can accurately predict temperature profiles for diverse torpedo ladle cars, accounting for variables like lining age and initial refractory temperature. This results in the possibility of choosing the most favourable torpedo ladle car to minimize heat loss in the hot metal. Being able to reduce the hot metal temperature heat loss by one degree Celsius can save around one million euros. Furthermore, the commercialized Ansys Twinbuilder currently used at Tata Steel has several drawbacks, in contrast to the NN-model, which offers flexibility, transparency instead of a black box, and access to detailed model information.

6.1. Future research

This section will investigate the potential applications of this model for further research at Tata Steel IJmuiden, as well as examine its deployment possibilities within the plant.

The digital twin's accurate hot metal temperature predictions can be used to incorporate reinforcement learning into the Tata Steels plant's operations. By using reinforcement learning, we can effectively use these predictions in making decisions about choosing the torpedo car that would cause the minimum heat loss in the hot metal for a given scenario. The system intelligently selects the optimal car through reinforcement learning algorithms and ensures its deployment within the plant. This process is essential for selecting the most favourable torpedo cars, enabling practical decision-making in plant operations.

Furthermore, Tata Steel IJmuiden has temperature profile data of more processed than only that of the torpedo ladle car. For instance, the steel ladles employed in Tata Steel IJmuiden exhibit various temperature-time profiles. The thermodynamics of the steel ladle system is different from that of the torpedo ladle car. The steps involved in making the digital twin of the torpedo ladle car can be similar to those for the creation of a digital twin of the steel ladle. However, the digital twin of the steel ladles would differ in hyperparameters due to different systems and the possibility that instead of three different models to create the digital twin, less or more could be necessary. The methodology of the NN-model creation of the torpedo ladle car can also be employed for the steel ladle case. During the early stages of testing our steel ladle model, this approach has shown promising results and delivered accurate predictions. This encouraging success suggests exciting possibilities for refining the model specifically for the steel ladle application. Additionally, it indicates the potential to expand the use of the NN-model methodology across multiple processes at Tata Steel IJmuiden.

References

[1] Bożena Gajdzik and Radosław Wolniak. “Transitioning of Steel Producers to the Steelworks”. In: *Energies* 14.14 (July 2021), p. 4109. DOI: 10.3390/en14144109. URL: https://doi.org/10.3390/en14144109.

[2] Lauri Holappa. “A General Vision for Reduction of Energy Consumption and CO2 Emissions from the Steel Industry”. In: *Metals* 10.9 (Aug. 2020), p. 1117. DOI: 10.3390/met10091117. URL: https://doi.org/10.3390/met10091117.

[3] M. Garcia Campos et al. “Digital Twin of a Torpedo Ladle: The Simulated Future”. In: *Tata Steel Nederland* (Jan. 2023). DOI: 10.33313/387/245. URL: https://doi.org/10.33313/387/245.

[4] Ahindra Ghosh and Amit Chatterjee. *IRON MAKING AND STEELMAKING*. PHI Learning Pvt. Ltd., Feb. 2008, pp. 46, 207–209.

[5] Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational methods for fluid dynamics*. Springer, Aug. 2019.

[6] van Beurden and Put. “Thermal finite-element model of the IJmuiden torpedo ladle”. In: *TATA Steel* ().

[7] David J. Lucia, Philip S. Beran, and Walter A. Silva. “Reduced-order modeling: new approaches for computational physics”. In: *Progress in Aerospace Sciences* 40.1-2 (Feb. 2004), pp. 51–117. DOI: 10.1016/j.paerosci.2003.12.001. URL: https://doi.org/10.1016/j.paerosci.2003.12.001.

[8] Steven L. Brunton and J. Nathan Kutz. *Data-Driven science and engineering*. Cambridge University Press, Feb. 2019, pp. 3–12.

[9] Xiaocan Li, Shuo Wang, and Yinghao Cai. “Tutorial: Complexity analysis of Singular Value Decomposition and its variants”. In: *University of Chinese Academy of Sciences* (June 2019).

[10] P.K. Sadasivan and D. Narayana Dutt. “SVD based technique for noise reduction in electroencephalographic signals”. In: *Signal Processing* 55.2 (Dec. 1996), pp. 179–189. DOI: 10.1016/s0165-1684(96)00129-6. URL: https://doi.org/10.1016/s0165-1684(96)00129-6.

[11] Carla Martin and Mason A. Porter. “The extraordinary SVD”. In: *arXiv (Cornell University)* (Mar. 2011). DOI: 10.48550/arxiv.1103.2338. URL: http://arxiv.org/abs/1103.2338.

[12] James Demmel. “Accurate Singular Value Decompositions of Structured Matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 21.2 (2000), pp. 562–580. DOI: 10.1137/S0895479897328716.

[13] James Demmel et al. “Computing the singular value decomposition with high relative accuracy”. In: *Linear Algebra and its Applications* 299.1-3 (Sept. 1999), pp. 21–80. DOI: 10.1016/s0024-3795(99)00134-2. URL: https://doi.org/10.1016/s0024-3795(99)00134-2.

[14] Jengnan Tzeng. “Split-and-Combine singular value decomposition for Large-Scale matrix”. In: *Journal of Applied Mathematics* 2013 (Jan. 2013), pp. 1–8. DOI: 10.1155/2013/683053. URL: https://doi.org/10.1155/2013/683053.

[15] Jian Xue et al. “Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network”. In: *IEEE Intinternational Conference on Acoustic, Speech and Signal Processing (ICASSP)* (May 2014). DOI: 10.1109/icassp.2014.6854828. URL: https://doi.org/10.1109/icassp.2014.6854828.

[16] Gal Berkooz, Philip Holmes, and John L. Lumley. “The proper orthogonal decomposition in the analysis of turbulent flows”. In: *Annual Review of Fluid Mechanics* 25.1 (Jan. 1993), pp. 539–575. DOI: 10.1146/annurev.fl.25.010193.002543. URL: https://doi.org/10.1146/annurev.fl.25.010193.002543.

[17] Lorenz Pyta, Mathias Hakenberg, and Dirk Abel. “Model Reduction and Control of a Compressible Channel Flow with Combustion”. In: *IFAC Proceedings Volumes* 47.3 (Jan. 2014), pp. 7001–7006. DOI: 10.3182/20140824-6-za-1003.01927. URL: https://doi.org/10.3182/20140824-6-za-1003.01927.

[18] Steven L. Brunton and J. Nathan Kutz. *Data-Driven science and engineering*. Cambridge University Press, Feb. 2019, pp. 440–449.

[19] Sean Mortara, J. C. Slater, and Philip S. Beran. “Analysis of nonlinear aeroelastic panel response using proper orthogonal decomposition”. In: *Journal of Vibration and Acoustics* 126.3 (July 2004), pp. 416–421. DOI: 10.1115/1.1687389. URL: https://doi.org/10.1115/1.1687389.

[20] Muruhan Rathinam and Linda R. Petzold. “A new look at proper orthogonal decomposition”. In: *SIAM Journal on Numerical Analysis* 41.5 (Jan. 2003), pp. 1893–1925. DOI: 10.1137/s0036142901389049. URL: https://doi.org/10.1137/s0036142901389049.

[21] Peter J. Schmid. *Data-driven and operator-based tools for the analysis of turbulent flows*. Jan. 2021. DOI: 10.1016/b978-0-12-820774-1.00012-4. URL: https://doi.org/10.1016/b978-0-12-820774-1.00012-4.

[22] Steven L. Brunton and J. Nathan Kutz. *Data-Driven science and engineering*. Cambridge University Press, Feb. 2019, pp. 274–287.

[23] Peter J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* 656 (July 2010), pp. 5–28. DOI: 10.1017/s0022112010001217. URL: https://doi.org/10.1017/s0022112010001217.

[24] N. Benjamin Erichson et al. “Randomized Dynamic Mode decomposition”. In: *Siam Journal on Applied Dynamical Systems* 18.4 (Jan. 2019), pp. 1867–1891. DOI: 10.1137/18m1215013. URL: https://doi.org/10.1137/18m1215013.

[25] Peter J. Schmid et al. “Applications of the dynamic mode decomposition”. In: *Theoretical and Computational Fluid Dynamics* 25.1-4 (Aug. 2010), pp. 249–259. DOI: 10.1007/s00162-010-0203-9. URL: https://doi.org/10.1007/s00162-010-0203-9.

[26] Wasuta Renkjumnong. *SVD and PCA in Image Processing*. Jan. 2007.

[27] Steven L. Brunton and J. Nathan Kutz. *Data-Driven science and engineering*. Cambridge University Press, Feb. 2019, pp. 24–25.

[28] M. Statheropoulos et al. “Noise reduction of fast, repetitive GC/MS measurements using principal component analysis (PCA)”. In: *Analytica Chimica Acta* 401.1-2 (Nov. 1999), pp. 35–43. DOI: 10.1016/s0003-2670(99)00494-8. URL: https://doi.org/10.1016/s0003-2670(99)00494-8.

[29] Felipe L. Gewers et al. “Principal component analysis”. In: *ACM Computing Surveys* 54.4 (May 2021), pp. 1–34. DOI: 10.1145/3447755. URL: https://doi.org/10.1145/3447755.

[30] Sasan Karamizadeh et al. “An overview of principal component analysis”. In: *Journal of Signal and Information Processing* 04.03 (Jan. 2013), pp. 173–175. DOI: 10.4236/jsip.2013.43b031. URL: https://doi.org/10.4236/jsip.2013.43b031.

[31] In Lee and Yong Jae Shin. “Machine learning for enterprises: Applications, algorithm selection, and challenges”. In: *Business Horizons* 63.2 (Mar. 2020), pp. 157–170. DOI: 10.1016/j.bushor.2019.10.005. URL: https://doi.org/10.1016/j.bushor.2019.10.005.

[32] B Mahesh. “Machine Learning Algorithms - A Review”. In: *International Journal of Science and Research (IJSR)* 1 (Jan. 2020), pp. 381–386. DOI: 10.21275/ART20203995.

[33] Jafar A. Alzubi, Anand Nayyar, and Akshi Kumar. “Machine Learning from Theory to Algorithms: An Overview”. In: *Journal of physics* 1142 (Nov. 2018), p. 012012. DOI: 10.1088/1742-6596/1142/1/012012. URL: https://doi.org/10.1088/1742-6596/1142/1/012012.

[34] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media, Sept. 2019, pp. 1–34.

[35] Keiron O’Shea and Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv (Cornell University)* (Nov. 2015). URL: https://arxiv.org/pdf/1511.08458.pdf.

[36] Renee C. Swischuk et al. “Projection-based model reduction: Formulations for physics-based machine learning”. In: *Computers and Fluids* 179 (Jan. 2019), pp. 704–717. DOI: 10.1016/j.compfluid.2018.07.021. URL: https://doi.org/10.1016/j.compfluid.2018.07.021.

[37] Jacques Bourquin et al. “Advantages of Artificial Neural Networks (ANNs) as alternative modelling technique for data sets showing non-linear relationships using data from a galenical study on a solid dosage form.” In: *European Journal of Pharmaceutical Sciences* 7.1 (Dec. 1998), pp. 5–16. DOI: 10.1016/s0928-0987(97)10028-8. URL: https://doi.org/10.1016/s0928-0987(97)10028-8.

[38] D.J. Livingstone, D.T. Manallack, and I.v. Tetko. “Data modelling with neural networks: Advantages and limitations*”. In: *Journal of Computer-Aided Molecular Design* 11 (1996), pp. 135–142.

[39] Imanol Bilbao and Javier Bilbao. “Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks”. In: *The 8th IEEE International Conference on Intelligent Computing and Information Systems* (Dec. 2017). DOI: 10.1109/intelcis.2017.8260032. URL: https://doi.org/10.1109/intelcis.2017.8260032.

[40] John D. Kelleher, Brian Mac Namee, and Aoife D’Arcy. *Fundamentals of Machine learning for Predictive data analytics: algorithms, worked examples, and case studies*. July 2015. URL: https://dl.acm.org/citation.cfm?id=2815672.

[41] Yanyan et al. “Decision tree methods□ applications for classification and prediction”. In: □□□□□□ 27.2 (Jan. 2015), pp. 130–135. URL: http://www.cqvip.com/QK/97872X/201502/664559388.html.

[42] N. Yuvaraj et al. “Automatic detection of cyberbullying using multi-feature based artificial intelligence with deep decision tree classification”. In: *Computers and Electrical Engineering* 92 (June 2021), p. 107186. DOI: 10.1016/j.compeleceng.2021.107186. URL: https://doi.org/10.1016/j.compeleceng.2021.107186.

[43] Oliver Kramer. *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Vol. 51. Springer, June 2013.

[44] Malti Bansal, Apoorva Goyal, and Apoorva Choudhary. “A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning”. In: *Decision Analytics Journal* 3 (June 2022), p. 100071. DOI: 10.1016/j.dajour.2022.100071. URL: https://doi.org/10.1016/j.dajour.2022.100071.

[45] Mariana Segovia and Joaquin Garcia-Alfaro. “Design, modeling and implementation of digital Twins”. In: *Sensors* 22.14 (July 2022), p. 5396. DOI: 10.3390/s22145396. URL: https://doi.org/10.3390/s22145396.

[46] Min Qi et al. “Machine Learning based Digital Twin Framework for Production Optimization in Petrochemical Industry”. In: *International Journal of Information Management* 49 (Dec. 2019), pp. 502–519. DOI: 10.1016/j.ijinfomgt.2019.05.020. URL: https://doi.org/10.1016/j.ijinfomgt.2019.05.020.

[47] André Hürkamp et al. “Simulation-based digital twin for the manufacturing of thermoplastic composites”. In: *Procedia CIRP* 100 (Jan. 2021), pp. 1–6. DOI: 10.1016/j.procir.2021.05.001. URL: https://doi.org/10.1016/j.procir.2021.05.001.

[48] Hang Gong et al. “An efficient digital twin based on machine learning SVD autoencoder and generalised latent assimilation for nuclear reactor physics”. In: *Annals of Nuclear Energy* 179 (Dec. 2022), p. 109431. DOI: 10.1016/j.anucene.2022.109431. URL: https://doi.org/10.1016/j.anucene.2022.109431.

[49] Pin Wu et al. “A Digital Twin Framework Embedded with POD and Neural Network for Flow Field Monitoring of Push-Plate Kiln”. In: *Future Internet* 15.2 (Jan. 2023), p. 51. DOI: 10.3390/fi15020051. URL: https://doi.org/10.3390/fi15020051.

[50] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights Into Imaging* 9.4 (June 2018), pp. 611–629. DOI: 10.1007/s13244-018-0639-9. URL: https://doi.org/10.1007/s13244-018-0639-9.

[51] Javaid Butt and Vahaj Mohaghegh. “Combining digital twin and machine learning for the fused filament fabrication process”. In: *Metals* 13.1 (Dec. 2022), p. 24. DOI: 10.3390/met13010024. URL: https://doi.org/10.3390/met13010024.

[52] Paul Beurden Van. “On the thermal management of torpedo ladle car logistics at Tata Steel in IJmuiden”. In: *Zenodo (CERN European Organization for Nuclear Research)* (Nov. 2021). DOI: 10.5281/zenodo.5655210. URL: https://zenodo.org/record/5655210.

[53] Alexandre M. Bayen and Timmy Siauw. *Interpolation*. Jan. 1, 2015, pp. 211–223. DOI: 10.1016/b978-0-12-420228-3.00014-2. URL: https://doi.org/10.1016/b978-0-12-420228-3.00014-2.

[54] A. Kayode Coker. *Numerical computation*. Jan. 1995, pp. 1–102. DOI: 10.1016/b978-088415280-4/50002-9. URL: https://doi.org/10.1016/b978-088415280-4/50002-9.

[55] Pardeep Singla, Manoj Duhan, and Sumit Saroha. *Different normalization techniques as data preprocessing for one step ahead forecasting of solar global horizontal irradiance*. Jan. 2022, pp. 209–230. DOI: 10.1016/b978-0-323-90396-7.00004-3. URL: https://doi.org/10.1016/b978-0-323-90396-7.00004-3.

[56] Jason Brownlee. *Better deep learning*. Machine Learning Mastery, Dec. 2018.

[57] Hoss Belyadi and Alireza Haghighat. *Model evaluation*. Jan. 2021, pp. 349–380. DOI: 10.1016/b978-0-12-821929-4.00009-3. URL: https://doi.org/10.1016/b978-0-12-821929-4.00009-3.

[58] Balachandra Kumaraswamy. *Neural networks for data classification*. Jan. 2021, pp. 109–131. DOI: 10.1016/b978-0-12-820601-0.00011-2. URL: https://doi.org/10.1016/b978-0-12-820601-0.00011-2.

[59] Deepak Kumar Sharma et al. *Deep learning applications for disease diagnosis*. Jan. 2022, pp. 31–51. DOI: 10.1016/b978-0-12-824145-5.00005-8. URL: https://doi.org/10.1016/b978-0-12-824145-5.00005-8.

A

Appendix

A.1. Interpolation data

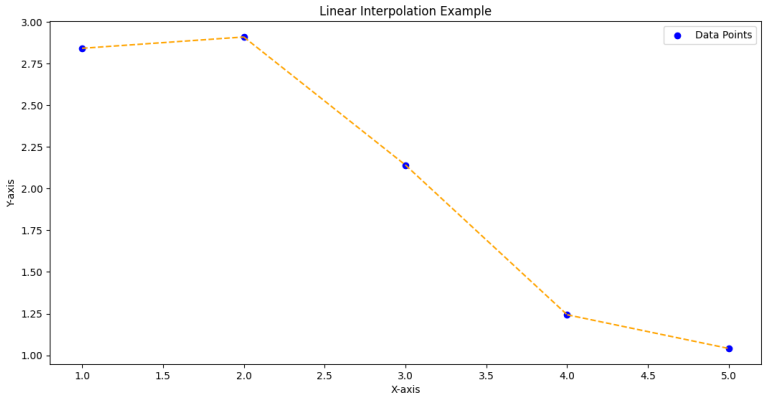


Figure A.1: Scatter plot showing data points (blue) and their linear interpolations (orange lines) for trend analysis.

Linear interpolation is used to fill in gaps between existing data points. This is done by connecting the original data points by linear lines. Figure A.1 shows an example of connecting the original blue data points with the orange linear lines. On these orange lines, data points can be added to fill in the gaps between the existing data points. To achieve this, we use the following formula:

$$\hat{y}(x) = y_i + \frac{(y_{i+1} - y_i)(x - x_i)}{(x_{i+1} - x_i)} \tag{A.1}$$

Here, i represents the index of the data points. We can vary i from 0 to n, where n is the total number of data points in the dataset. This formula helps us estimate the value y at a given position x by interpolating between the neighboring data points (x_i, y_i) and (x_{i+1}, y_{i+1}).[53]

A.2. R^2

To evaluate how well the predicted data fits the actual data, a R^2 calculation can be done. This done by calculating the ratio of the regression sum of squares to the total sum of squares:

$$R^2 = \frac{SSR}{SST} \tag{A.2}$$

SSR is the regression sum of squares and measures the variation of the predicted values $\hat{Y}$, about the mean of the observed Y's and $\hat{Y}$:

$$R^2 = 1 - \frac{SSE}{SST} \tag{A.3}$$

$SSR = SST - SSE$ where SST is the total sum of squares, which is the difference between Y and $\bar{Y}$. SST can be expressed as:

$$SST = \sum (Y_i - \bar{Y})^2 \tag{A.4}$$

$\bar{Y}$ is the mean of the predicted value is calculated as followed:

$$\bar{Y} = \frac{\sum Y_i}{N} \tag{A.5}$$

N is the number of predicted values and $\sum Y_i$ sum of the predicted values

. SSE is the error sum of squares and it's calculation is shown in equation A.6. It is the sum of the differences squared between the actual Y's and the predicted $\hat{Y}$.[54]

$$SSE = \sum (Y_i - \hat{Y}_i)^2 \tag{A.6}$$

The R^2 has the following properites:

- $0 \leq R^2 \leq 1$
- R^2 is closer to one, more similar the values of Y's and $\hat{Y}$
- when R^2 is zero or close to zero the Y's and $\hat{Y}$ are dissimilar.

A.3. Root Mean Square Error (RMSE)

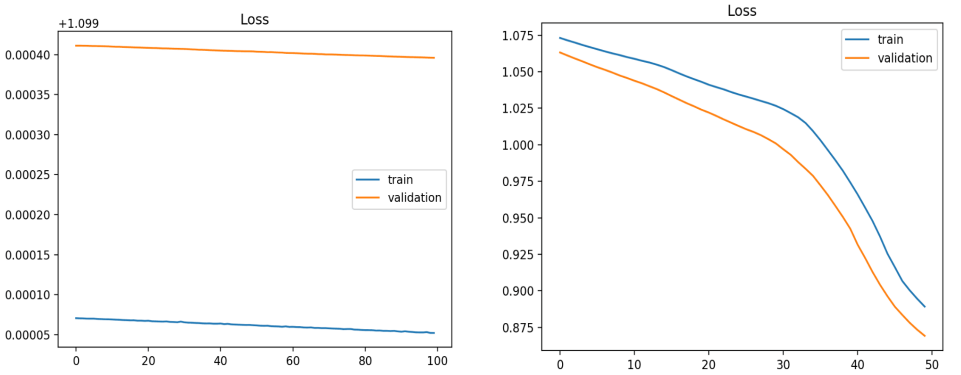
RMSE is the root of the mean square error between the predicted values ($Y_{p,i}$) and the actual values ($Y_{a,i}$). In the report the following equation is used to calculate the RMSE in degrees Celcius:[55]

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_{p,i} - Y_{a,i})^2} \tag{A.7}$$

A.4. Learning curve

A learning curve plots loss over epoch, this would indicate on the y-axis how the model is learning and on the x-axis the experience of the training. During the training of a machine learning model, the current state of the model at each step of the training algorithm can be evaluated. The training dataset gives an idea of how well the model is learning. There is also a hold-out validation dataset that is not part of the training. The evaluation dataset is used to evaluate how well the model is generalizing.

The learning curve consist therefore of two curves a training and a validation curve. The shape and dynamics of these curves can help evaluate the behavior of the machine learning model and can guide is to find the type of configuration for the machine learning model to improve the learning and/or performance. The learning curve can help observe three common dynamics: underfit, overfit, and good fit. These dynamics are explained using an example. The example assumes that we are looking at a minimizing metric, meaning the smaller the relative score on the y-axis the better the model is learning.



(a) Shows persistent high validation and training losses, indicating potential underfitting.

(b) Demonstrates insufficient training duration, leading to underfitting.

Figure A.2: Figures (a) and (b) illustrate examples of underfitting characteristics in learning curve.[56]

Underfitting indicates that the model cannot learn the training dataset. Underfitting can be identified by the training curve only. The training curve can show a flat line or noisy values of relatively high loss, this indicates that the model was not able to learn the training dataset at all. Figure A.2a gives an example of this behavior. Another way to identify overfitting is wel the training curve is decreasing and continues to decrease at the end of the plot. This behavior is shown in figure A.2b. This indicates that the model is still capable of further learning. In this case, the training process is halted prematurely and there is a possibility of further improvements in the learning of the training data.

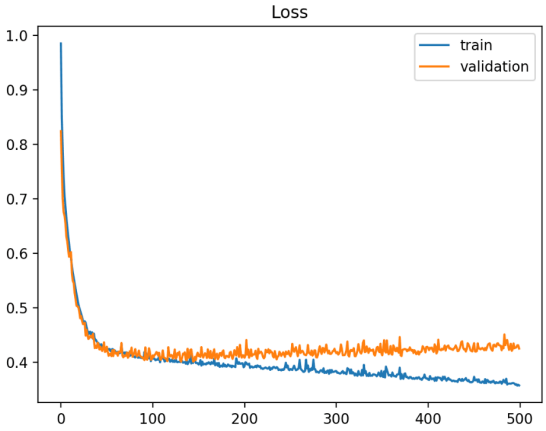


Figure A.3: Example of overfitting characteristics in learning curve, where the training curve keeps decreasing but the validation curve starts to rise, indicating poor generalization.[56]

Overfitting indicates that the model is learning to well, so including the noise or random fluctuations in the dataset. Overfitting causes that the more specialized the model becomes to the training data, the less well it can generalize the new data, where an increase in generalization error can occur. This increase in generalization error can be evaluated by the performance of the model on the validation curve. Overfitting occurs when the model has more capacity than required for the problem and too much flexibility. Overfitting can also occur when the model is trained for too long. When looking at the learning process, if the training curve keeps decreasing as experience grows, or if the validation curve decreases initially but then starts to rise again, it indicates that the model is overfitting. Figure A.3 shows an example of this phenomenon.

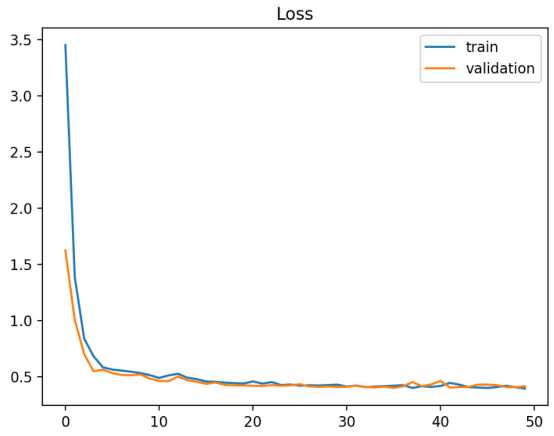


Figure A.4: Example of a learning curve when the model is fitting correctly. [56]

The goal of the learning curve is to find a model that gives a good fit and exists between overfit and underfit models. A good fit can be identified by training and a validation curve that decreases to a point of stability with a minimal gap between the two final loss values this can be seen in figure A.4. The loss of the model is in most cases lower on the training curve than the validation curve. Therefore, there is a small gap between the two cures, which is referred to as the ”generalization gap”. [56]

A.5. k-Fold Cross-validation

In k-fold cross-validation, the data set is divided into k ”folds”. K is usually between 5 and 10. In most cases the dataset is first shuffled and then divided into k folds. In the first iteration of model training, the first fold indicates the test data set and the remaining folds indicate a training dataset. Figure A.5 gives an example of this division of the data set into different folds and which fold is used as test data per iteration. For every k iteration, a metric score is calculated on the test data set for the evaluation of the model.

K-fold cross-validation can assist in evaluating whether the model is biased, depending on the samples chosen as test datasets. Additionally, the model does not have access to the test dataset during training. K-fold cross-validation can be used to create a more generalized model that can handle the entire dataset.[57]

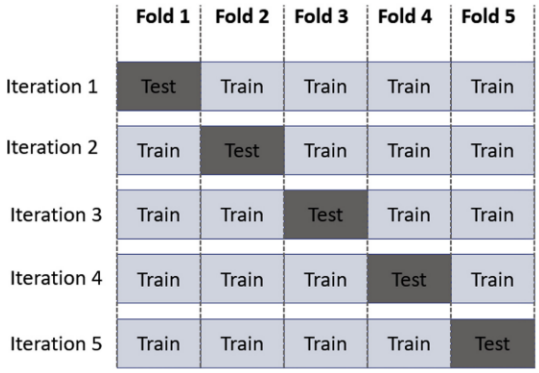


Figure A.5: Visualisation showing how the dataset is divided into separate folds for k-fold cross-validation.[57]

A.6. Recurrent Neural Network (RNN)

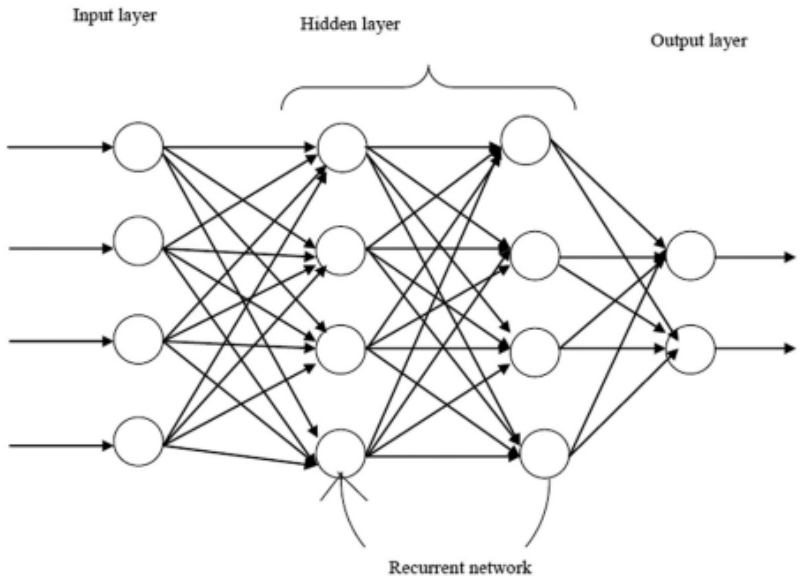


Figure A.6: Schematic drawing of a recurrent neural network.[58]

Recurrent neural network (RNN) is a type of neural networks that can be employed for sequence analysis. In a RNN, information travels from layer to layer with existing loops in the network so that each state is influenced by its previous states. Because of, the addition of loops RNNs has an ability that remember past computations and show dynamic time-based patterns. RNNs have hidden layers with an additional self-looped connection, this gives the model the ability to use information of the hidden layer used at one moment of time as input for the next. The structure of a RNN is shown in Figure A.6.[59]

$$o = f(h_t, \theta) \tag{A.8}$$

$$h_t = g(h_{t-1}, x_t, \theta) \tag{A.9}$$

In the equation o represents the value given to a node, θ encapsulates the weights and bias for the network, and h_t is the state of the hidden layer at time t. Figure A.7 show the values of the equation representations in the recurrent neural network. Where the x_t are the input nodes of the total RNN architecture shown in Figure A.6.[58]

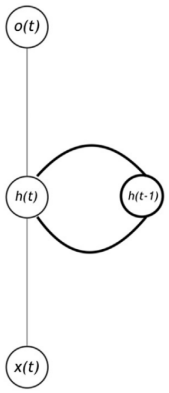


Figure A.7: Schematic drawing of recurrent neural network.[58]

B

Appendix

B.1. Singular Value Decomposition of CM and WL models

In the report the results of the Singular Value Decomposition (SVD) of the HM model is showed and discussed. In this section, the results of the SVD of the CM model represented in Figure B.1 and the SVD of the WL model in Figure B.2 where the relative error as function of the number of $n\_components$ is plotted. Interestingly, for the CM and the WL model we observe a similar behavior as the model discussed in the section 4.1, with the lowest relative error at 31 $n\_components$. Beyond this point, the relative error starts to increase, Therefore, choosing $n\_components = 31$ is the most optimal amount of $n\_components$ due to its accuracy and the low dimensionality.

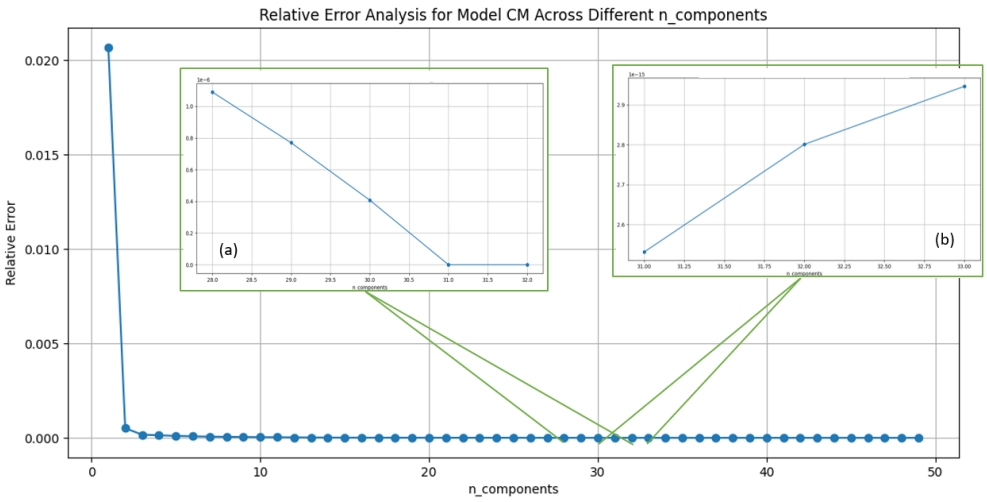


Figure B.1: Relative error of U_m ranging from 1 to 50. Subfigures (a) and (b) are zoomed views showing the relative error for U_m 28 to 32 and 31 to 33, respectively for the CM model.

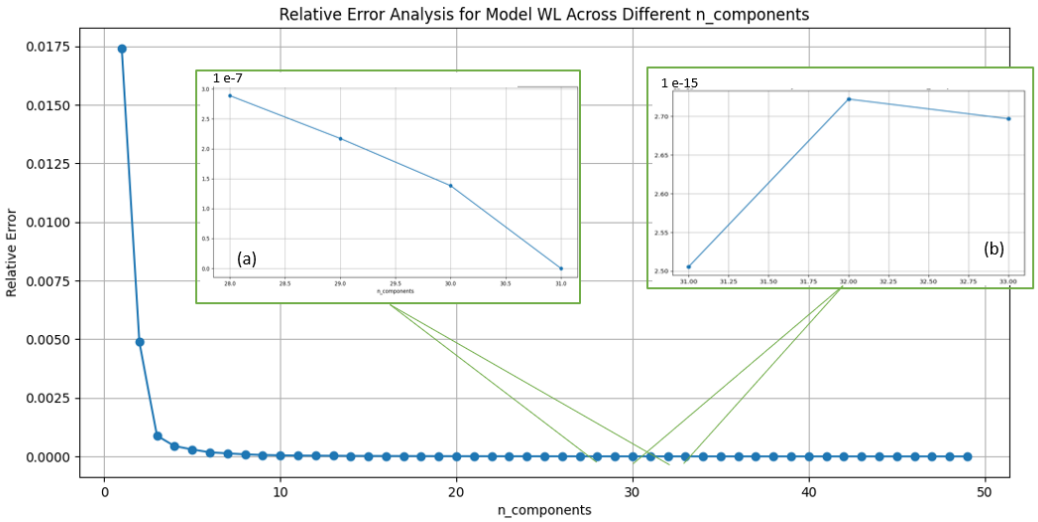
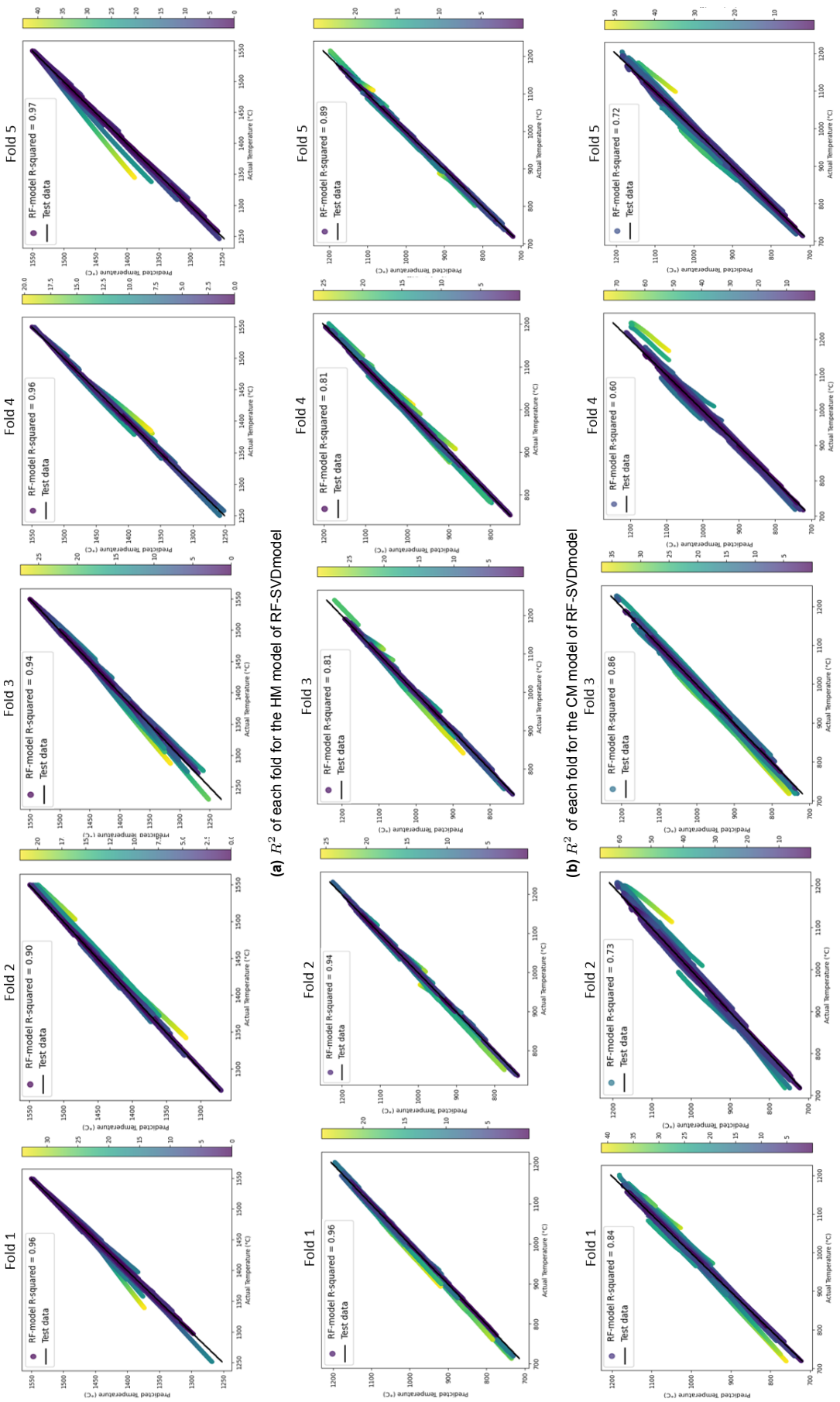


Figure B.2: Relative error of U_m ranging from 1 to 50. Subfigures (a) and (b) are zoomed views showing the relative error for U_m 28 to 32 and 31 to 33, respectively for the WL model.

B.2. K-fold Cross Validation of CM HM and WL

The results of the k-fold Cross Validation results are discussed in section 5.1.1. Figure B.3a, Figure B.3b and Figure B.3c represents the visualizations of the results of k-fold cross-validation, where for each fold, the test data and predicted values by the RF-SVDmodel are plotted against each other. The black line in the represents the test data, while the scattered points on the color bar illustrate the predicted values. The color bar serves as an indicator for the difference between the original test data and the predicted values in degrees Celsius.



(a) R^2 of each fold for the HM model of RF-SVDmodel

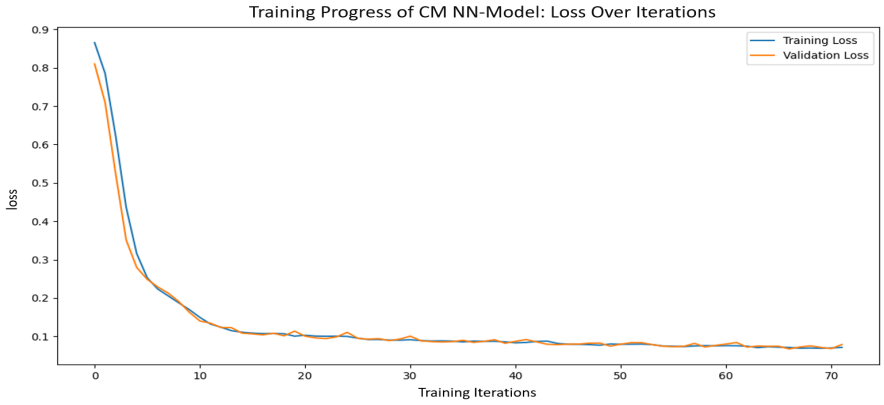
(b) R^2 of each fold for the CM model of RF-SVDmodel

(c) R^2 of each fold for the WL model of RF-SVDmodel

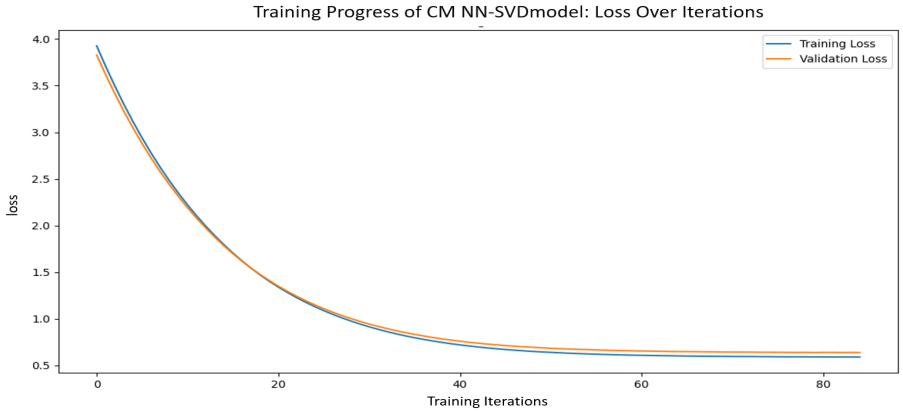
Figure B.3: The k-fold cross validation for the CM,WL and HM model.

B.3. Learning Curves of the WL and CM models for NN-SVDmodel and NN-model

The results of the learning curves of the HM model is discussed and explained in section 5.1.2. Figure B.4 and B.5 show the learning curves of the NN-SVDmodel and the NN-model for WL and CM, showing the progression of training and validation loss over epochs. In CM model for NN-SVDmodel and NN-model we see that the validation and training curve overlap. For the WL model we see this overlap in the NN-model but in the NN-SVDmodel we see a small ”generalization” gap between the two curves. The overlap in validation and the training loss curve indicates a good learning behavior of these models during training, and suggesting an effective learning without significant under- or overfitting. For the WL model of the NN-SVDmodel there is a small gap but still under the 5%, which according to the model performance criteria still indicates that this model is generalizing well. However, due to a small generalization gap there could be a small overfitting be present in this model.

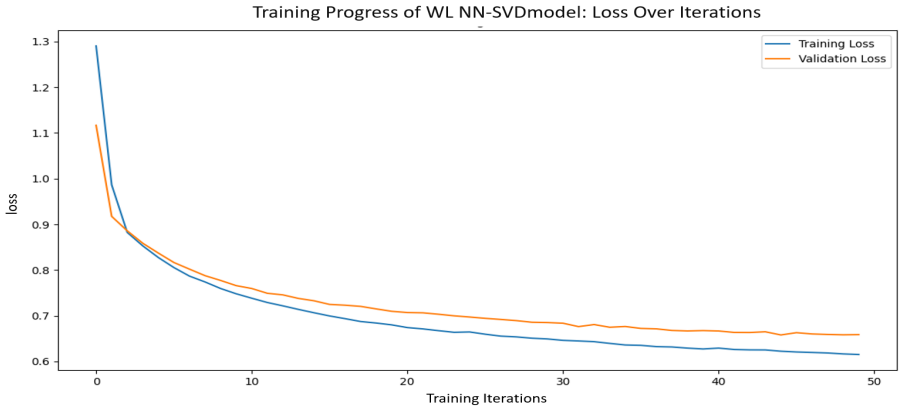


(a) Learning curve of NN-model

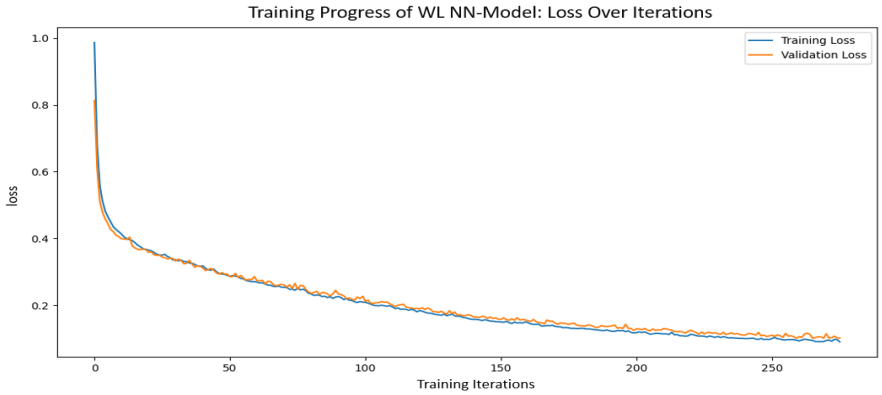


(b) Learning curve of NN-SVDmodel

Figure B.4: Learning curves of the CM model for both the NN-SVD model and the NN model



(a) Learning curve of NN-SVDmodel

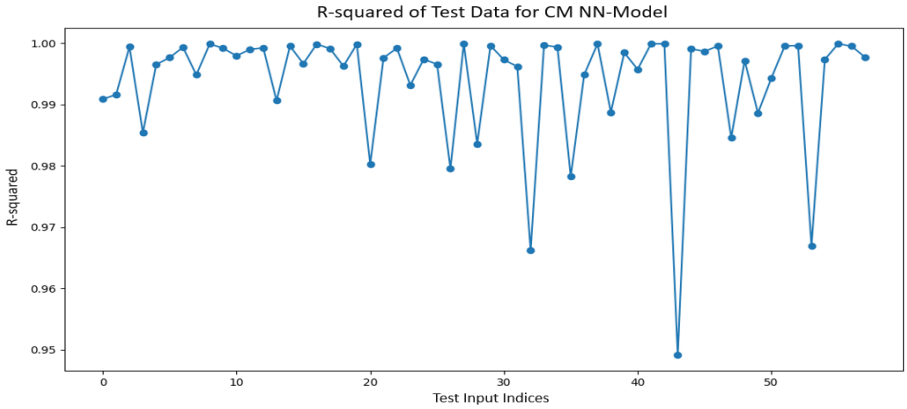


(b) Learning curve of NN-model

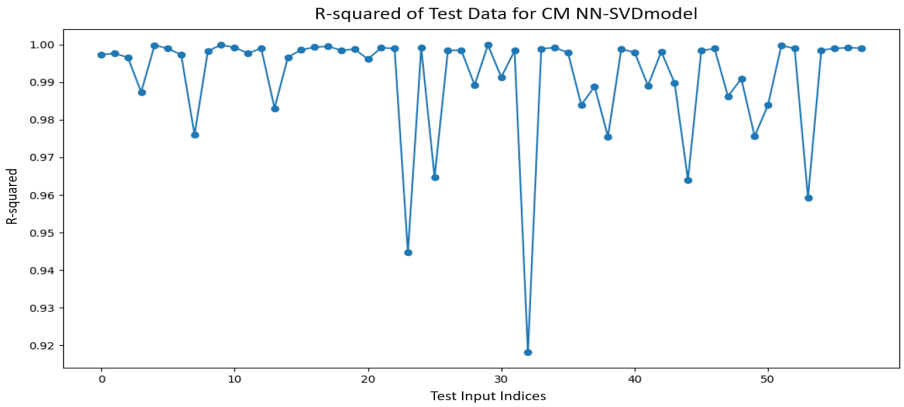
Figure B.5: Learning curves of the WL model for both the NN-SVD model and the NN model

B.4. Test Evaluation of the WL and CM models for NN-SVDmodel and NN-model

The results of the evaluation on the test data of the HM model is discussed and explained in section 5.2.In Figure B.6 and B.5, illustrates the R^2 of the test data for the NN-model and NN-SVDmodels for CM and WL models. For both both models of the CM model, a R^2 value ranging between 0.92 and 0.99 is achieved. For the WL, a R^2 value ranging between 0.86 and 0.99 is achieved. This indicates that both models is accurately capturing the underlying relationships within the data.

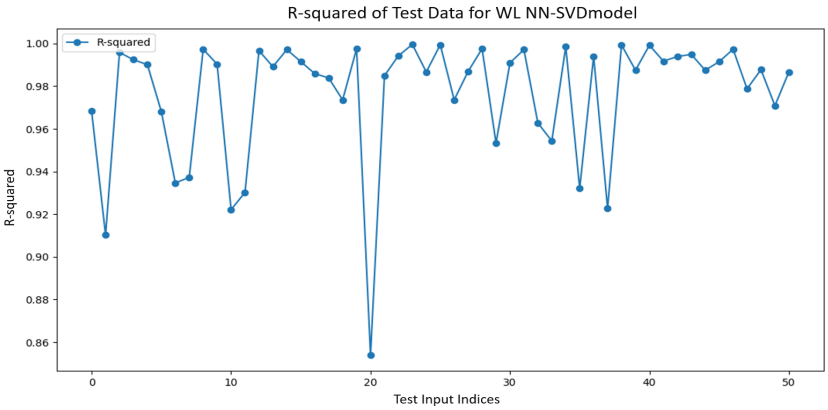


(a) R^2 values of the test data for NN-model

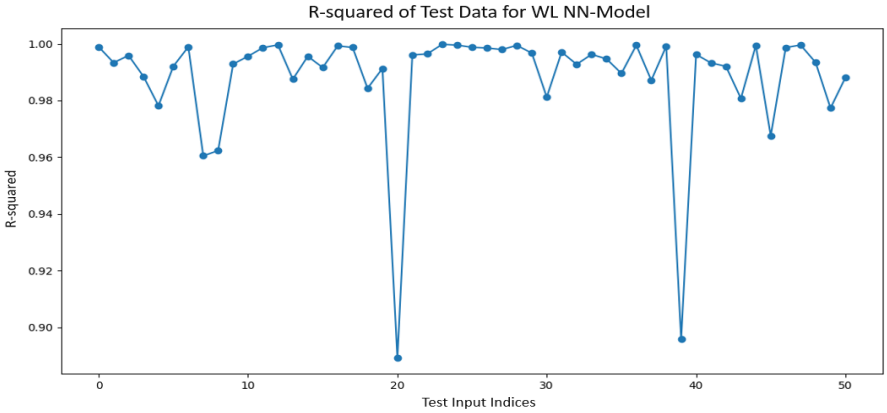


(b) R^2 values of the test data for NN-SVDmodel

Figure B.6: The R^2 values for the test data of the CM model for both the NN-SVD model and the NN model.



(a) R^2 values of the test data for NN-model

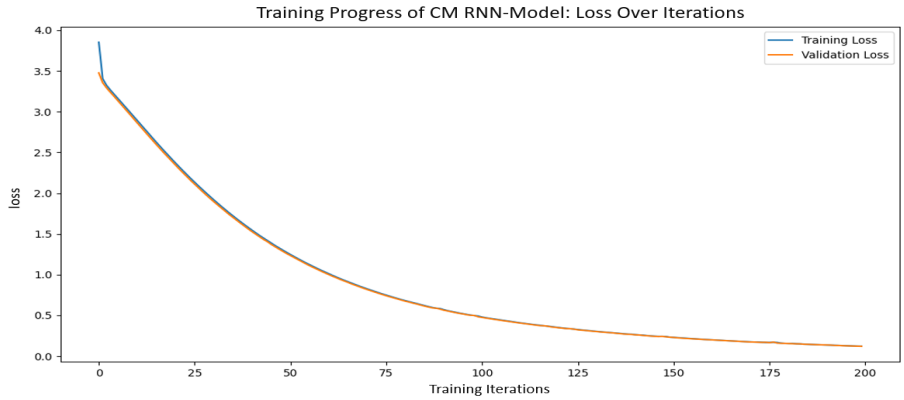


(b) R^2 values of the test data for NN-SVDmodel

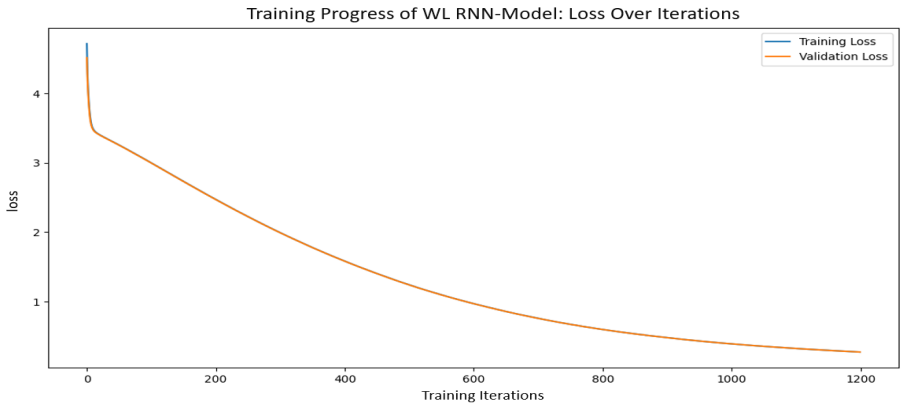
Figure B.7: The R^2 values for the test data of the WL model for both the NN-SVD model and the NN model.

B.5. Learning Curves of the WL and CM models for RNN-model

The results of the learning curves of the HM model is discussed and explained in section 5.1.2 for the RNN-model. Figure B.8a and B.8b show the learning curves of the RNN-model for WL and CM, showing the progression of training and validation loss over epochs. In CM and WL models we see that the validation and training curve overlap. The overlap in validation and the training loss curve indicates a good learning behavior of these models during training, and suggesting an effective learning without significant under- or overfitting.



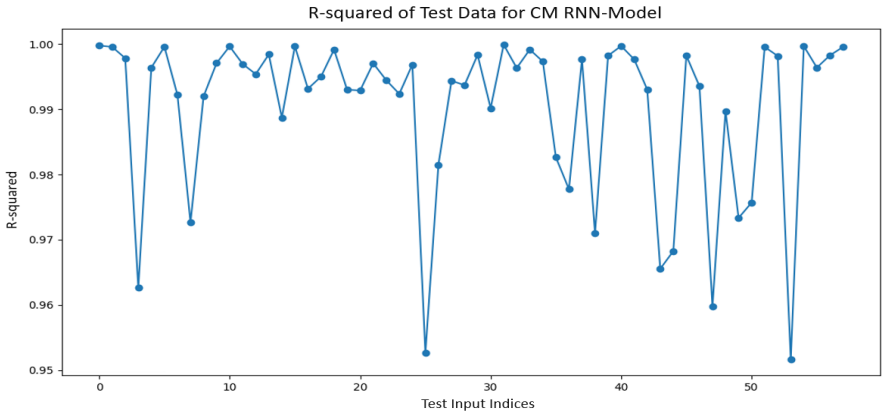
(a) Learning curve of CM model



(b) Learning curve of WL model

Figure B.8: Learning curves for the RNN model's CM and WL models.

The results of the evaluation on the test data of the HM model is discussed and explained in section 5.2 for the RNN-model.In Figure B.9a and B.9b, illustrates the R^2 of the test data for the RNN-model for CM and WL models. For the CM model, a R^2 value ranging between 0.92 and 0.99 is achieved. For the WL, a R^2 value ranging between 0.88 and 0.99 is achieved. This indicates that both models is accurately capturing the underlying relationships within the data.



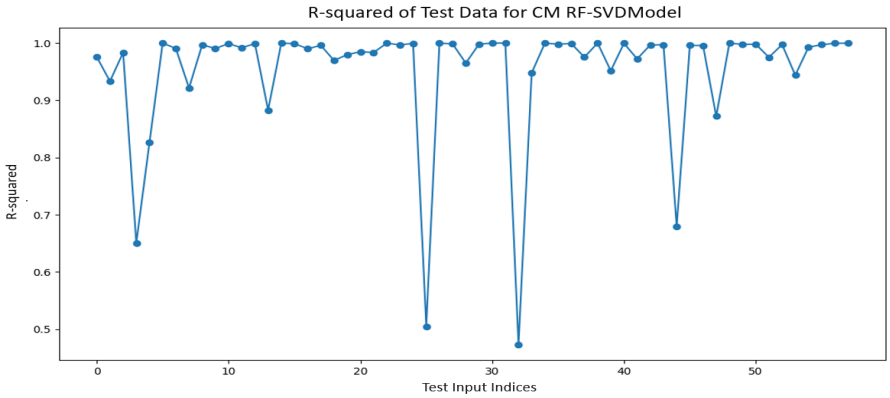
(a) R^2 values of the test data for CM model

(b) R^2 values of the test data for WL model

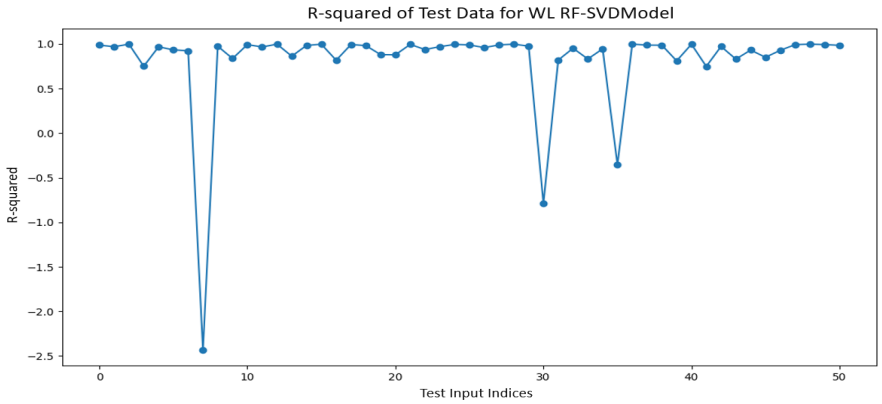
Figure B.9: The R^2 values for the test data of the RNN model's CM and WL models.

B.6. Test Evaluation of the WL and CM models for RF-SVDmodel

The results of the evaluation on the test data of the HM model is discussed and explained in section 5.2 for the RF-model. In Figure B.10a and B.10b, illustrates the R^2 of the test data for the RNN-model for CM and WL models. For the CM model, a R^2 value ranging between 0.5 and 0.99 is achieved. For the WL, a R^2 value ranging between -2.5 and 0.99 is achieved. This range is not meeting the criteria that R^2 should be close to 1 for all test datasets, suggesting that there are instances where the model struggles to accurately capture the underlying relationships within the data.



(a) R^2 values of the test data for CM model



(b) R^2 values of the test data for WL model

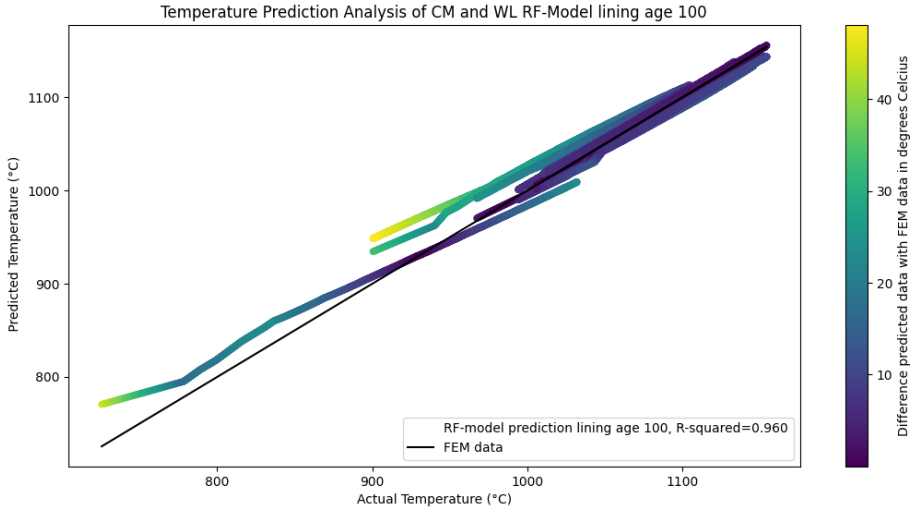
Figure B.10: The R^2 values for the test data of the RF-SVDmodel's CM and WL models.

C

Appendix

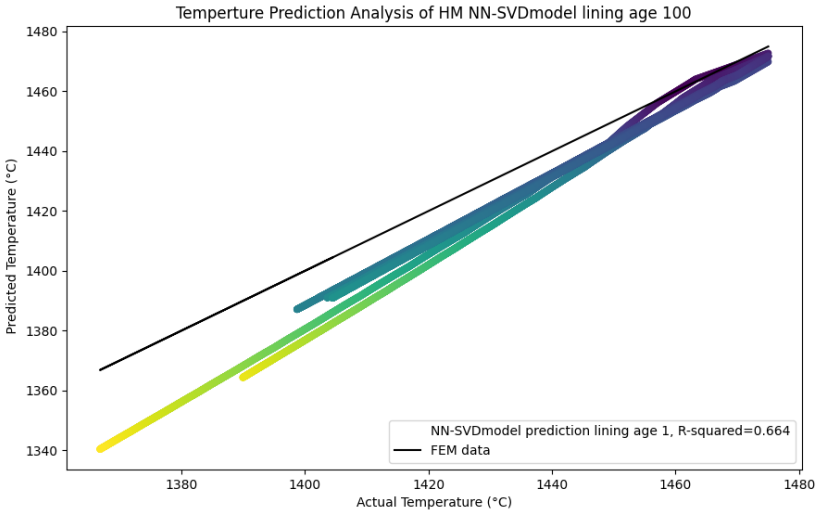
All the models were evaluated on three different validation datasets. For the RF-model, the NN-model, the NN-SVDmodel and the RNN-model the lining age 1 validation dataset results are shown in section 5.1. The validation datasets for the RF-model, representing lining ages 100 and 400, are shown in section C.1, while those for the NN-SVDmodel are presented in section C.2. Additionally, the results for the NN-SVD model are presented in section C.3 and for the RNN-model are presented in section C.4.

C.1. RF-Model



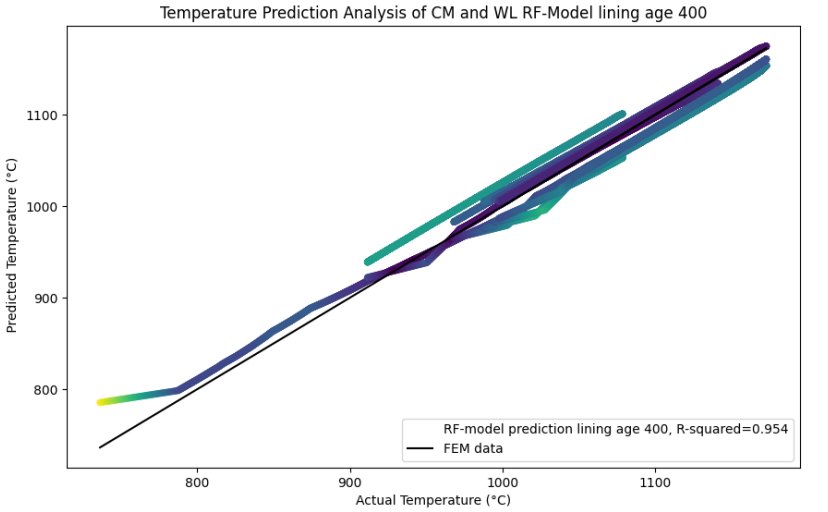
#	Δ ($C\circ$)
1	-45.030
2	12.877
3	-48.074
4	-8.446
5	-24.353
6	7.095
7	-6.719
8	11.834
9	0.560
10	6.649
11	-8.011
12	9.048
13	-8.332
14	10.709
15	-6.108
16	10.595
17	-6.278

Figure C.1: The results of the validation dataset of lining age 100 for WL and CM model of RF-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in full cycle prediction.



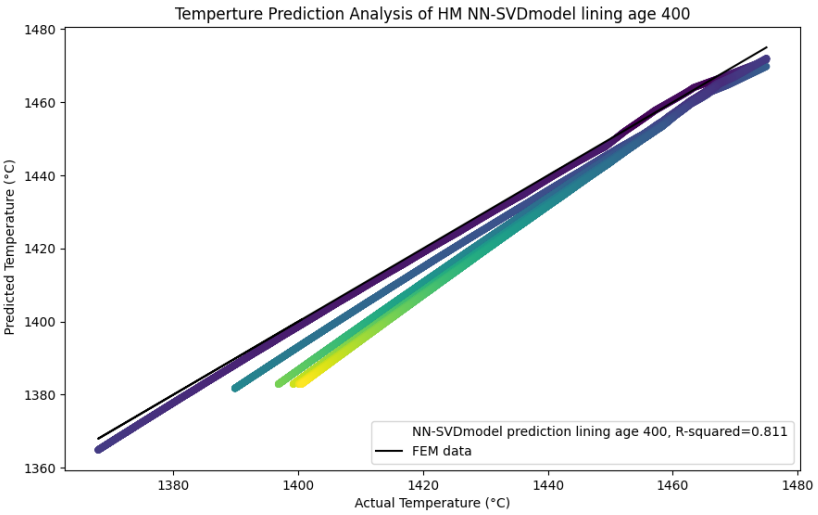
#	$\Delta\ (C\circ)$
1	178.765
2	169.961
3	146.978
4	147.449
5	148.787
6	148.350
7	148.952
8	149.762

Figure C.2: The results of the validation dataset of lining age 100 for HM model of RF-model. The table shows the temperature differences for the last temperature value of each prediction of the HM model, which are the initial refractory temperatures used in full cycle prediction



#	$\Delta\ (C\circ)$
1	-49.062
2	25.471
3	-27.382
4	6.778
5	-14.618
6	14.995
7	-16.747
8	20.763
9	-12.097
10	19.124
11	-8.188
12	12.522
13	-6.202
14	12.109
15	-5.548
16	11.253
17	-6.146

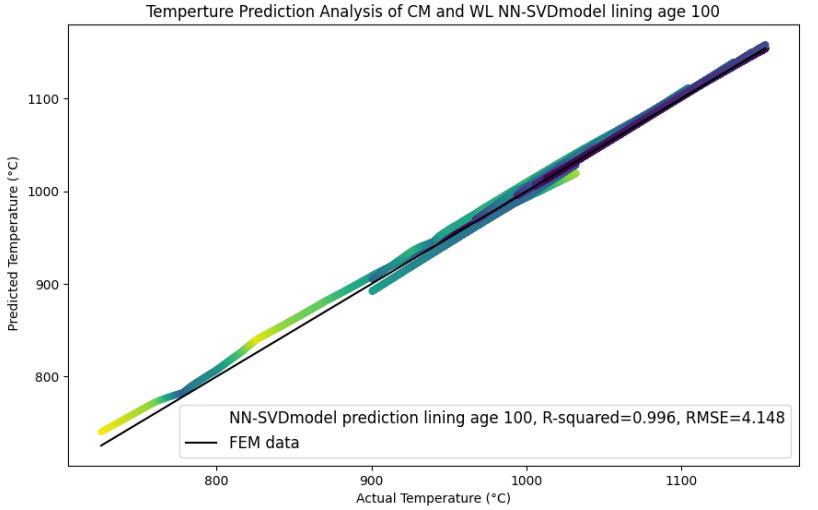
Figure C.3: The results of the validation dataset of lining age 400 for WL and CM model of RF-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models.



#	Δ ($C\circ$)
1	3.748
2	5.998
3	13.345
4	16.319
5	17.239
6	17.519
7	17.628
8	17.678

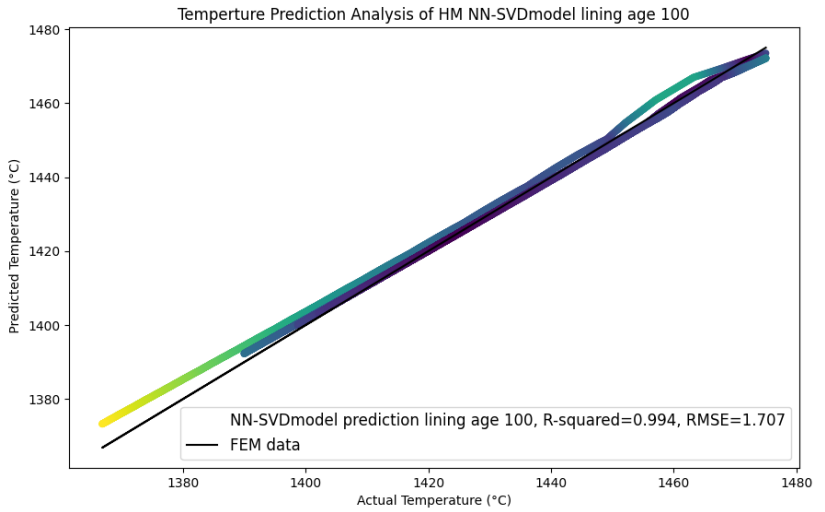
Figure C.4: The results of the validation dataset of lining age 400 for HM model of RF-model. The table shows the temperature differences for the last temperature value of each prediction of the HM model.

C.2. NN-SVDmodel



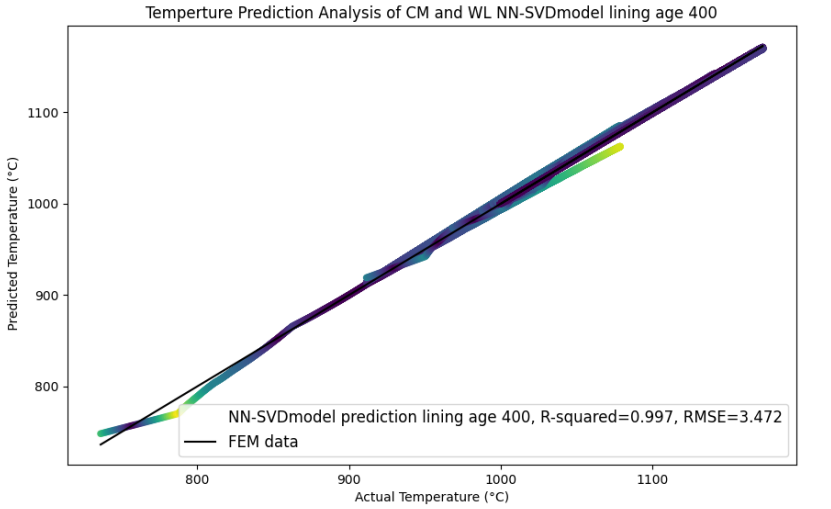
#	Δ ($C\circ$)
1	-14.774
2	2.611
3	8.247
4	-2.023
5	-1.934
6	-2.036
7	-1.233
8	-1.006
9	-0.384
10	-0.374
11	-0.400
12	-0.190
13	-0.007
14	0.147
15	-0.059
16	0.285
17	0.009

Figure C.5: The results of the validation dataset of lining age 100 for WL and CM model of NN-SVDmodel. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in full cycle prediction.



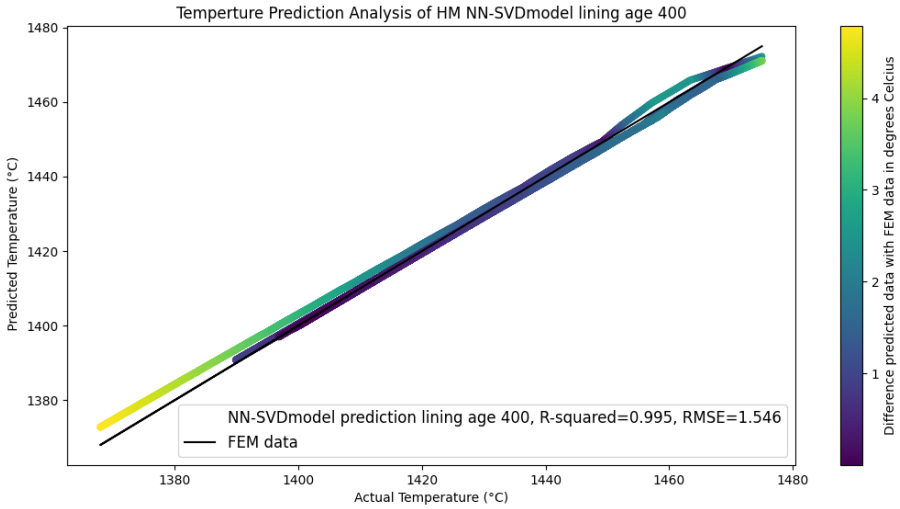
#	$\Delta\ (C\circ)$
1	-6.417
2	-2.317
3	-1.282
4	-0.991
5	-0.903
6	-0.946
7	-0.882
8	-0.889

Figure C.6: The results of the validation dataset of lining age 100 for HM model of NN-SVDmodel. The table shows the temperature differences for the last temperature value of each prediction of the HM model.



#	$\Delta\ (C\circ)$
1	-11.926
2	15.999
3	-2.325
4	0.612
5	-3.546
6	1.511
7	-3.220
8	2.534
9	-2.379
10	3.079
11	-2.962
12	1.666
13	-0.523
14	1.480
15	-0.242
16	1.542
17	-0.411

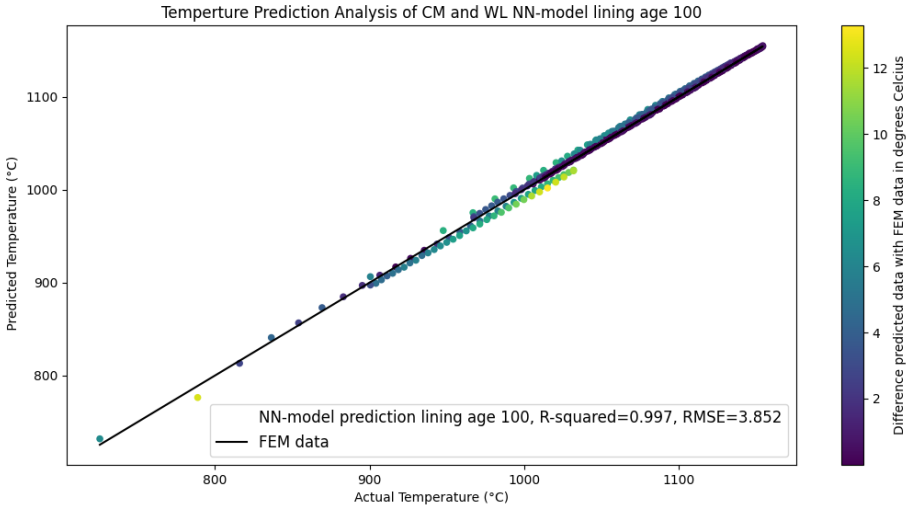
Figure C.7: The results of the validation dataset of lining age 400 for WL and CM model of NN-SVDmodel. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in full cycle prediction.



#	$\Delta\ (C\circ)$
1	-4.124
2	-3.066
3	-0.776
4	0.047
5	0.170
6	0.158
7	0.038
8	0.011

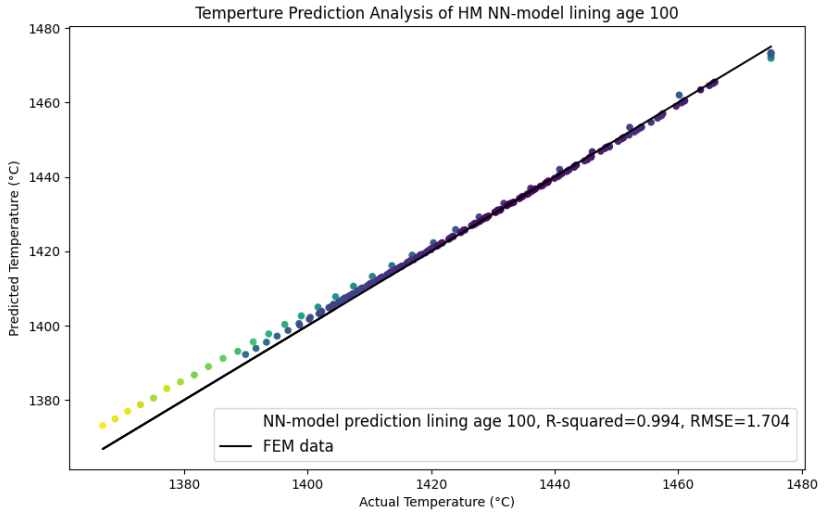
Figure C.8: The results of the validation dataset of lining age 400 for HM model of NN-SVDmodel. The table shows the temperature differences for the last temperature value of each prediction of the HM model.

C.3. NN-model



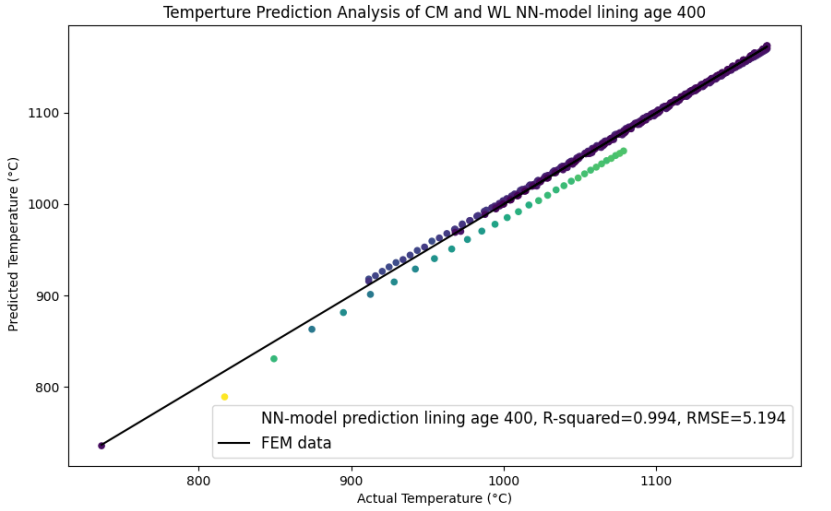
#	$\Delta\ (C\circ)$
1	-6.377
2	0.828
3	3.108
4	-2.337
5	-3.620
6	-2.935
7	-1.624
8	-1.419
9	-0.353
10	-0.663
11	-0.217
12	-0.195
13	0.232
14	0.248
15	0.207
16	0.436
17	0.286

Figure C.9: The results of the validation dataset of lining age 100 for WL and CM model of NN-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in full cycle prediction.



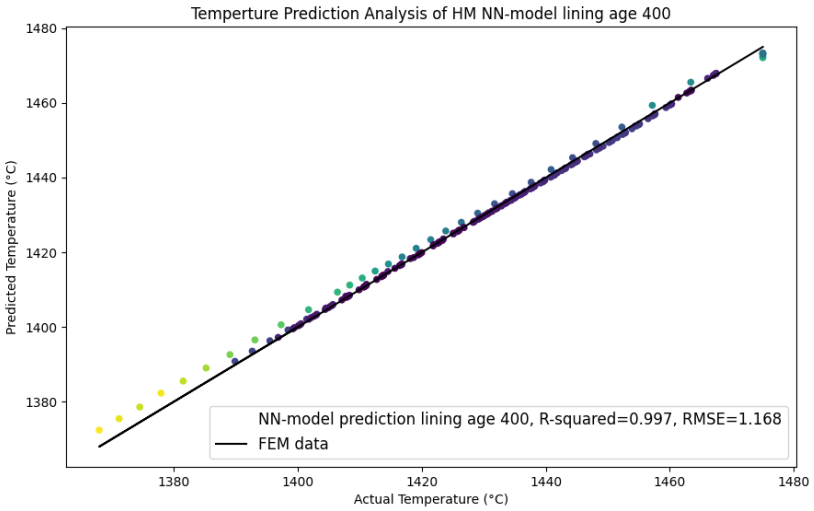
#	$\Delta\ (C\circ)$
1	-6.321
2	-2.289
3	-1.437
4	-1.250
5	-1.213
6	-1.280
7	-1.225
8	-1.236

Figure C.10: The results of the validation dataset of lining age 100 for HM model of NN-model. The table shows the temperature differences for the last temperature value of each prediction of the HM model.



#	$\Delta\ (C\circ)$
1	1.114
2	20.274
3	-6.479
4	-0.877
5	-4.461
6	0.946
7	-3.362
8	1.625
9	-2.334
10	2.360
11	-2.837
12	1.027
13	-0.409
14	0.899
15	-0.118
16	0.988
17	-0.281

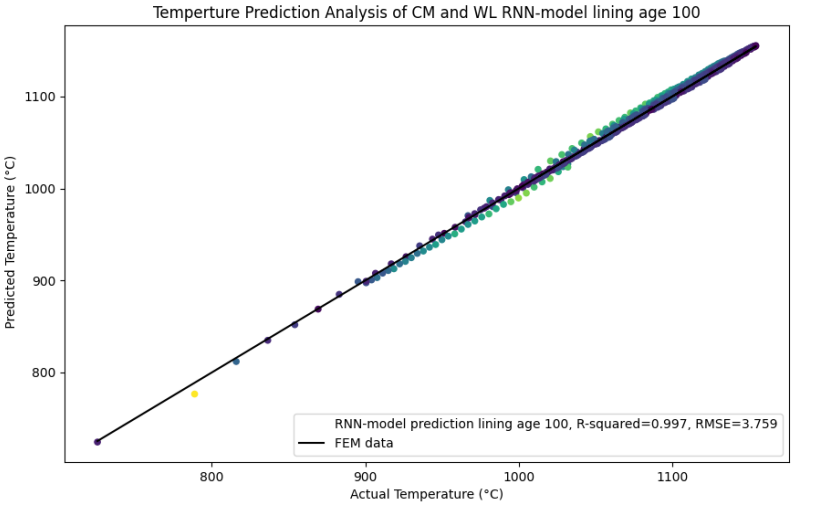
Figure C.11: The results of the validation dataset of lining age 400 for WL and CM model of NN-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in full cycle prediction.



#	Δ ($C\circ$)
1	-3.786
2	-3.104
3	-1.027
4	-0.296
5	-0.210
6	-0.236
7	-0.365
8	-0.397

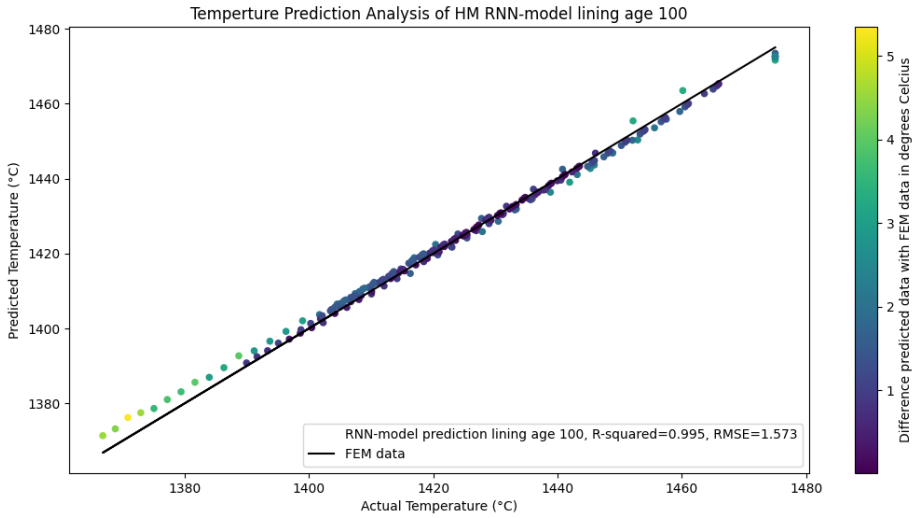
Figure C.12: The results of the validation dataset of lining age 400 for HM model of NN-model. The table shows the temperature differences for the last temperature value of each prediction of the HM model.

C.4. RNN-model



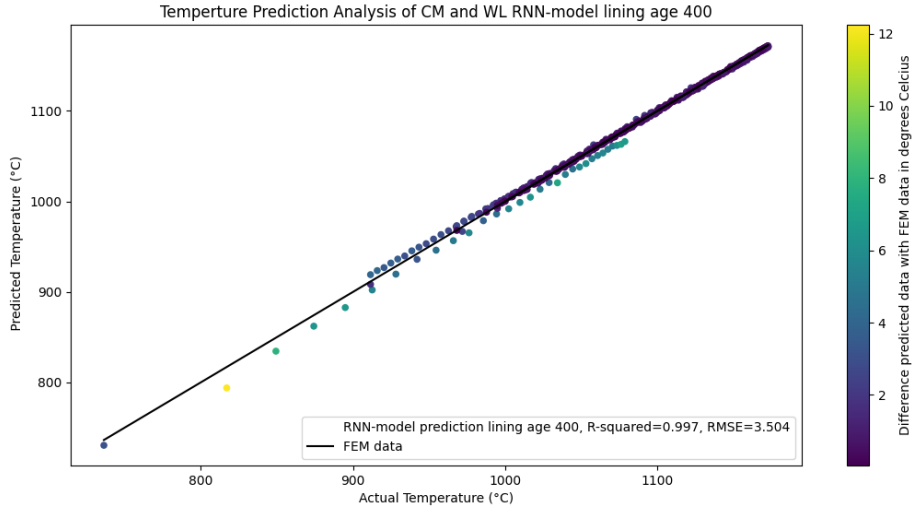
#	Δ ($C\circ$)
1	1.115
2	-4.709
3	3.015
4	-5.881
5	-1.677
6	-4.773
7	0.225
8	-2.807
9	1.406
10	-1.738
11	1.502
12	-1.359
13	1.935
14	-0.949
15	1.902
16	-0.774
17	1.979

Figure C.13: The results of the validation dataset of lining age 100 for WL and CM model of RNN-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in full cycle prediction.



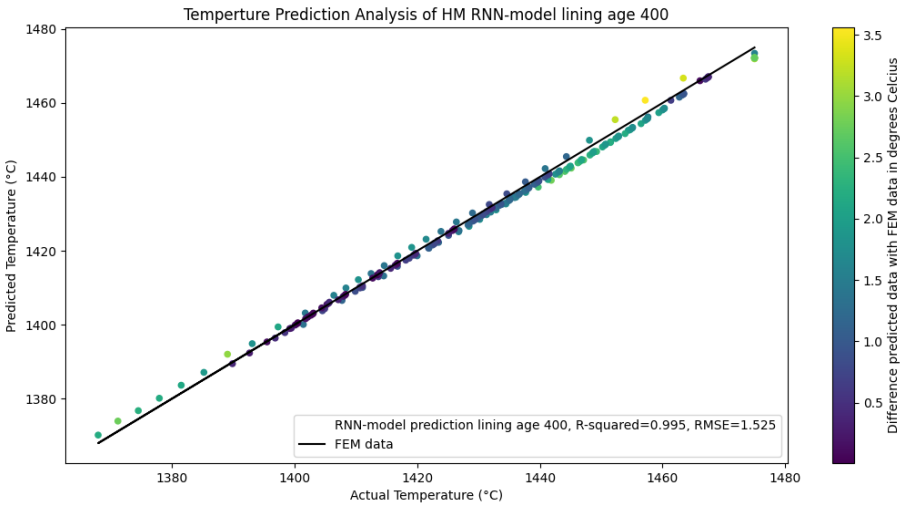
#	$\Delta\ (C\circ)$
1	-4.550
2	-0.834
3	-0.954
4	-1.223
5	-1.369
6	-1.512
7	-1.485
8	-1.510

Figure C.14: The results of the validation dataset of lining age 100 for HM model of RNN-model. The table shows the temperature differences for the last temperature value of each prediction of the HM model



#	$\Delta\ (C\circ)$
1	5.867
2	12.482
3	-7.559
4	0.150
5	-5.061
6	0.649
7	-3.720
8	1.659
9	-2.573
10	2.207
11	-3.005
12	0.796
13	-0.588
14	0.611
15	-0.289
16	0.674
17	-0.445

Figure C.15: The results of the validation dataset of lining age 400 for WL and CM model of RNN-model. The table shows the temperature differences for the last temperature value of each prediction of the WL and CM models, which are the initial refractory temperatures used in full cycle prediction.



#	Δ ($C\circ$)
1	-1.535
2	-1.746
3	-0.174
4	0.247
5	0.214
6	0.144
7	-0.019
8	-0.062

Figure C.16: The results of the validation dataset of lining age 400 for HM model of RNN-model. The table shows the temperature differences for the last temperature value of each prediction of the HM model

D

Appendix

This appendix provides information on the data visualization of the Plotly dashboard developed for analyzing Finite Element Model data used in creating the digital twin. The dashboard consists of a series of tabs. The first three tabs provide an overview of various output datasets and their corresponding input values for the CM, WL, and HM models. Furthermore, two tabs provide information on the input distribution of the CM, WL, and HM models. The last tab offers insights into temperature changes over time for each dataset and the influence of certain input values on this change.

D.1. Tabs 1-3: Output Plots with Input Values

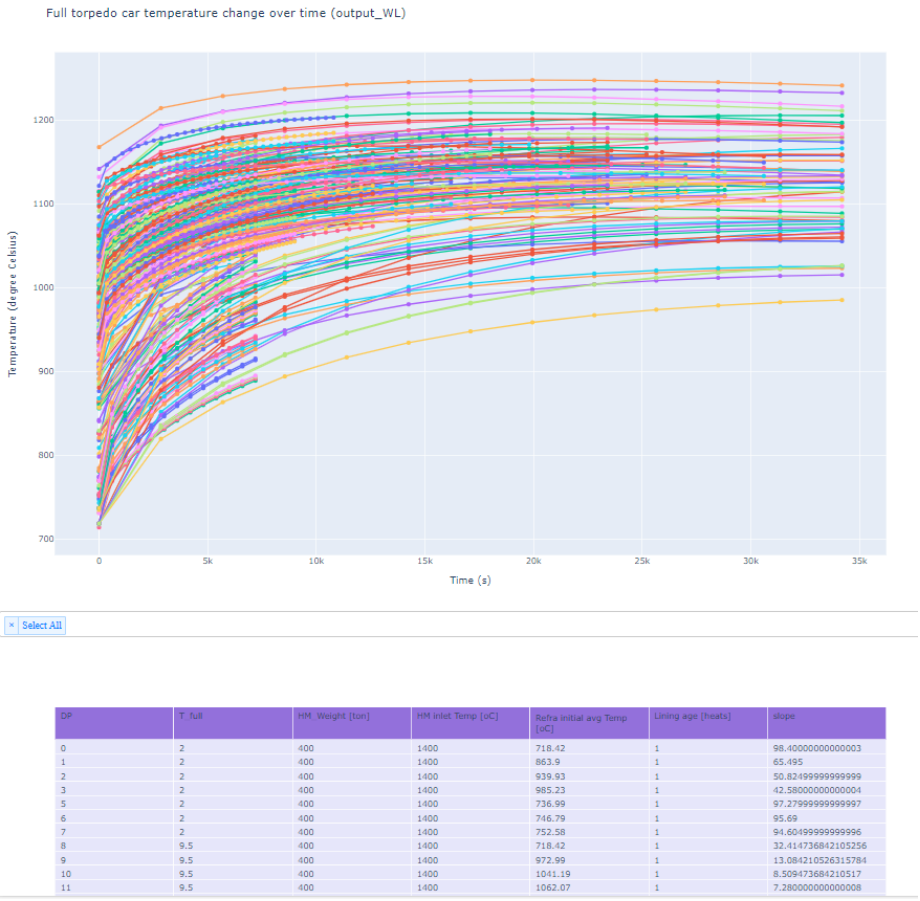


Figure D.1: Image of the tab of the WL model created in the Plotly dashboard.

Tab 1 consist of the WL model datasets. featuring a total of 252 datasets temperature-time distributions ranging from 13 to 31 values.Each dataset is associated with five corresponding input values. This tabs displays all the temperature- time distributions, with the table below providing the input values corresponding to each distribution. In Figure D.1 illustrates this layout. When you move your mouse over the various lines in the graph, the DP number along with the y and x values at that point will appear. The DP number can be found in the table below helping to match each curve with its corresponding input value. In Figure D.1 there is a ”select all” button, where you can choose which datasets you want to have displayed in the graph. The Table below is automatically adjusted and gives you the input values corresponding to the curves shown in the graph. Tabs 2 and 3 follow the same structure as Tab 1 but displaying the HM and CM models respectively.

D.2. Tabs 4 and 5: Input Distribution

Tab 4 provides the input distribution for the CM model, while Tabs 4 and 5 offer the input distributions for the HM and WL models, respectively. These are already explained and provided in Section 3.3.

D.3. Tab 6: Temperature Change Analysis

Tab 6 provides insights into the temperature change over time in the torpedo ladle car. For each dataset, the maximum and minimum temperature values are subtracted and divided by the time. In Figure D.2, these values are plotted against the initial temperature of the refractory for the CM, WL, and HM models. Each plot allows users to choose input values and observe how these values are distributed around the plot for certain input values. This gives us inside in how the rate of the temperature of the refractory varies under different conditions.

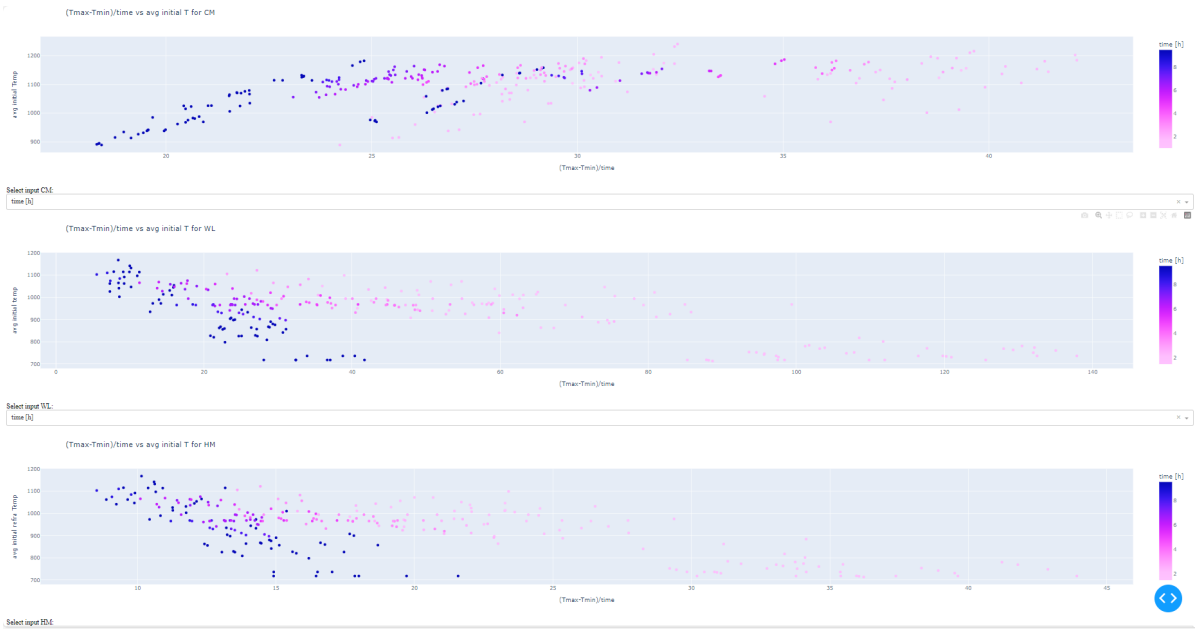


Figure D.2: Graphs of the WL, CM and HM model of the initial refractory temperature over the temperature change divided by the time per dataset.