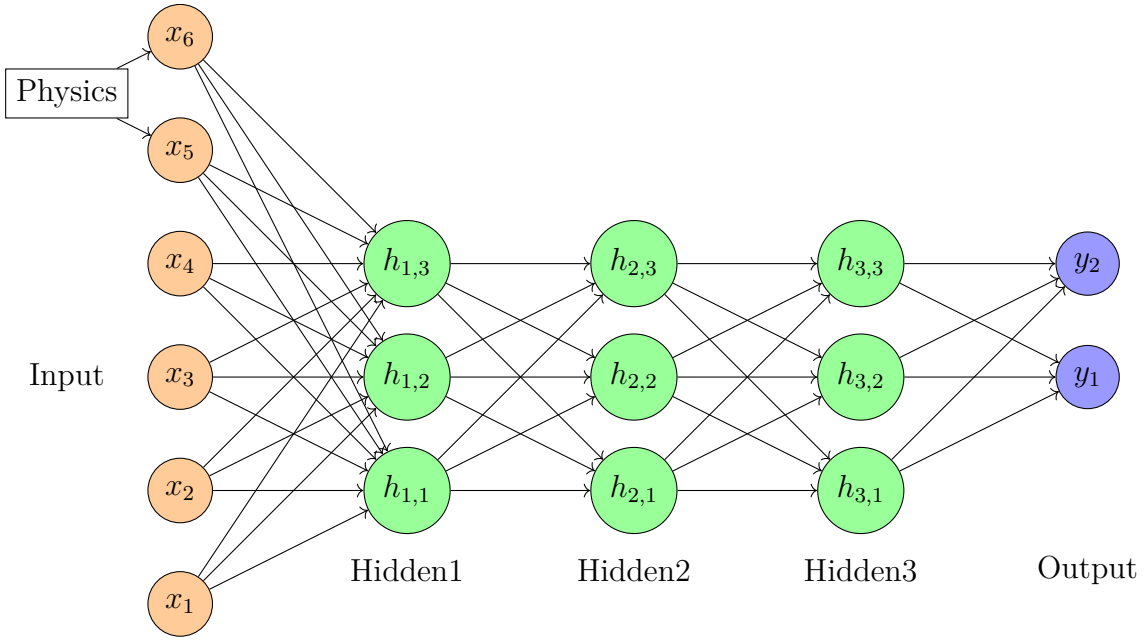
CHALMERS
UNIVERSITY OF TECHNOLOGY



Physics Informed Neural Network for thermal modeling of an Electric Motor

Master's thesis in Sustainable electric power engineering and electromobility

Karl Svantesson & Jesper Stensson

DEPARTMENT OF ELECTRICAL ENGINEERGING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2023

www.chalmers.se

Master's thesis 2023

Physics Informed Neural Network for thermal modeling of an Electric Motor

Karl Svantesson & Jesper Stensson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Electric Power Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

Physics Informed Neural Network for thermal modeling of an Electric Motor
Karl Svantesson Jesper Stensson

Supervisors: Erik Ströby, Jiasheng Wang, Volvo Group Technology
Examiner: Torbjörn Thiringer, Department of Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Division of Electric Power Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2023

Abstract

Artificial intelligence and machine learning are becoming increasingly significant, and the need to investigate the potential for different areas arises. This project investigated the potential of utilizing data-driven techniques for the thermal model of the motor drive system at Volvo GTT. The aim was to incorporate the already-known physics of the system into the data-driven models through different constraints to achieve higher performance. The physics-informed neural network was built using the PyTorch framework, and the project tested multiple types of networks and hyperparameters. Another model was created using the Nerve framework developed by Volvo. The Nerve model only took one week to develop, which is significantly shorter than the four months it took to develop the PINN model. The Nerve model underwent training on a large amount of data, but its ability to accurately predict the output of the thermal model was inconsistent.

It was shown that the self-developed data-driven gated recurrent unit model can model the system effectively. Further, the data-driven model with physical constraints performed better than the models without incorporating previously known physics. The best performance acquired in this investigation showed an 80 % reduction in MAE loss for power loss estimations and a 53 % reduction in winding temperature estimations when incorporating physics compared to a purely data-driven model. Based on the results, it is clear that PINNs have great potential for applications where there is a shortage of available data. In such cases, traditional data-driven models may not be able to accurately capture the dynamics of the thermal model as effectively as the PINNs.

Keywords: PINN, Nerve, MAE, data-driven modeling, GRU, Artificial intelligence, Machine Learning

Acknowledgements

Firstly we would like to thank Volvo Group Technology and the Simulation team for allowing us to write our Master's thesis with them. Special thanks to Erik Ströby, Jiasheng Wang Björm Kleman, Henrik Berggren, and Noel Hall that has given feedback throughout the entire project. Secondly, we would like to direct thanks to Parthasarathy Dhasarathy for the help with setting up the Nerve framework. Rahul Nath for providing help with the physical representation of the thermal model. At last, we would like to thank our examiner Torbjörn Thiringer for the feedback and guidance.

Jesper Stensson & Karl Svantesson
Gothenburg, June, 2023

List of Acronyms

AI	Artificial Intelligence
BMS	Battery Management System
ESS	Energy Storage System
FFNN	Feed-Forward Neural Network
GRU	Gated Recurrent Unit
GSP	Global Simulation Platform
LSTM	Long-Short Term Memory
MAE	Mean Absolute error
MDS	Motor Drive System
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
PINN	Physics Informed Neural Network
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROM	Reduced Order Modeling

Nomenclature

$P_{WindingCoolant}$ Power from heat transfer from stator windings to the coolant liquid
$T_{CoolantOut}$ Temperature of the coolant liquid out of the MDS
$T_{Winding}$ Temperature of stator windings
$A_{WidningCoolant}$ Area between the stator windings and coolant liquid
$\rho_{WindingCoolant}$ Heat transfer coefficient between winding and coolant
$P_{StatorCoolant}$ Power from heat transfer from stator to the coolant liquid
T_{Stator} Temperature of stator
cp_{em} Specific heat capacity of the stator
$A_{StatorCoolant}$ Area between the stator and coolant liquid
$\rho_{StatorCoolant}$ Heat transfer coefficient between stator and coolant
$P_{WindingStator}$ Power from heat transfer from stator windings to the stator
$A_{WindingStator}$ Area between the stator windings and stator
$\rho_{WindingStator}$ Heat transfer coefficient between winding and stator
P_{thloss} Thermal loss
$m_{winding}$ Mass of windings
$cp_{winding}$ Specific heat capacity winding material
t_s Sample time
$cp_{Coolant}$ Specific heat capacity coolant medium
$m_{Coolant}$ Mass of coolant liquid

Contents

1
Introduction

The transportation sector is the fastest-growing cause of greenhouse gas emissions globally. It is now accounting for 17 % of the global emissions [1]. This is one reason Volvo Trucks is developing and deploying electric trucks. When developing the trucks, Volvo is currently using simulations based on physics-based models, which are reduced order models (ROM), to evaluate the performance and capabilities of the trucks. The ROMs are an efficient way of conducting many tests simultaneously, is more time efficient, and acts as a supporting factor when determining which components to use in the finalized products. In the electric motor (EM), surrounding factors affect these physically based models, which are challenging to incorporate into the current system. One solution for these problems could be to integrate AI models of the driveline components, which can quickly predict the output once trained on sufficient data.

AI is a popular trend at the moment. Much literature is being released, and more funding is directed at AI research. According to [2] there has been an apparent increase in machine learning over the last ten years, which does not seem to stop. One reason for the increasing use cases for AI technology is the funding for this research. Over the last years, the funding has exponentially increased [3].

With the gaining popularity and attention of AI and the possible potential, researchers are looking into more data-driven models and AI utility. Several drives are causing this growing trend, where one of the main factors is that data availability and the techniques of extracting data are increasing. The data can then be used to train the models and extract patterns and underlying relationships. This is desirable when applying to various parts of the electric vehicle driveline, for example, the thermal parts, which have very complex physical relations. When integrating AI into a ROM, some aspects have to be kept in mind since the type of model depends on the type of problem that is supposed to be solved. No solution fits all. However, one problem with AI integration is that it can be challenging to get an insight into how the AI works and its internal mechanisms of it [4].

Physical informed neural networks (PINNs) utilize prior knowledge about a systems behavior which many other machine learning/deep learning methods do not [5]. With physical knowledge, the network can be governed by the physical rule and constrain possible solutions. For example, the motor drive system (MDS) can not get indefinitely hot. Since the physical differential equations govern the network, the need for large amounts of high-quality data decreases due to the physical constraints.

The main topic studied in this report is the thermal model of the MDS in the Volvo trucks driveline, which is currently a black box model. This report will investigate the thermal model of the MDS using a Physical-Informed Recurrent Neural Network (PIRNN) in order open this black box. The PRINN is a hybrid approach (grey-box model) where initial and boundary conditions, together with physics, are incorporated into the neural network, which helps it predict the output of a system with less and more noisy data. The Recurrent Neural network is capable of storing elements and using those elements for unkown outputs.

1.1 Previous research

The previous research on Physical Informed Neural Networks are both very narrow and young. There is no clear way of implementing physics in machine learning and there are no right or wrongs. One of the first articles published about this subject is [6] in 2017, where Maziar Raissi is the author. He has since then published 3 other articles where the latest was in 2021, [6] [5] [7]. Most of the research available on PINNs today is refereeing to Raissi's work in some way. The main focus of PINN research is currently on proving that PINN works and can improve the area of neural networks for the better, where this thesis aims to contribute further and broaden the research of Physical Informed Neural Networks.

1.2 Problem description

Multiple factors must be considered when using software to simulate an entire driveline's behavior, computation speed, system robustness, the accuracy of the results, and the possibility of understanding how the system or components work. The problem with the current thermal model is that it is nearly impossible to understand how it works since it is a black box model.

Since the popularity of AI is continuously increasing, it is suitable for Volvo Group Technology to investigate the use cases for AI integration and learning for components on their driveline. This could increase the computational speed of their simulations, their system's robustness, and the results' accuracy. However, the time to train the network has to be kept in mind. For large datasets, the training time could be very high, reaching up to days, but if the model is accurate enough, it may be worth it. Thermal problems have been a problem for a long time due to complicated dynamics, but PINNs could solve these problems [8].

1.2.1 Scope

AI-based techniques are currently getting a large amount of attention. How viable are these techniques, and are they predicting sufficient and robust results? This project aims to create a PINN that could be integrated into the Volvo software models with good accuracy and quick computational speed while maintaining as

low network complexity as possible. The report should provide the reader with knowledge about the use case and suggest other components that could be effectively integrated with AI. The development time of AI technology from start to finish will be investigated and compared with developing AI methods from existing frameworks. The accuracy, computational speed results, and network architecture will be compared and discussed.

1.2.2 Purpose

This thesis aims to evaluate if a physics-informed AI methodology is a possible and viable way of modeling the thermal model of a drive system. A comparison between a regular AI methodology and a physics-informed is necessary to evaluate if a purely data-driven method is more viable than one with physics included.

2

Theory

2.1 AI, Machine Learning and Deep Learning

The area of AI and machine learning comes in many different forms and sizes. Machine learning is a subarea of AI, and Deep Learning is a subarea of Machine learning, as is represented in figure 2.1. Where ML aims to solve a specified task, two examples are Reduced order modeling (ROM) or image processing. At the same time, AI incorporates a broader view and includes what the name indicates, intelligence. The ability to learn to solve a new problem that was previously unknown. There is a discussion about the definition of intelligence. Hence the distinction between ML and AI is unclear. However, this chapter and project will only look at the area of Machine Learning. [9] [10]

Machine Learning can be used to learn a specific behavior of a system and mimic its characteristics based only on the inputs and outputs of that system. It can derive a pattern from a previously unknown data set and identify different classes. The problems that ML can solve could be divided into three groups. [9] [10]

1. **Classification:** Here the objective is to classify several data points. The main thing could be different numbers, shapes, and pictures, i.e., The different groups within the data are identified and assigned a classification or a name.

2. **Clustering:** Unlike a classification problem, a clustering problem aims to group data points with similar attributes. It does not require setting a classification or a label on those groups.

3. **Prediction:** The objective is to predict the specific behavior of a system based solely on historical data from that system.

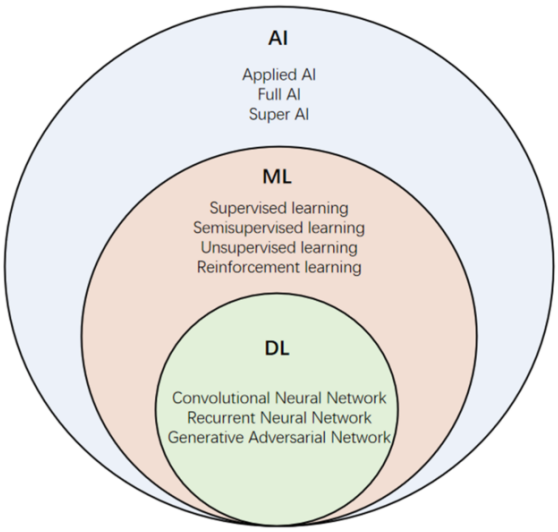


Figure 2.1: AI, Machine learning and deep learning from [11]

Further, Machine Learning can be divided into four different groups. *Supervised learning*, which presumes that both inputs and outputs are known and then trains the model to mimic this. *Unsupervised Learning*, does not know the answer beforehand but recognizes different patterns and sorts these into groups. *Semi-supervised Learning*, which combines supervised and unsupervised learning. *Reinforced Learning* where the model accounts for changes in the original surroundings. [9] [10]

2.1.1 Supervised Learning

Supervised Learning makes predictions of unseen data based on historical data. It uses big data sets where each input correlates to an answer or output. The model is then trained to mimic these data sets, becoming more accurate for each new training data set. After training, the supervised learned model can take input data not included in the training data and predict the output. [9] [10]

There are different applications for this, but they are usually divided into two groups; Classification and Regression problems. When solving a classification problem with Supervised Learning, the model is trained with, for example, pictures of different objects. The answer to what kind of object it is is known. The model is trained to recognize patterns and characteristics of the different objects to classify later objects where the answer is unknown. A regression problem is where prior sequential data is known for a system, and the model makes predictions on the future values of that system based on that preliminary data. [9] [10]

2.1.2 Unsupervised Learning

Unsupervised Learning is where the model is trained with data in which the answer is unknown. It recognizes different training data characteristics and then groups

them based on these charecteristics. The model does not know the label or name of these groups, just that they have similar characteristics. [9] [10]

2.1.3 Semi-supervised Learning

Semi-supervised Learning is the combination of supervised and unsupervised Learning. Most of the training data is unknown, except for a few of every group. The model is then trained and divided the data into different groups. Further, the model labels all the data points based on the few known ones. In this way, both supervised and unsupervised learning are combined to solve the problem. [9] [10]

2.1.4 Reinforced Learning

Reinforced Learning is where the model is trained to reach a predefined goal while accounting for a changing environment. For example, a chess-bot, where the model (chess-bot) makes the moves based on the current situation. This means that there is a new set of inputs for every prediction or move the model makes. The longer the model is trained, the more it can understand the long-term effect of its moves. [9] [10]

2.2 Neural Networks

Neural networks are a subsection of machine learning and can be implemented in supervised, unsupervised, semi-supervised, and reinforced learning. It utilizes so-called neurons and layers to learn a specific task. The learning is done with a large amount of data related to that specific task. Depending on the task, the network architectures can look different and it is a constant choice between different trade-offs. For example, if the network size, number of neurons, and layers are increased, the system can handle more complex problems while at the same time requiring more computational power.

There are a large number of different neural networks. One common neural network is a Feed Forward Neural Network (FFNN). Recurrent Neural Networks (RNN) is another fairly common network. Different networks all have their own advantages and disadvantages, where the choice depends on the application at hand. [9] [10]

Building a neural network demands that a set of parameters is decided, for example, the number of neurons (vertical) and layers (horizontal). There are different kinds of architectures in which these neurons and layers can be connected. The most common or traditional is the FFNN, where the input is connected to the first hidden layer, which is connected to the next layer until the output layer is reached. Figure 2.2 illustrates a FFNN. [10]

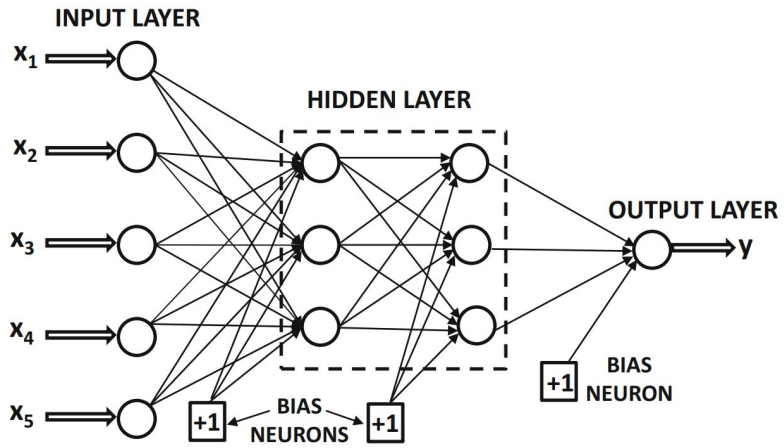


Figure 2.2: Feed Forward Neural Network [10]

In each neuron of a FFNN, there is an operation where all outputs from the previous layer are multiplied together with their respective weight values and then added together. This weight value is calculated and changed during training to remember specific patterns. More about this in section 2.2.1. The product of the sum of previous values and then the weight is put through an activation function which scales it to a smaller value. This operation is illustrated in figure 2.3. The output of the activation function is the output of that neuron which is passed on to the next layer. This process is repeated until the output layer is reached. [10]

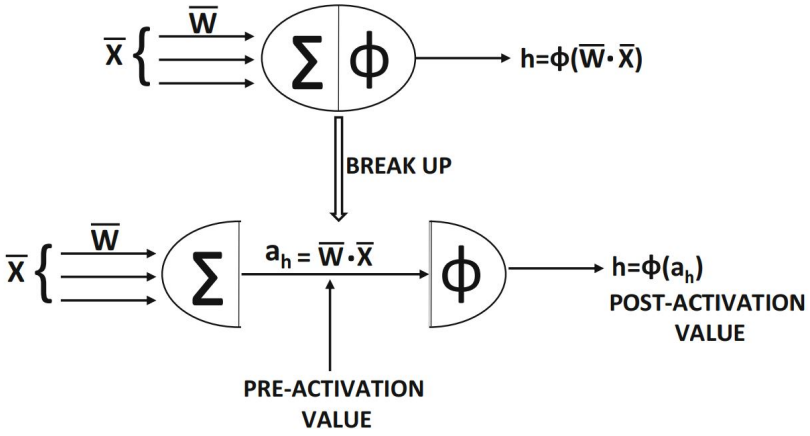


Figure 2.3: Operation in one neuron in a FFNN [10]

Besides the inputs and weights, the neuron can have another term called biases, which is added after all the inputs and weights are combined.

$$\text{output}_{\text{neuron}} = \phi\left(\sum_{n=0}^{i}(x_i \cdot w_i) + b_i\right) \tag{2.1}$$

This is represented in (2.1). Where $\text{output}_{\text{neuron}}$ is the output of the neuron, ϕ is the activation function, i is the number of neurons in the previous layer, x is the

inputs, w is the weight values, and b is the bias value. [10]

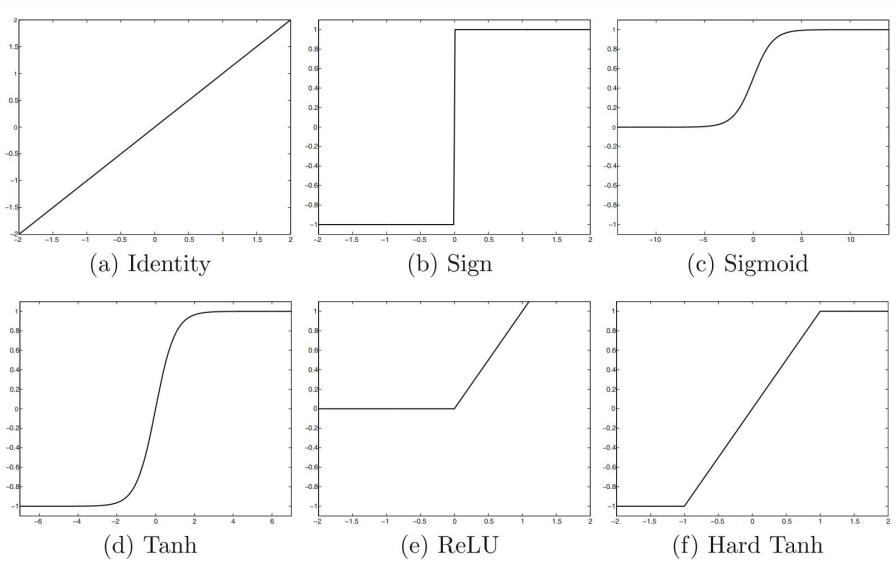


Figure 2.4: Visualisation of different activation functions [10]

Different activation functions can be used depending on the system the network should mimic. For example, the Tanh and Sigmoid functions are good when predicting non-linearities, whereas the Tanh is used when there could be negative values in the data set. The Hard Tanh and ReLU (Rectified linear Unit) functions are similar to the Tanh and Sigmoid. An advantage of these functions is that the training time is reduced. Visualization of different activation functions is shown in figure 2.4. The FFNN just discussed, calculates an output based on a set of inputs. This output is calculated solely on these inputs and does not consider previous inputs. For some systems, this can be a problem, for example, when modeling times-series data or interpreting text with several words that by themselves do not mean anything but, when put together in order, forms a sentence. [10]

The RNN addresses this and incorporates a memory part from previous inputs. The RNN takes an input and calculates an output, similar to the FFNN. Nevertheless, unlike the FFNN, each neuron in the RNN has one more output and input. One takes the output of that neuron from the previous time step and feeds it as an input. The other feeds the output to the next time step. It can be seen as a loop. In figure 2.5, a basic RNN is represented, where A is the entire network, h is the output, t is the current time stamp, X is the input, and tanh is the activation function. In this way, the network can predict the output at time t based on the prediction made at time *t-1*. At any time, t, the prediction will be made on the prediction from *t-1*, along with all the previous predictions. In this way the RNN can handle sequential- and time-series data, which is why it works well for dynamic systems. However, one problem with a basic RNN is its capability to remember long-term dependencies sufficiently enough. Every time a new time step is taken, the trace from the previous *t-1* gets smaller and smaller. The prediction from *t-1000*

will still be there at time t, but the impact on the prediction will be minimal. This is called vanishing gradients. [10]

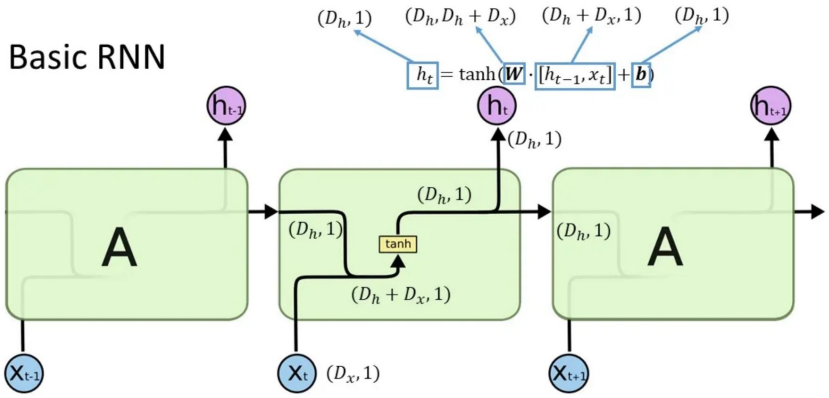


Figure 2.5: Visualisation of a Basic RNN [12]

One way to solve this is through a Long-Short Term Memory (LSTM) or Gated Recurrent Unit (GRU) network, a version of RNNs. The LSTM network can remember previous outputs, like the RNN, but it can also forget or reduce the impact of previous outputs. This is done by having a *cell state*, *forget gate*, an *input gate*, and an *output gate* within the neuron. The cell state remembers the previous cell state. The forget gate determines how much of the previous cell state is to be remembered and uses a sigmoid function to do this. The function outputs a value between 0 and 1, where one is essential, and 0 is unimportant. The input gate determines how much the current input should impact the cell state. [10]

Moreover, like the forget gate, it utilizes a sigmoid function to do this. The output gate regulates how much of the cell state should go to the output. This is made with a sigmoid function.

The GRU works in a similar way to the LSTM but has a less complex architecture with fewer inputs and outputs. The GRU has a *relevant gate*, *forget gate*, *input gate*, and a cell state that is combined with the output. The relevant gate determines what is worth keeping from the previous cell state. The input gate and forget gate determine how much of the new input should be stored. A visualization of an LSTM and GRU can be seen in figure 2.6, where σ is a Sigmoid function and tanh is a Tanh function. Both the sigmoid and Tanh function is shown in 2.4. The GRU has a faster computational time due to its simpler architecture compared to the LSTM. It can handle long-time dependencies, but not as well as the LSTM. So the GRU is suitable when trying to model systems with relatively short-time dependencies. [10]

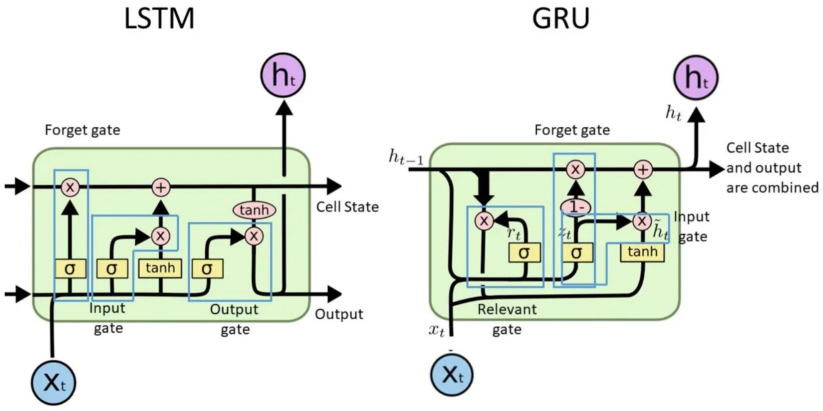


Figure 2.6: Visualisation of a LSTM and a GRU [12]

2.2.1 Training

When a neural network is trained, it aims to mimic a system based on data from that system, both input, and output. One basic component for this is backpropagation. It can be described as a forward and backward phase. In the forward phase or forward pass the input goes through the network and produces an output, the output is compared with the correct value for that set of inputs.

$$\mathcal{L} = \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.2}$$

The comparison is done with a loss function, one popular loss function is Mean Squared Error (MSE) and is represented in (2.2), where $\mathcal{L}$ is the loss function, n is the number of input sets, y_i is the true value and $\hat{y}_i$ is the predicted value. [10]

The next phase is the backward phase and the goal is to minimize the loss function by altering the weights and biases. When using backpropagation the minimization is made by calculating the derivative of the loss function with respect to the weights and biases,

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ij}} \tag{2.3}$$

$$\frac{\partial \mathcal{L}}{\partial b_{ij}} = \frac{\partial \mathcal{L}}{\partial z_j} \cdot \frac{\partial z_j}{\partial b_{ij}} \tag{2.4}$$

This is made through the chain rule and is shown in (2.3), where $\partial\mathcal{L}$ is the partial derivative of the loss function, ∂w_{ij} is the partial derivative of the weight of the i neuron in the j layer, ∂z_{ij} is the partial derivative of the input to the activation function in (2.1) of the i neuron in the j layer. The same is done for the bias term according to (2.4). [10]

$$w = w - \eta \cdot \frac{\partial \mathcal{L}}{\partial w_{ij}} \tag{2.5}$$

$$b = b - \eta \cdot \frac{\partial \mathcal{L}}{\partial b_{ij}} \tag{2.6}$$

Then all the weight and biases are updated by subtracting the partial derivatives from the previous value. This is represented in (2.5) and (2.6), where η is the learning rate. Backpropagation is one basic way of training a network, but there are a large amount of more advanced optimization algorithms that are built upon backpropagation that can be used as well. One popular is the Adam optimizer and its variants like AdamW. Another one is the SGD. The choice of optimizers depends on the task at hand and is often chosen based on trying different optimizers and taking the best-performing one. [10]

One problem when training the network can be overfitting the training data. Overfitting is when the network learns the training data set too well and fails to handle new data outside the training data set. It fails to generalize and does not mimic the system, only the training data set. There are a lot of different strategies to prevent overfitting. One example is having a dropout layer, where randomly selected neurons will be removed/turned off or dropped out. In this way, the network can not rely on one specific path but must generalize to minimize the loss function. Another way to prevent overfitting is to incorporate an early stop in training, which will stop the training process if the network has stopped minimizing the loss function on the new data. This is done by simultaneously testing the network on new data as it is trained. The new data is usually called validation data. [10]

Another parameter that needs deciding is how the network will receive feedback or backpropagate during training. There are many different techniques, and here, an educated guess will be made and tuned through testing. Here there is a difference between the traditional and cross-coupling networks. The traditional is harder to train, i.e., slower, since the back-propagation or feedback needs to travel through more neurons. However, one advantage is that it is easier to write the training code for the traditional network than cross-coupling.

2.2.2 Pretraining

Pretraining is a method that could be useful to increase the robustness of the neural network [13]. Like the human brain functions, pre-trained neural networks can utilize previously known knowledge about a system to handle tasks within the system better [14]. In the case of this report, one example of pretraining could be to utilize vehicle log data which is rather noisy. However, much data is available to pre-train the network and make it learn the general behavior of the thermal system. Then fine-tune it with training on GSP data to accurately estimate the outputs from the thermal model. In this case, winding temperature and thermal losses.

2.3 Modeling

The current model for the entire driveline is built in Simulink with some exceptions, with Python or C code. The models are based on physical equations, where the primary inputs are torque request, speed, and voltage. With the help of physical functions, the current and power is then calculated. From theory, this is the opposite of traditional modeling, where a current, voltage, and speed are input into a motor, and torque is an output.

According to [15], the concept of a digital twin has had some different definitions and is referred to in different ways; it was described as a "probabilistic simulation of a vehicle" by [16] which is closely related to the goal of this thesis. A summation of benefits with a digital twin is concluded in [15], which describes increases in productivity and quality while reducing time, cost, and complexity. For the automotive industry, there are multiple stages of a digital twin, from a prototype of a new vehicle to a simulation twin and, at last, a reuse twin, which is a way to use data to conclude performance [15]. The PINN system being developed for a thermal model is an example of a Simulation twin software upgrade, where the developers upgrade the current software for digital twins in use.

One significant reason for using simulation modeling when constricting vehicles is that the overall production time decreases, the cost decreases, and enables quality improvements. Simulations enable multiple tests to run simultaneously for different types of drive cycles to determine the vehicle's performance and possible changes that must be made. If a change occurs, it is more economically efficient to do so in the development phase, in the simulation environment, than for an actual test vehicle.

Currently, numerous advanced software tools exist designed to simulate entire drivelines. Simulink, a simulation and model-based design software, is a prevalent choice among engineers and researchers due to its user-friendly interface and versatile functionalities. In addition to Simulink, there are multiple other tools available for Simulink. The user can combine it with other programs or scripts, as well as multiple toolboxes that simplify the use of example, neural networks for the user. Maxwell2D is an example software tool for the virtual analysis of motors designed by the user. It employs the Finite Element Method (FEM) to model the motor's electromagnetic properties and simulate its behavior under different operating conditions. The results obtained through such simulations can provide valuable insights into the motor's performance and heat distribution and can aid in its optimization.

Utilizing these advanced software tools has significantly enhanced the design and development of drivelines. By employing virtual simulations, engineers and researchers can analyze various driveline components under diverse operating conditions, thereby identifying and mitigating potential issues. Furthermore, these tools allow for the efficient optimization of driveline designs and evaluation of their performance before their physical implementation.

2.4 Driveline & components

The driveline of a fully electric truck consists of various interdependent components, including the Energy Storage system that provides electrical energy to power converters, charge unit, motor drive system, shafts, gears, auxiliaries, and wheels. Together, these components facilitate the efficient functioning of the vehicle. Despite their high-level nature, each component has a complex underlying structure. For instance, the Energy Storage system requires State of Charge (SOC) estimating software and State of Health (SOH) estimation mechanisms to ensure optimal performance. Similarly, the converters necessitate regulating systems that control the motor's power output. In a driveline system, efficient power transfer from the source to the load is critical for achieving optimal performance. However, all the components within the driveline suffer from various types of losses, which can significantly impact the system's overall efficiency.

One of the sources of loss in the driveline is mechanical losses in the drive shafts. Transferring torque through the shafts involves friction and other forms of resistance, resulting in a loss of power. The physical properties of the drive shafts, such as material and design, can affect the amount of energy lost through mechanical losses. Another source of loss in the driveline is the switching losses of the power converters. Power converters are used to convert the electrical power from the battery into a suitable format for the load. During the switching process, power is dissipated in heat, resulting in a loss of efficiency. Magnetic losses within the motor contribute to the losses in the driveline. Motors rely on the interaction between magnetic fields to produce torque and rotational motion. However, these interactions are imperfect, resulting in some energy loss as heat. The magnitude of these losses depends on the motor's design, the material used in the construction of the magnetic cores, and the quality of the electrical insulation.

In addition to the above sources of loss, thermal losses are a factor in most of the components in the driveline. When a current flows through a component, it generates heat, which can cause inefficiencies and potentially damage the component if not dissipated adequately. This is why thermal models are pretty challenging to model. It is here that AI models (PINN in this case) potentially could help due to the ability to map the underlying system based on labeled input, output, and differential equations. In this instance, given the advanced level of development of the MDS, this component will be the central focus of the study. Specifically, the thermal model within the MDS will be subject to a more comprehensive and in-depth analysis.

Traditionally physical models in Simulink are built in the same direction as motor drive rigs. However, at Volvo, the Simulink model is in the opposite direction. Instead of inputting a current and speed into the motor drive while applying a voltage over it and getting a torque, this model takes a torque request, applied speed, and voltage and then outputs a current. This report will model the thermal model subcomponent in the same direction as the current Simulink model. It will depend

on the torque, speed and temperatures, and coolant flows.

2.4.1 The motor drive systems thermal model

The current thermal model for Volvos MDS is a black box model provided by a supplier. It takes the inputs into some unknown matrix and then provides outputs based on the matrix. The exact equations for the model do not exist since the matrix used is based on FEM analysis. This is a working solution, but since it is a black box, it is difficult to understand what it does and physically model it into the virtual drive line.

According to [17], there is no extensive research and testing on applying neural network models for temperature predictions of PMSMs. Furthermore, the traditional finite element method can only predict temperature based on linear data from the current, which is why Neural networks are looked into, since there is a possibility to model nonlinearities in the historical data [17].

2.4.2 Other components suitable for AI integration

The battery is an essential part of the driveline of electric vehicles. Due to the complexity of the material in the battery and the difficulties in modeling the electrical and thermal dynamics, AI integration is suitable for modeling this component. When doing equivalent circuit modeling, commonly used in battery management systems (BMS), it is crucial to balance the accuracy against the model complexity [18]. One of the most critical tasks the BMS has is to accurately estimate the different states of the battery to ensure that it is being used safely. However, this is difficult due to the batteries' complexity and nonlinearities [18]. The primary states that could affect the vehicle's performance are the SOH and SOC; as mentioned earlier, these have to be accurately estimated to ensure the safety of the person operating the vehicle.

Another example of neural network implementation is in [19], where a feed-forward neural network is built to control the voltage level of a DC/DC converter, where the network outperforms better than a classic linear controller.

As mentioned earlier, when it comes to AI integration and implementation, no one model fits all problems; instead, for each problem, a unique AI model has to be developed with different amounts of complexity. The more complex the problem is, the more advanced the neural network has to be to cope with complex relations and nonlinearities.

2.5 PINN Implementation and Possibilities

The main difference between the Physically informed neural network and a regular neural network is mainly the loss function, where the physics are implemented. In the loss function governing equations, boundary, and initial conditions can be

added [20]. Instead of being purely data-driven like other neural networks PINNs take the underlying physics into account; however, to create a PINN, one has to know the physics [20]. The amount of literature on real case applications is scarce, and it has yet to be done for PMSMs, which makes this project unique. There are some application concepts in other areas; in [21], a PINN is used to determine the dynamical states of the power system, e.g., rotor angle and frequency. The reason for the implementation is that a PINN does not require as much data as other neural networks, which enables a reduction of the network complexity. In the case of [21], the PINN can compute the dynamic behaviors 87 times faster than the current numerical calculations.

2.6 Thermal Modeling

There are three types of heat transportation within the PMSM, conduction, convection, and radiation [22]. Conduction is when two elements are close to each other, ex, within the stator where the laminated copper wires generate heat due to resistances in the wires and conduct it to the motor housing. Both the hysteresis losses and eddy currents generate heat that conducts through the rotor shaft. Heat convection occurs due to some fluid or gas transporting the heat. In the stator, the air can conduct heat away from it. However, in this thesis, the thermal model is based on Forward Euler estimations that have been matched with the driveline's current black box thermal model. The Simulink model is converted to forward Euler equations to enable implementation in the RNN, converting it to a PINN.

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{2.7}$$

Algorithm 1 General forward Euler algorithm from source [23]

Inputs:
y_0, t_{end}
Initialize:
$t_0 = 0, y_0$
for $n = 0 : t_{end}/h$ **do**
$t_n = nh$
$y_{n+1} \leftarrow y_n + hf(t_n, y_n)$
end for

Algorithm 2 Forward Euler to estimate winding temperature in the thermal model, here x contains winding temperatures

Inputs:
$t_s, Heat_{loss}, x_{vector} = [x_0, ...]$
Initialize:
$x_0 = T_{ambient}$
for i:N **do**
$\dot{x} \leftarrow f(x_0, Heat_{loss}(i))t_s + x_{vector}(i)$
$x_{vector}.append(\dot{x})$ ▷ append adds to the list
end for

The forward Euler equations can be found in section 3.1.1. The general forward Euler can be seen in (2.7) and its algorithm is shown in algorithm 1. The case for this thesis can be described with algorithm 2, where the winding temperature can be estimated based on constants, the current winding temperature, and the power loss from the thermal model.

3
Methods

This section will describe the method that was used to produce results. The models was created using Python [24] and PyTorch [25]. At first, a simple Recurrent neural network was created, then a more advanced Long-Short-Term-Memory- and Gated recurrent unit RNN was designed. This is to be able to compare the performance of the different networks to provide a sufficient recommendation on what Volvo should focus on regarding AI implementation on their virtual driveline. Then governing physics on how the thermal model behaves was added to the more advanced RNN to form a PINN. Furthermore, Volvo already has an existing framework for training and creating neural networks. This was also added in the comparison of performance between different simulation methods. Various parameters were used in the investigation based on the end application and user.

3.1 Pre-calculations

Different parameters and data can be acquired by measuring parts within the driveline. When generating data from the virtual driveline, the output from the thermal model is both power loss, winding temperature, and temperatures from other hot spots. In this investigation, the power loss and winding temperature will be predicted. The data for winding temperature from the test vehicles are more scarce because measurement equipment is rather expensive.

$$P_{in}(t) = u(t)i(t) \tag{3.1}$$

$$P_{out}(t) = T(t)w(t) \tag{3.2}$$

$$P_{thloss}(t) = (P_{in}(t) - P_{out}(t)) \tag{3.3}$$

However, in that case, one could use the Voltage, Current, Torque, and Speed to estimate the winding temperature. In the field test data, the power loss is not logged directly. However, the input current and voltage can be used to calculate the input power. The power output can also be calculated from the logged data since torque and motor speed are logged. This is done by (3.1) - (3.3)

In order to increase the accuracy of the neural network, physical calculations of the winding temperature and power loss will be fed into it in order to provide more information to the network. This was then compared with the accuracy of a network without this information, in order to be able to evaluate how the physically informed network performed compared to the standard network. From physics, the winding

temperature can be calculated based on the power loss.

There are test rig data which is more detailed, where a vehicle is using a virtual cycle, and forces are applied precisely how they would be in reality. This equipment is installed with more data collection sensors, which gives more data and a higher quality of data. In the case of creating an entirely data-driven model of components within the virtual driveline, this type of data would be sufficient since the data is created by the same drive cycles. When a model is precise, the gap between simulations and actuality decreases, resulting in more reliable outcomes that can be utilized with greater assurance. Another advantage is that the simulations will be faster due to the data-driven model. With a data-driven model, calculations are unnecessary; the model will get input and use the weights in the network to predict an output.

3.1.1 Forward Euler calculations

From the forward Euler method, the winding temperature, stator temperature, and coolant outflow temperature can be calculated based on the coolant inflow temperature, the coolant flow, and the power loss from the thermal model. In this case, the coolant inflow, temperature, and ambient temperature are assumed to be constant at 18 l/min, the coolant temperature set to 50 degrees Celsius, and the ambient to 25 degrees Celsius. This was done to ensure that the forward Euler method provides an accurate and similar representation of the thermal system.

$$P_{WindingCoolant} = (T_{CoolantOut} - T_{Winding})A_{WidningCoolant}\rho_{WindingCoolant} \tag{3.4}$$

$$P_{StatorCoolant} = (T_{CoolantOut} - T_{Stator})A_{StatorCoolant}\rho_{StatorCoolant} \tag{3.5}$$

$$P_{WindingStator} = (T_{Winding} - T_{Stator})A_{WindingStator}\rho_{WindingStator} \tag{3.6}$$

$$\dot{T}_{Winding} = \frac{P_{thloss} + P_{WindingCoolant} - P_{WindingStator}}{m_{Winding}cp_{Winding}}t_s + T_{Winding} \tag{3.7}$$

$$\dot{T}_{Stator} = \frac{P_{StatorCoolant} + P_{WindingStator}}{m_{Stator}cp_{em}}t_s + T_{Stator} \tag{3.8}$$

$$\dot{T}_{CoolantOut} = \frac{(T_{CoolantIn} - T_{CoolantOut})Flow * cp_{Coolant} - P_{WindingCoolant} - P_{StatorCoolant}}{m_{Coolant}cp_{Coolant}}t_s + T_{CoolantOut} \tag{3.9}$$

The winding temperature was then used as an input into the neural network which could use the calculated value as a guideline to predict a more accurate value.

3.2 Choice of AI methodology

From the theory, RNNs were selected since the data consists of sequential time series data. A few tests were made to see which type of recurrent neural network works

best, where the hyperparameters were held constant, and the kind of network was exchanged from LSTM to GRU. This was done to ensure that the best possible structure was in use, and from the literature, it is hard to conclude which network type is suitable for each use case.

3.3 Early stage testing

The first step was to create an RNN to see performance and estimate the time needed to develop and train a model. This ensures a sufficient recommendation regarding what Volvo should continue to investigate and implement into its current software. Multiple RNNs were created with increasing complexity. The added complexity was that instead of recurrent nodes adding LSTM or GRU nodes that can remember previous outputs. Other complexities were the network size (number of neurons and hidden layers). However, the training time and accuracy did not always increase with the increased performance.

A first approach to integrate the physics into the network was to calculate the values that the model should predict and then compare the predicted values to the calculated ones. From this comparison, a second loss term would then be added to the loss function together with the original loss. However, it was concluded that the same effect could be obtained by increasing the learning rate.

3.4 PINN

3.4.1 Case setup

The physically informed neural network consists of two essential parts. The first one is the LSTM or GRU recurrent neural network, which trains and learns upon given inputs and outputs. The second is the governing physics within the system, which is added to the model's training process.

The PINN operates and trains similarly to the RNN. It compares the output of the model with an already-known output. This comparison's MSE produces a specific loss representing the model's accuracy. From this loss, the influence of each neuron is calculated and changed accordingly, as explained in section 2.2.1.

To integrate physics principles into a neural network, it is first necessary to establish a robust and comprehensive understanding of the underlying physics. In thermal modeling, multiple phenomena are at play, including heat transfer and generation. However, since heat-related physics is very complicated, it will be simplified. One characteristic of the temperature is that it can not change immediately, it has a dynamic response to changes in different parameters. Hence if the model would predict that the temperature changes too quickly than what is physically possible, the training process should penalize this in the loss function. This was done with a maximum gradient constraint on the temperature.

$$\mathcal{L}_{RNN} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.10}$$

$$T_{gradient} = max(\hat{y}_{winding}(t+1) - \hat{y}_{winding}(t)) \tag{3.11}$$

$$Max_{gradient} = max(y_{winding}(t+1) - y_{winding}(t)) \tag{3.12}$$

$$Gradient_{penalty} = mean(T_{gradient} - Max_{gradient}) \tag{3.13}$$

$$\mathcal{L}_{total} = \mathcal{L}_{RNN} + \Gamma Gradient_{penalty} \tag{3.14}$$

The temperature gradient ($T_{gradient}$) was calculated according to (3.11), where the slope between each value in the estimated winding temperature ($\hat{y}_{winding}$) is calculated. The max temperature gradient is calculated in the same way but with the true values ($y_{winding}$) instead. The gradient penalty was calculated by taking the mean of every value where the temperature gradient exceeds the max gradient. The total loss was calculated by the loss from the model added with a tuning factor Γ and the gradient penalty variable according to (3.14). This implementation aimed to reduce unrealistic oscillations in the predicted output. The tuning factor Γ was added as a hyperparameter to tune the effect this slope error should have on the loss function.

The forward Euler approximation together with the heat loss calculation is not 100 % accurate when predicting the output, unlike the output from the simulations from Simulink, which is assumed to be the true value. However, the calculated values were used as input into the network to evaluate the effects of the physical information. The aim of this implementation was to give the network some idea about how the characteristics of the output behave.

The network was created using the PyTorch library, which is built on TensorFlow and is used to simplify the modeling of neural networks. Here the network structure and forward function is defined. The training and validation are done simultaneously, while the testing is done afterward, at this stage, the MSE values are being outputted, and where the performance of the network can be determined. The mean average error (MEA) and root mean square error (RMSE) values are calculated to determine the accuracy of the model that is being evaluated.

The model for creating training, validation, and test data is all-physics-based. The thermal part of this model is based on FEM calculations, whereas in Simulink, it is a black-box model from the suppliers.

3.4.2 Data

The data is based on drive cycles that are available in GSP. The results from the Simulink model are then cut into the first hour of the cycles. The cycles are then randomly selected as training, validation, and test cycles. Out of 15 cycles, 12 are

training cycles, two are for validation, and one is for testing (80 % training, 13 % validation, and 7 % testing data). This split is done to be able to train on as much data as possible due to the limited amount of acquired data. The cycles then get an ID which will ensure that the training cycles do not overlap, which could cause issues due to the reset between each cycle. The problem with the reset is that the final winding temperature could be 135 degrees Celsius, and the first value of the next cycle could have a winding temperature of 25 degrees Celsius. If the cycles are not divided correctly, the network will try to catch this unrealistic behavior and create oscillations on the predicted values. The data is standardized (mean = 0 and standard deviation = 1) to remove units from the data and improve the training process. The standardization is done to ensure that each input/output (feature) will have a similar impact on the model [26].

Since each drive cycle contains different operating points, the network will perform better if the training data set contains operation points over the entire possible range of the thermal model. Therefore, switching the cycles between test and training is essential to see how it affects the results.

3.4.3 Hyperparameter test

Multiple parameters must be selected to acquire a good network to train and learn the system's dynamics while maintaining an excellent ability to generalize new data. All of the parameters are tunable to certain extents. To make sense of all the parameters, a test where a standard network was selected and the other parameters were changed individually to determine the contribution to the network accuracy. A summation of the hyperparameters incorporated can be seen in table 3.1.

Table 3.1: Settings for the Physics informed neural network

Hyper Parameter	**Explenation**
Network Type	Type of Recurrent neural networks, such as LSTM or GRU
Activation Function	Activation nodes on each of the neuron layers, ReLU or tanH
Epoch	Amount of times the entire dataset is trained on (sent through the training and validation loop)
Learning Rate	The magnitude of adjustments made to the weights during training, with a higher learning rate resulting in more significant weight updates per iteration
Batch Size	How many samples in the dataset the network trains on before adjusting the weight of the neurons within it
Hidden Dimensions	The Width (number of neurons in each layer) and the Depth (number of hidden layers) of the Network
Drop out	The probability of turning off a neuron within the network
Weight decay	A regularization factor for the neurons within the network
Early Stopping Patience	How many times the validation loss is allowed to be larger than the best validation loss value before stopping the training
Early Stopping Delta	How much the validation loss has to be decreased to count as a reduction
Learning Rate Decay factor	A factor of how much the learning rate decreases after a set condition (ex. if validation loss has not decreased in 4 Epochs)
Physical Weight (Γ)	How much the temperature gradient slope will have an impact on the loss function

3.4.4 Physical effects test

Several tests were conducted to evaluate if physics contributes to more accurate results, faster training, and the potential to train on fewer data. The tests were made on similar networks to generate comparable results.

3.4.4.1 Performance

The first test was divided into three parts to see the performance effects of integrating physics into the neural network. The first test was to train an RNN without physics. The next test was only to input the pre-calculated values for the winding temperature and the power loss. The final test was to incorporate a gradient constraint on the temperature gradient, which will prohibit oscillating behavior. To ensure that the range of the data set was adequate, the test cycle was exchanged to another cycle, and the same test method was conducted again to compare the results and see if it had an impact.

3.4.4.2 Lower Amount of available data

The next test evaluated the network's ability to predict the winding temperature and power loss with significantly reduced data. Instead of utilizing the 12 drive cycles to train the network as the previous test did, this test used five cycles to train on, one to validate, and one to test on. This was done for the test that performed best of the two earlier performance tests.

3.5 Framework - Nerve

A team at Volvo has created a general Framework, similar to PyTorch, Keras, and Tensorflow, upon said packages. This framework is mainly developed for users at Volvo that want to conduct simulations and tests with AI estimation. However, the framework is in the development phase and is not developed for physically informed neural networks, but it estimates how a neural network should perform.

Regarding the scope of this thesis, there are other aspects to consider. For Volvo to further integrate AI and machine learning into its current system, this framework can help simplify the process. From a user's perspective, it is mostly about acquiring and prepossessing the data instead of developing a complete neural network and determining which structure architecture should be used. The advantage of using the Nerve Framework is that the user can have limited experience with AI technology, machine learning, or neural networks. However, to further developing it or tuning it for specific use cases could be a requirement.

Nerve can train on many different concepts (Trucks) simultaneously and is being developed to act as a helping hand when selecting vehicle components.

3.5.1 Case Setup - Nerve

The Nerve Framework approach differs slightly from when developing an entirely new neural network. In Nerve, the network used to solve tasks is Transformers with some conventional layers, which differs from the network developed from scratch.

Since it is pre-developed, there are also more difficulties with integrating physics into the neural network. The models built by the framework are currently entirely data-driven, which makes estimations of non-measured values impossible.

The Nerve framework works similarly to the original script that was created. However, there is more prepossessing needed. The framework only accepts certain types of files, which also need to have certain information in them. Since the data is a time series, a time vector must be included. The framework then loads all the data for each concept/cycle and slices it into batches of chosen size, 128 samples for GSP data. However, this depends on how much data is being used. In the case of field test data where there are possibilities to acquire high amounts of data, the slice size should be increased to speed up the training process. When slicing the data, it is only possible to present the results in those slices, indicating how well the model performs at a randomized place in a random drive cycle.

The user then defines the training process, where batch size, epochs, learning rate, etc, is set. The type of network is already predefined as a transformer network, as mentioned above. However, the user selects the architecture, in this case, six hidden layers with 256 neurons in each layer. In this case, the neural network training should be quick when the prediction is limited to one motor for one type of electric truck. Furthermore, with this framework, estimating more components in the MDS is possible. If enough data is acquired, the entire could be estimated by a neural network, which will be an entirely data-driven model of the system.

The network aims to accurately predict the winding temperature output and the thermal model's power loss. The power loss can then be used with the forward Euler formulas described previously and the coolant flow and coolant in flow temperature to estimate the stator and coolant outflow temperatures. The winding temperature can also be calculated from the power loss, which will then be compared to the predicted winding temperature. The stator temperature estimation should be correct if the estimated winding temperatures are similar.

If the trained model does not perform reasonably accurately, there are options to increase the performance. Since the data is limited by the number of drive cycles in GSP, one alternative is to take the generated results from GSP and introduce a random error. When introducing this error, the model should be trained to be more general than for each specific cycle, preventing overfitting and increasing the possibility of more accurate correct complex behavior. Another option is to pre-train the model. This is done by introducing a general dataset and then training on a specific task dataset, in this case, the dataset for the thermal model. This is effective when the task-specific dataset is limited. Hence it is a viable solution in this case.

3.5.1.1 Tests

The tests set up had some differences; from the first to the second test, the amount of data was increased from 4.6 to 5.6 million data points (21.7 % increase in the amount of data). The network architecture for the cases had the same setup of 6

hidden layers and 256 neurons within each of the layers. The setup for Nerve is that the input is the same as in the PINN case (Motor speed, Torque, Voltage, and Current), but in this case, only one output is predicted, which means the network is trained on two separate data sets. One data set predicts the power loss for the thermal model, and one predicts the winding temperature. The settings for the nerve tests can be seen in table 3.2.

Table 3.2: Settings for the tests of the Nerve framework

Settings	**Values**
Batch size	128
#Epochs	128
#Validation batches	12
Learning Rate	1e-4

4

Results

4.1 PINN

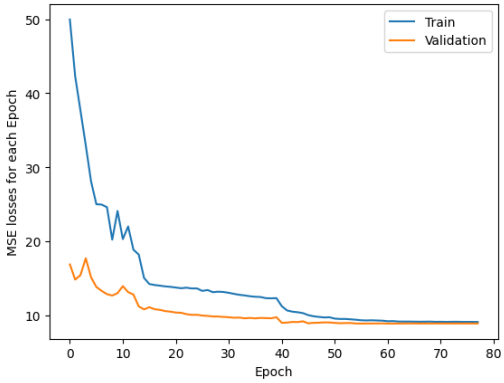
4.1.1 Network Structure

The first part of building a neural network is selecting the appropriate network type and size. Due to the dynamic behavior of the thermal model, a regular RNN could not suffice. The network type was then a hyperparameter, either an LSTM or a GRU, where the one that performed best was selected. Both of the networks had the same hyperparameters.

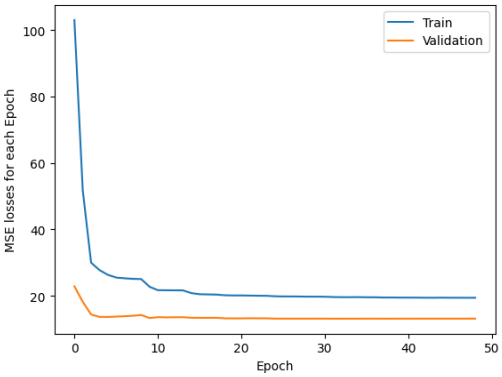
Table 4.1: Settings for comparison between GRU and LSTM

Hyper Parameter	**Setting**
Network Type	GRU/LSTM
Activation Function	TanH
Epoch	200
Learning Rate	1e-3
Batch Size	10 % of a whole cycle
Hidden Dimensions	[512,256,256,128,128,256]
Drop out	50 %
Weight decay	1e-5
Early Stopping Patience	16
Early Stopping Delta	1e-6
Learning Rate Decay factor	50 %
Physical Weight	5

When evaluating the training and validation, the main result from GRU compared to LSTM is that the training loss converges with the validation loss, which is a desired behavior, since the network fits well with the training data and generalizes well with new data. This can also be seen in figure 4.2a and 4.2b, where the general solution to the test set is predicted better by the GRU network than by the LSTM network.



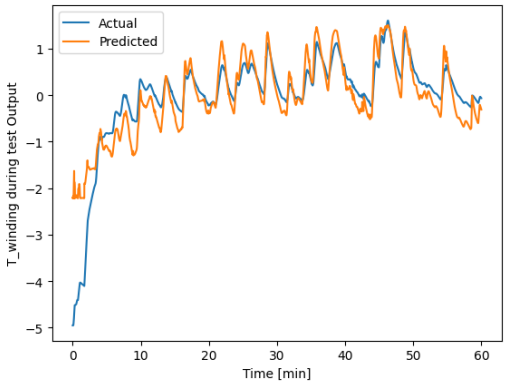
(a) Training and validation for GRU



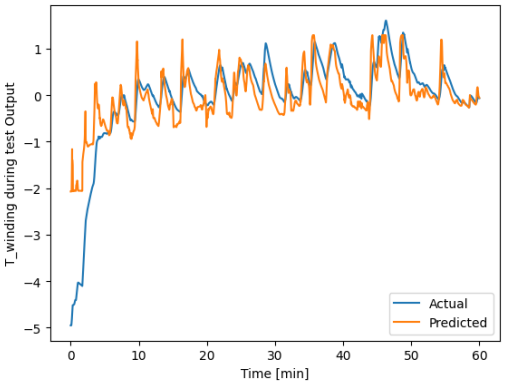
(b) Training and validation for LSTM

Figure 4.1: Training and validation of GRU and LSTM

From figure 4.2a and 4.2b it was found that GRU performed better for this system. As can be seen in the figure, the predicted value fits better for the GRU network, where oscillations are more dampened, and the high peaks are limited. When comparing these results, the system dynamics are out of the highest importance. The fast switching temperature seen in the LSTM network is not possible in reality, which is why the GRU performs better. In figure 4.2a and 4.2b the *Actual* is a representation of the Simulink model which is based on FEM calculations and the *Predicted* is the output of the PINN.



(a) Winding Temperature prediction with GRU



(b) Winding Temperature prediction with LSTM

Figure 4.2: Performance of GRU and LSTM

To find the importance of the width and depth of the network, a standard network was compared with a narrow, wide, shallow, and deep network. From the test, it was found that good architecture should be both quite deep and wide. The other hyperparameters were tuned simultaneously, where a test network was compared with other networks where only one hyperparameter was changed and the rest held constant. In the case of this system, the depth and the width had high importance for the network to catch the dynamics. The dropout, early stopping, and learning

rate decay were used to prohibit the network from overfitting toward the training data.

4.1.2 Comparison of normal GRU with PI-GRU

The following results compare the network behavior with and without physical guidance. The hyperparameters are defined in the table below. The different tests are one standard GRU network without physics as inputs, one with physics as input but without the gradient constraint of the temperature gradient (Physical Weight = 0), and one with both physical calculations as an input and the temperature gradient active.

Table 4.2: Settings for comparison between regular GRU, physics-informed GRU, and physics-informed GRU with gradient constraint

Hyper Parameter	**Setting**
Network Type	GRU
Activation Function	TanH
Epoch	200
Learning Rate	1e-3
Batch Size	10 % of a whole cycle
Hidden Dimensions	[1024,512,256,128,256]
Drop out	50 %
Weight decay	1e-5
Early Stopping Patience	12
Early Stopping Delta	1e-6
Learning Rate Decay factor	50 %
Physical Weight	10 or 0

4.1.2.1 Performance

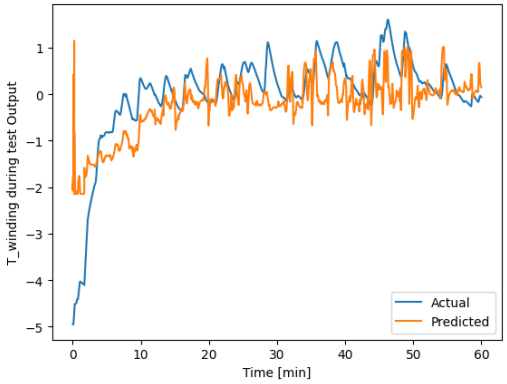
It is clear from the training and testing results that the networks that utilize physics, and data, have higher accuracy on the predicted winding temperature and power loss. The oscillating behavior is dampened when integrating physics, and the initial prediction does not deviate as much as the network without physics. In order to see the efficiency of incorporating the gradient constraint as well, the graphs have to be analyzed. It can be seen in figure 4.3 that there is a range problem with the data since the amplitude of the power loss is not reaching the top amplitudes of the test dataset. If the training data had a higher range, as in performance test 2, the prediction would be more accurate since the test data is within the training data range. The tables 4.3 and 4.4 shows that the networks for the second test learn the dynamics of the system much faster and with higher accuracy. However, there are more problems with oscillations in this case.

Table 4.3: Result for performance test 1

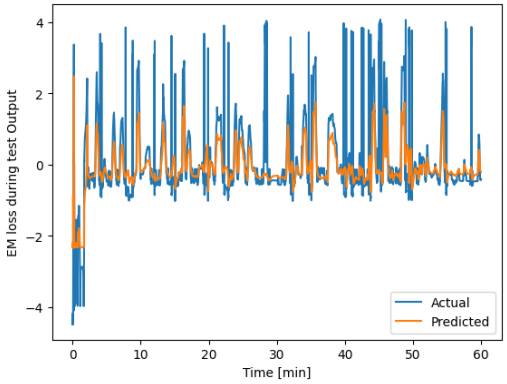
Performance test 1	Without Physics	With Physics	With Physics and Gradient Constraint
Epochs	58	200	124
MAE Power Loss	0.378	0.097	0.087
MAE Winding Temperature	0.523	0.323	0.333
RMSE Power Loss	0.646	0.194	0.179
RMSE Winding Temperature	0.706	0.496	0.497
Time to train	15 min 20 s	51 min	52 min

Table 4.4: Result for performance test 2 (switched test cycle)

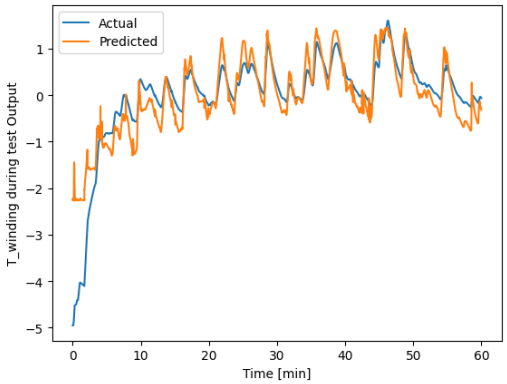
Performance test 2	Without Physics	With Physics	With Physics and Gradient Constraint
Epochs	35	42	58
MAE Power Loss	0.317	0.066	0.063
MAE Winding Temperature	0.382	0.186	0.178
RMSE Power Loss	0.409	0.099	0.097
RMSE Winding Temperature	0.507	0.246	0.242
Time to train	11 min	13 min	23 min



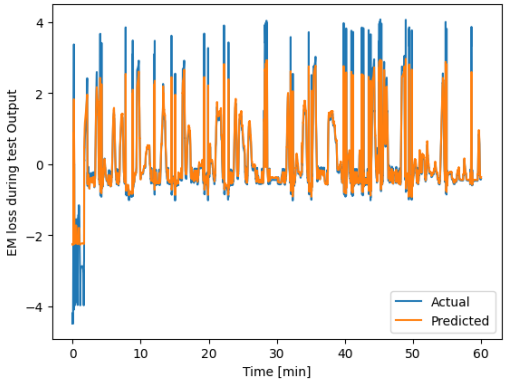
(a) No physics winding temperature



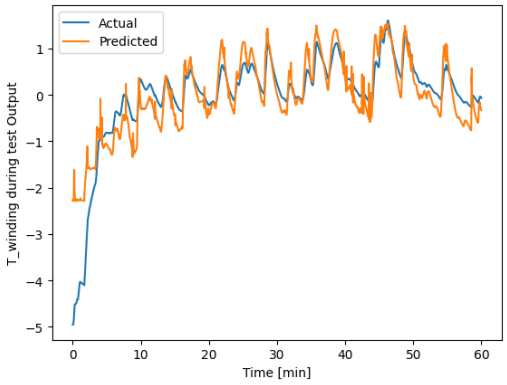
(b) No physics power loss



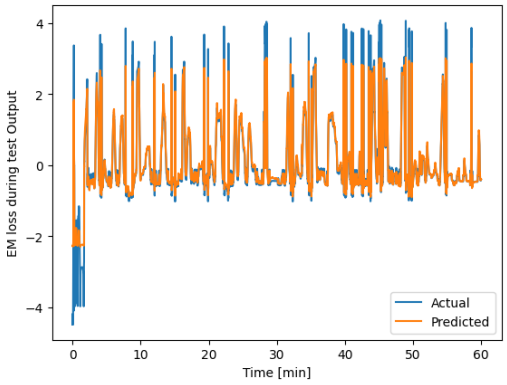
(c) With physical input winding temperature



(d) With physical input power loss



(e) With physical input and gradient constraint winding temperature

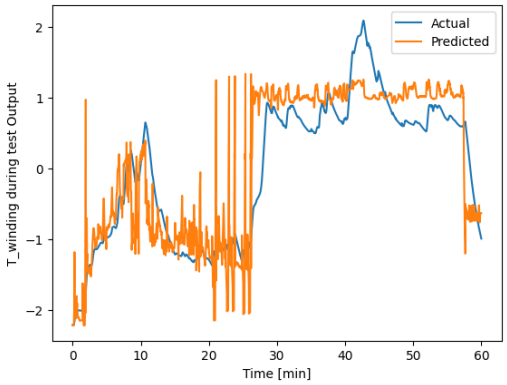


(f) With physical input and gradient constraint power loss

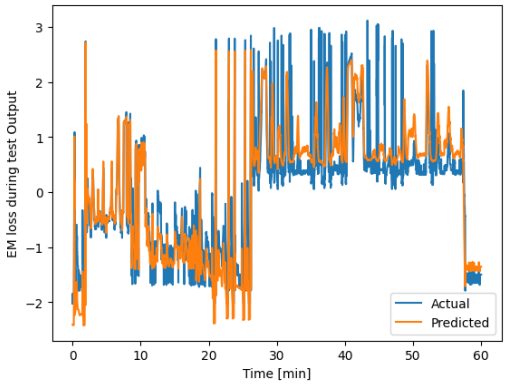
Figure 4.3: Performance test 1

As can be seen in the figures above, the model has issues with the predictions of the initial part of the test cycle. This is because the training data does not cover the test data, as seen in figure 4.5. Figure 4.5 - 4.7 shows the range of the training and testing data, and the area of operation for the winding temperature and power loss. The small range of the training data set also causes a significant difference in the

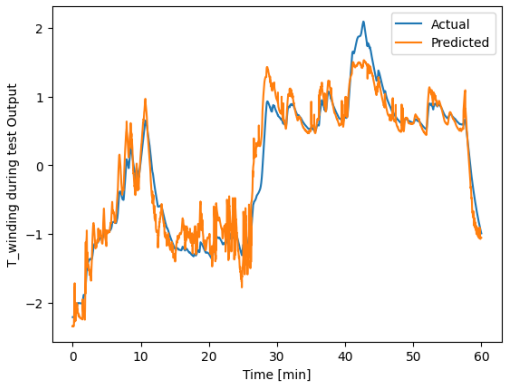
amplitude of the power estimation.



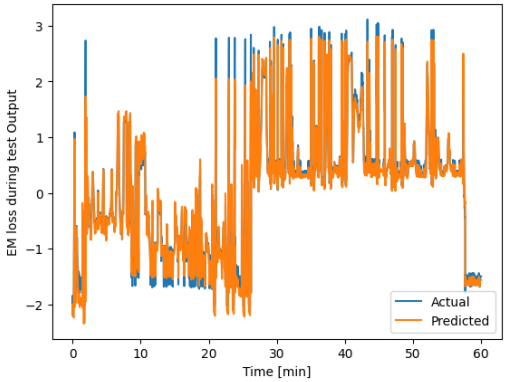
(a) No physics winding temperature



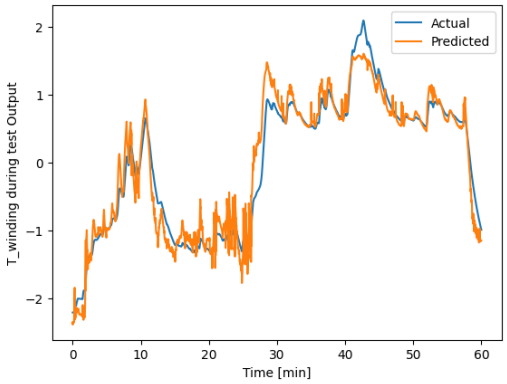
(b) No physics power loss



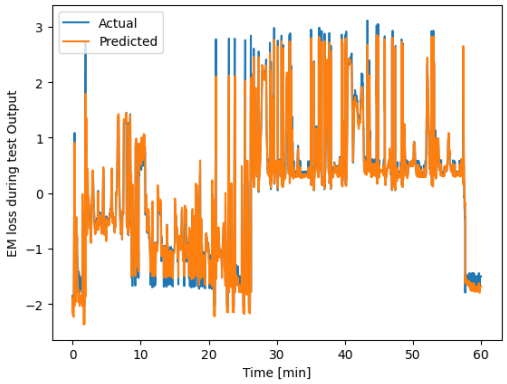
(c) With physical input winding temperature



(d) With physical input power loss



(e) With physical input and gradient constraint winding temperature



(f) With physical input and gradient constraint power loss

Figure 4.4: Performance test 2

From the second performance test, it can be noted that the temperature gradient constraint has a negligible effect on the oscillations of the system. Since there is a more extensive range of data in this case, the network predicts the amplitude of the

power loss more accurately. At 43 minutes, there is a significant difference in the temperature, which is caused by the limited range of the training data set, which can be seen in figure 4.6. The same reasoning applies to the power loss estimations. In this case, most of the test data is within the range of the training data, which is why the network is faster at learning the system's dynamics and needs fewer epochs to do so.

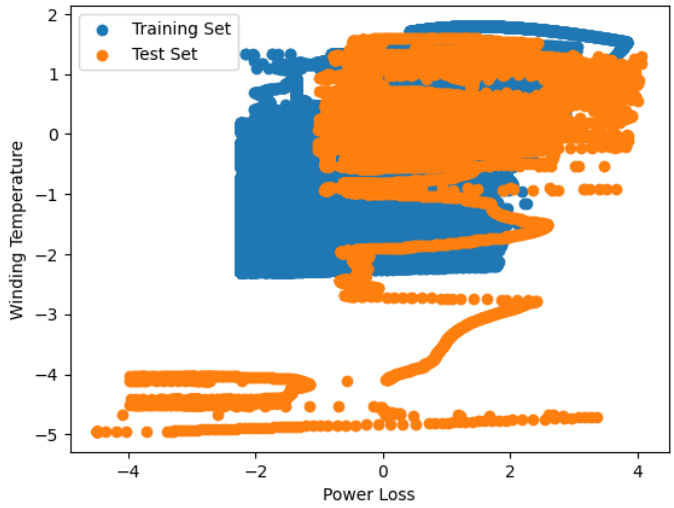


Figure 4.5: Scatter plot of training and testing data for performance test 1

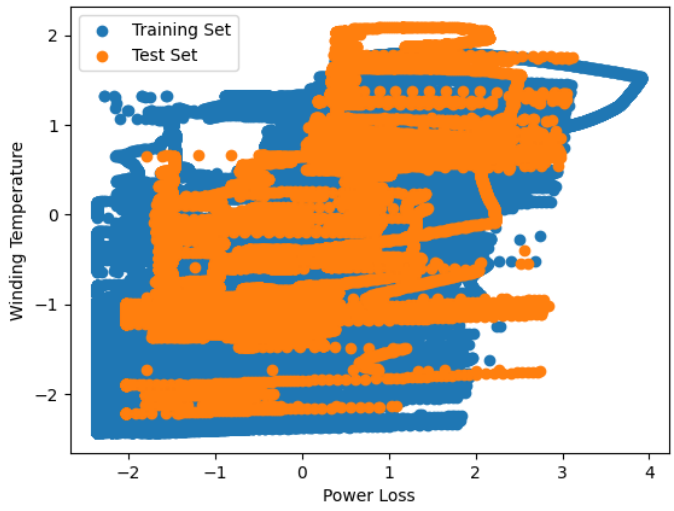


Figure 4.6: Scatter plot of training and testing data for performance test 2

4.1.2.2 Lower amount of available data

Since there is less data in this case, the normalization will be different due to the difference in standard deviations. The training and test data can be seen in figure 4.7, and it can be concluded that a more significant part of the test set is within the range of the training set. This analysis indicates that the network should be able to predict the correct output.

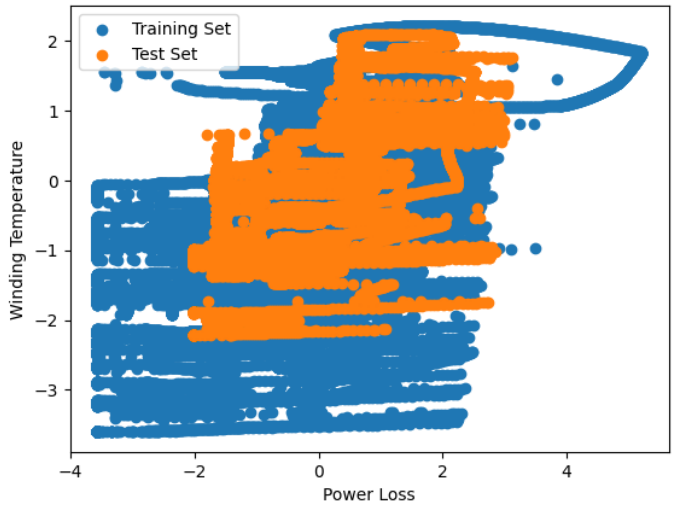
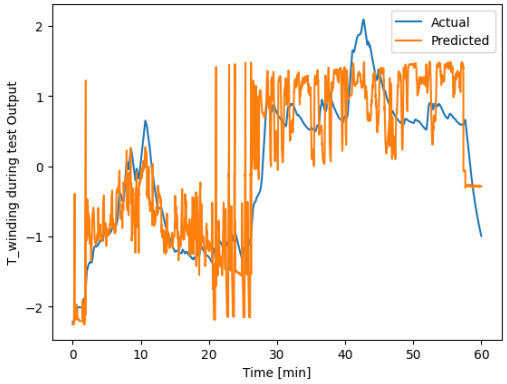


Figure 4.7: Scatter plot of Training and testing data

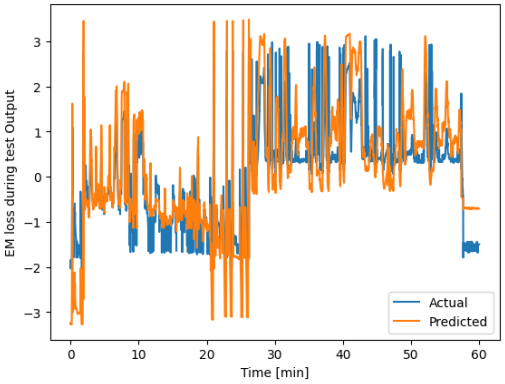
The tables below show that the PINN performs much better than the regular data-driven network. Similarly to the first performance test, there is a slight difference between the network with physics and the one with physics and a gradient constraint. From figure 4.8, it is more evident that the gradient constraint is dampening the oscillations of the signal. This result indicates that the PINN outperforms the regular neural network with less data.

Table 4.5: Result for performance test with less training data

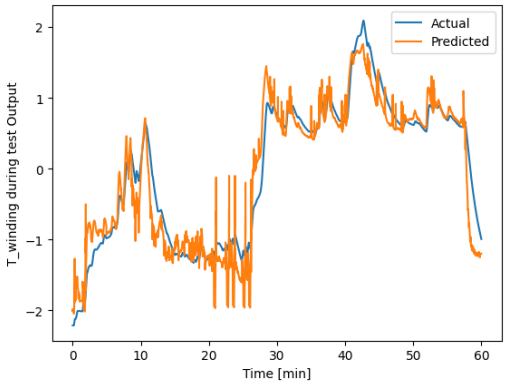
Less data performance test	Without Physics	With Physics	With Physics and Gradient Constraint
Epochs	20	18	85
MAE Power Loss	0.546	0.097	0.095
MAE Winding Temperature	0.452	0.222	0.278
RMSE Power Loss	0.693	0.187	0.130
RMSE Winding Temperature	0.540	0.302	0.356
Time to train	2 min	2 min	15 min 20 s



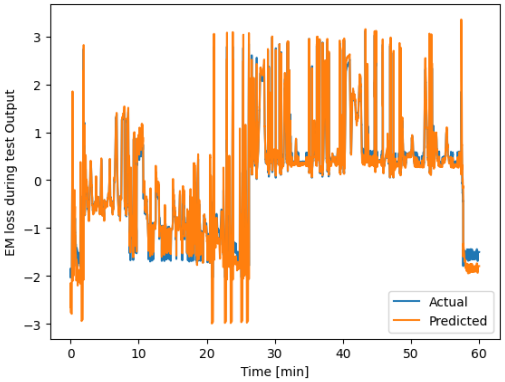
(a) No physics winding temperature



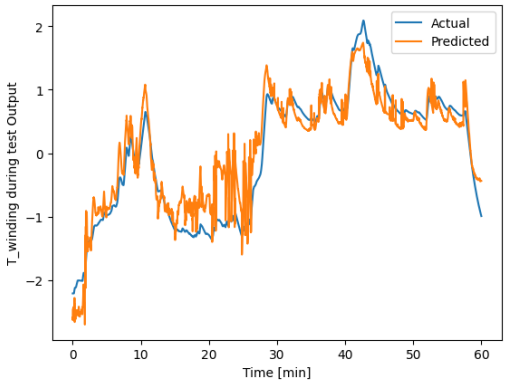
(b) No physics power loss



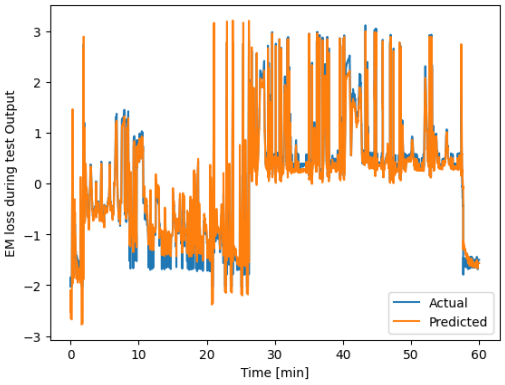
(c) With physical input winding temperature



(d) With physical input power loss



(e) With physical input and gradient constraint winding temperature



(f) With physical input and gradient constraint power loss

Figure 4.8: Less data performance test

4.2 Pre-Calculations of the system

The results below are the pre-calculated values based on the physical equations for the winding temperature and the power loss for chapter 3.1.1. The Forward Euler

estimation and power loss calculation are not a perfect representation of the system. However, it provides the neural network with crucial information and increases its performance.

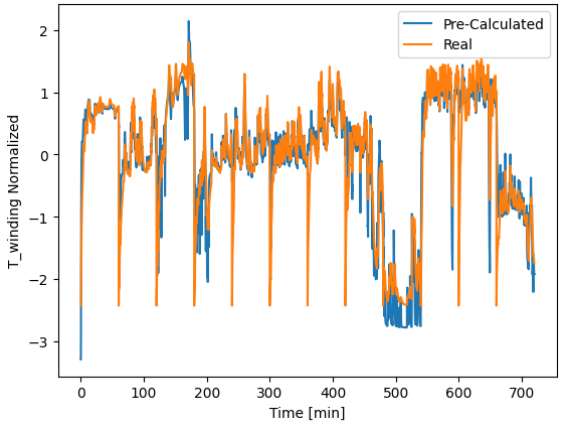


Figure 4.9: Simulated- and Calculated results for $T_{Winding}$

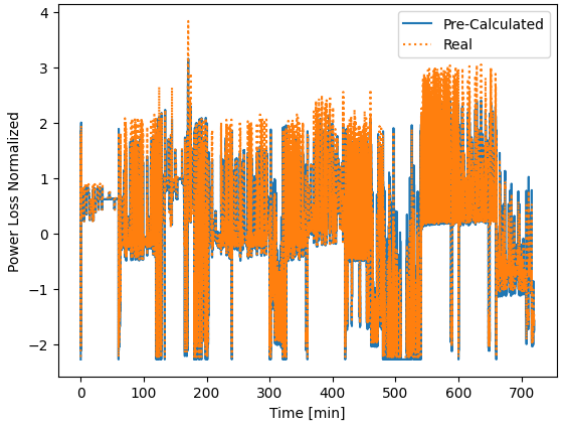


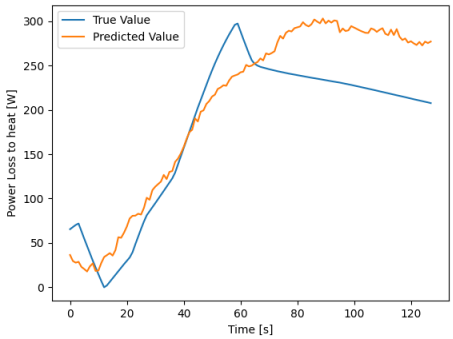
Figure 4.10: Simulated- and Calculated results for Power loss

4.3 Nerve

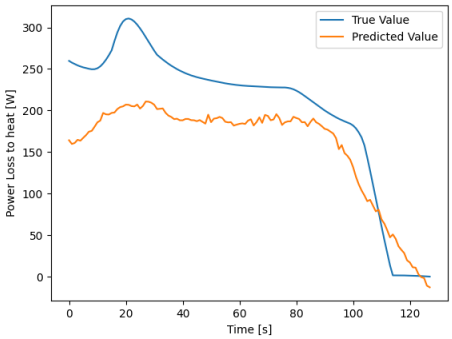
4.3.1 First training

The results are produced with 4.6 million data points for each signal in the dataset. As shown in figure 4.11 on a broad scale, the model has the ability, in simpler cases, to predict the correct output with an error margin. However, for more complex cases, figure 4.11a and 4.11d, the model has difficulties learning the system's behavior. The limited amount of data is likely the cause of this. The power loss is related directly to the amount of current in the thermal model, the voltage over it, and the coolant temperature input, which makes the thermal power losses more straightforward to predict than the winding temperature. This can be seen in figure 4.14, where the model has difficulties predicting the Winding temperature in all cases. In case 4.12a,

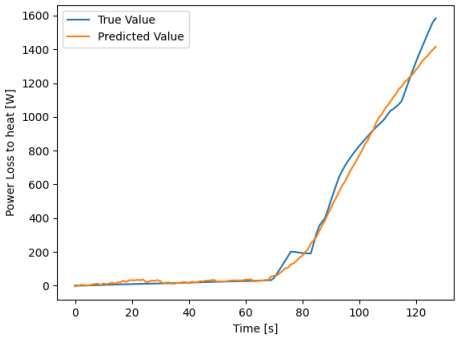
the model seems to be predicting the output relatively accurately since the difference in prediction is around 3 degrees C. However, to have a better prediction, more data is needed.



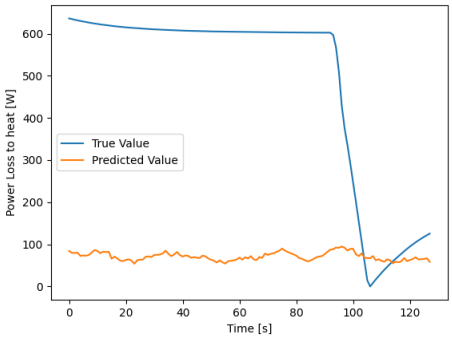
(a) Power Loss for random slice 1



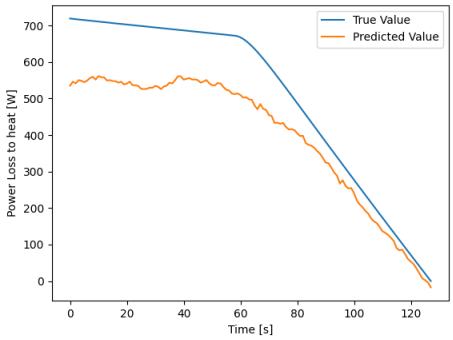
(b) Power Loss for random slice 2



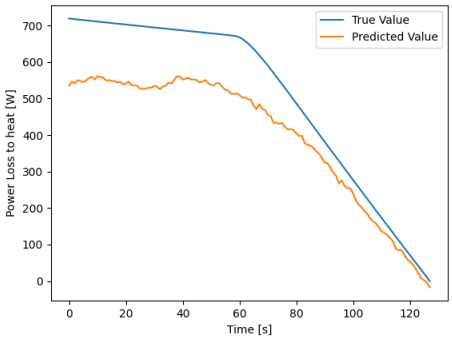
(c) Power Loss for random slice 3



(d) Power Loss for random slice 4

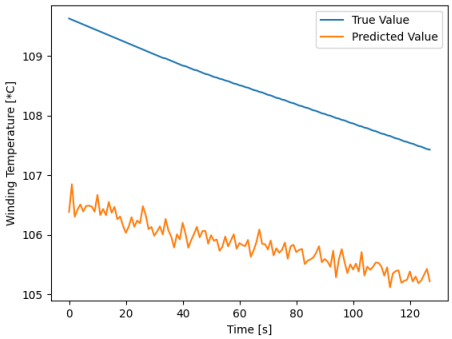


(e) Power Loss for random slice 5

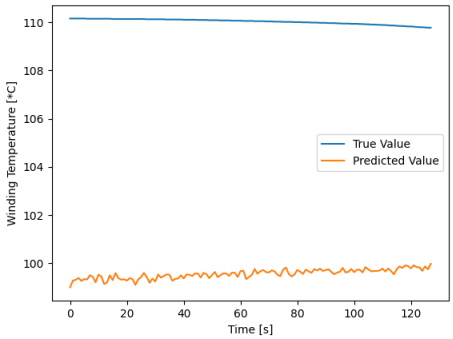


(f) Power Loss for random slice 6

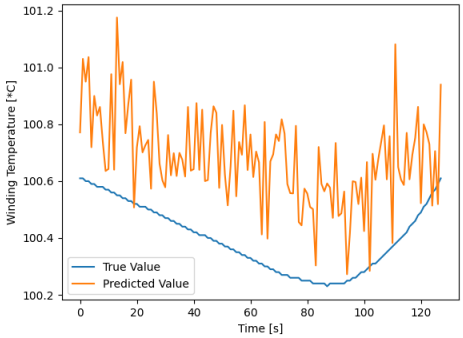
Figure 4.11: First Training with Nerve Framework - Power Loss



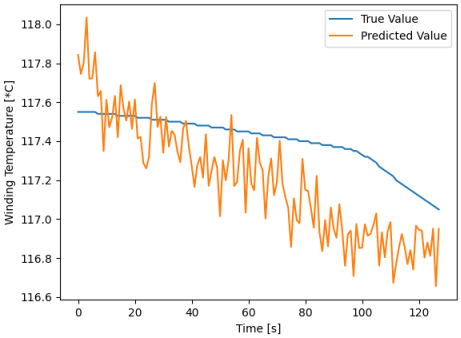
(a) Winding temperature random slice 1



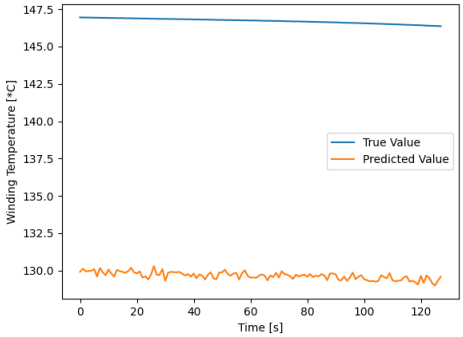
(b) Winding temperature random slice 2



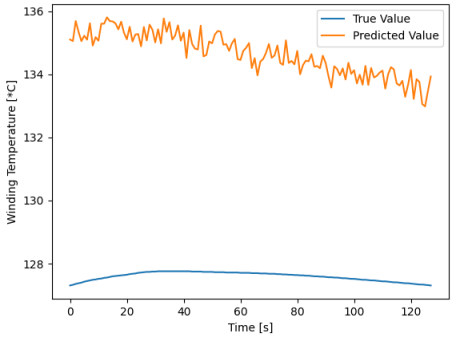
(c) Winding temperature random slice 3



(d) Winding temperature random slice 4



(e) Winding temperature random slice 5

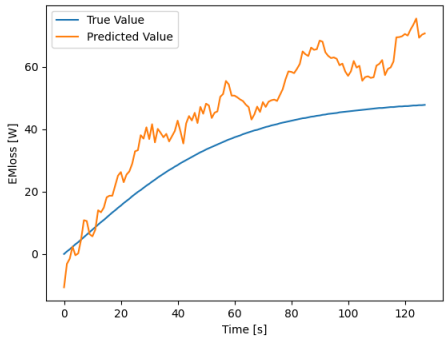


(f) Winding temperature random slice 6

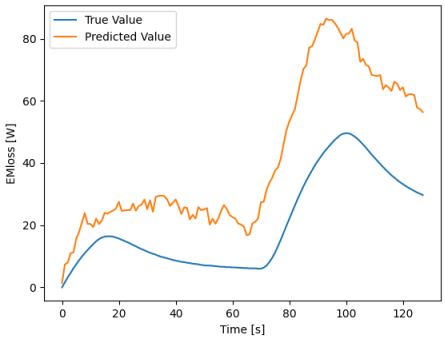
Figure 4.12: First Training with Nerve Framework - Winding Temperature

4.3.2 Second training

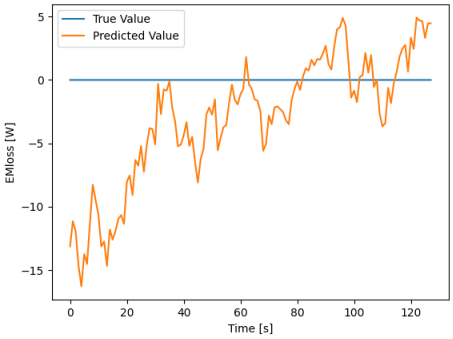
This time the amount of data was increased to 5.6 million data points. The results presented are similar to the first test. However, with a slight increase in performance, since the model has more data to train on. Furthermore, in the best cases presented, the prediction is almost identical to the actual value, which indicates that the artificial neural network model has the potential to be used as a data-driven model. However, more testing is required to ensure this.



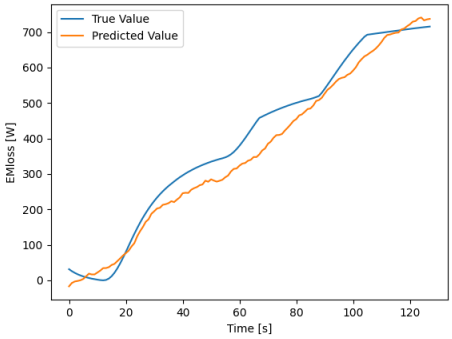
(a) Power Loss for random slice 1



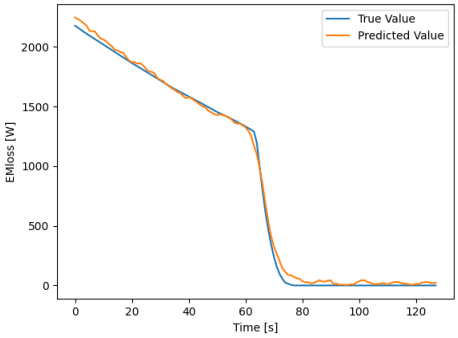
(b) Power Loss for random slice 2



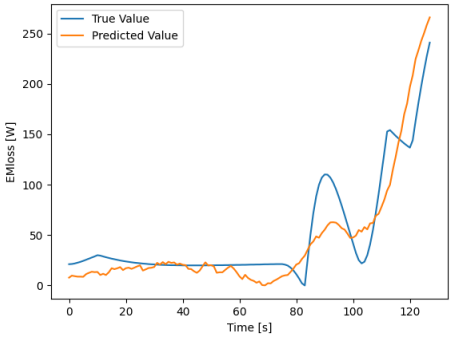
(c) Power Loss for random slice 3



(d) Power Loss for random slice 4

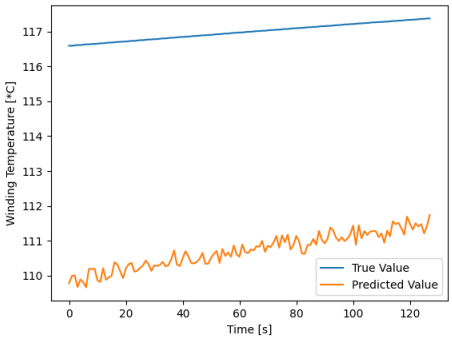


(e) Power Loss for random slice 5

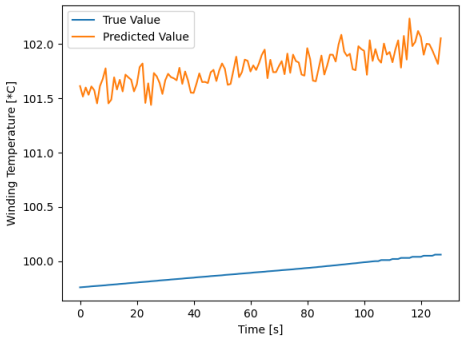


(f) Power Loss for random slice 6

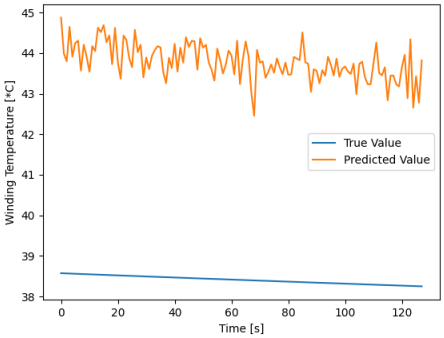
Figure 4.13: Second Training with Nerve Framework - Power Loss



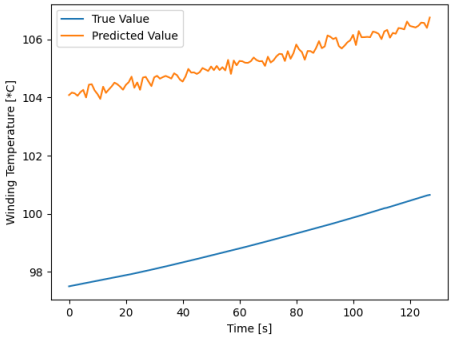
(a) Winding temperature random slice 1



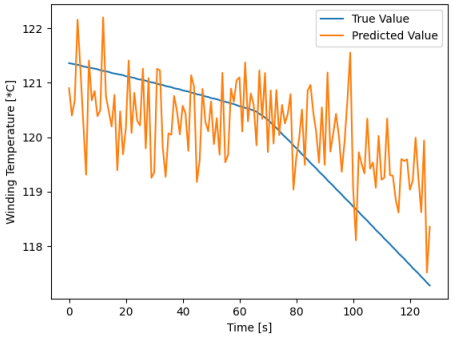
(b) Winding temperature random slice 2



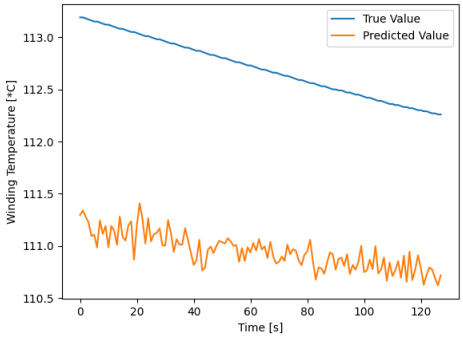
(c) Winding temperature random slice 3



(d) Winding temperature random slice 4



(e) Winding temperature random slice 5



(f) Winding temperature random slice 6

Figure 4.14: Second Training with Nerve Framework - Winding Temperature

5
Discussion

The concept of PINN is where Maziar Raissi [6] is one of the pioneers within the area and one of the most cited. With this being one of the first articles discussing this subject, there are some limitations in the published research. What can be addressed is the application versus proof of concept models available for PINNs, with very few applications. However, in the future, this method could potentially eliminate current methods such as FEM analysis.

In Volvo's case, incorporating PINNs is a satisfactory solution since field test data could be used. This is because that data is quite scarce and not very precise. However, this is the advantage of PINN since the data can be limited using governing equations, gradient constraints, and initial conditions, and the network will still suffice. However, the time aspect has to be kept in mind; due to the limited literature and knowledge in this area, waiting for the literature to increase could be more beneficial.

In the case of Volvo, AI technology is an excellent way to digitally build virtual components to conduct simulations for different use cases. However, there are always risks. To ensure that a model is capable of generating accurate predictions, it is essential that the data utilized for training be of adequate quality and possess sufficient generalizability.

One advantage of using AI models for driveline components is that they can be trained on new data as the new data comes in from test rigs or test vehicles. This will generally increase the model's accuracy since it will be trained on new scenarios and driver behaviors. However, continuously training the model is out of the scope of this thesis, but with further integration of AI for Volvo, this possibility arises.

5.1 PINN

5.1.1 Performance tests and results

From the results in the graphs in Chapter 4, it is clear to conclude that with physics integrated into the network, the accuracy increases. The difference in training time for using gradient constraints and not is negligible. The slight difference in performance with and without gradient constraints can be because the tuning of the physical weight parameter or the network effectively learns the dynamics of the ther-

mal model, and the condition is unnecessary.

For the case with fewer data, the results show that the physics significantly affects the performance, and the gradient constraint has a noticeable impact on the oscillations of the system. For the case of using field test data which is scarce, the results indicate that when incorporating physics, the network will probably be able to predict the desired output more effectively than without the physics.

It is impossible to say if the hyperparameters that are selected are optimal for this network since a lot of tuning is required and that takes quite an amount of time. However, that is outside the scope of this investigation. It seems from the results that it is clear that the PINN is outperforming the regular RNN. One cause for this is that the network has more information to train on and that the weighting process will find more correlations faster. However, the results may have differed if the network had been trained on more data. In this case, when not utilizing the physical knowledge, the regular RNN started to overfit the training data, which is an undesirable result since the network won't be able to accurately predict the output of the test data.

It is stated that the LSTM and GRU can remember past inputs more efficiently than the regular RNN. Although on the cost of increased memory and training. It is important to analyze the system that the neural network is trying to replicate, as it may not necessarily need to remember inputs from several time steps ago, but only those closest to it. For the thermal model, this is the case where the temperature does not necessarily depend on what the temperature was one or two hours ago, but rather what it was five to ten minutes ago. So it is not needed to have a long time memory. This can be observed where the GRU outperforms the LSTM. Where the GRU is not as good at handling long-time dependencies, but due to its simpler architecture has a faster computational time.

5.1.2 Development

The development time was quite short when the aspect of the lacking experience in AI development is considered. The advantage of creating a completely new AI model is that it is possible to customize it to the problem that is supposed to be solved during development. In comparison to Nerve, it is possible to change which type of neural network is utilized, which gives the method of creating a new framework more abilities to customize the networks for specific tasks, for example, other thermal models or other components in the drive line. However, this comes with the cost of time and acquiring the necessary knowledge to create the framework.

When it comes to modeling neural networks, there is a lot of trial and error, and this research was no exception. There are a lot of network structures and types that can be tested. It was found from many tests that, in this case, a wider network performed better than a deeper one. The solver that was used during testing was changed multiple times; solvers like Adam, SGD, and AdamW were tested, and it

was found that AdamW was the best-performing solver for this kind of task. However, there are many more solvers, and another could easily be better at solving these types of tasks, but a limit has to be drawn somewhere.

5.1.3 Other approaches

From the results of performance test 1, one conclusion that can be drawn is that the Neural Network has issues with the initial value of the test cycle. This could be because the first value is multiple deviations from the mean, while the majority of other values are closer to the mean value, or the test cycle not being represented by the training data set. One method to solve this issue could instead be to predict the rate of change in temperature. In this way, the network should be able to predict how the network temperature changes from one sample to another and could, based on inputs, predict an array of temperature changes. With that array, the winding temperature can be calculated by adding the initial temperature to the first index and then summarizing the changes. This was tested in the project, but without success, due to the sensitivity of the accuracy regarding the power loss predictions. One possible solution to this issue more tuning of the hyperparameters and acquiring more data to train on.

5.1.4 Optimal Solution

It is almost impossible to conclude if a network is an optimal solution. However, in this case, where the goal is to prove the concept of physically informed neural networks for drive line components, a network that can provide good enough results are considered successful. Depending on the different MDS, a new network with other hyperparameters may have to be created due to the differences in the motors and the available operating ranges. Another conclusion that can be drawn is that the training time increases when implementing physics into the network. This can lead to unnecessary computation time if more data is acquired and the implementation of physics is unnecessary. As can be seen in the results, the time to train is increased by a factor of 3.5 in the first- and 2 in the second performance test. In the case with fewer data points to train on, it takes the same time to train with the physics and 7.5 times as long time with physics and gradient constraints. The gradient constraint is only implemented on the winding temperature gradient, where the effects of those physics may not be the most effective. However, as seen in the figures, the oscillations are dampened, which is a desired behavior.

Weighted values were tested in order to find an optimal solution. However, weighting the values beforehand is a new hyperparameter requiring tuning. Some testing found that when it increased performance for winding temperature, the accuracy of the power loss decreased. More tuning could be required for this to be an acceptable method of acquiring the optimal solution.

5.2 Data Sensitivity

As can be seen in the results the NN is really sensitive to a limited amount of data if physics is not incorporated into the network. An ideal case would be to train a network on the entire operating range and get it to train on more cycles for a longer duration, this will make new test sets have a higher probability of being within the operating range of the network. From the results in section 4.1.2.1, it can be seen that if the test data set is out of range from the training set, the predicted results may be limited, as can be seen in figure 4.4e and 4.4f. This is something that has to be investigated further if a more accurate model should be created, ensure that the data that is being used to train the network contains all the realistic operating points.

Data sensitivity is a problem affecting NNs in general. With noisy and scarce data, the network may catch unwanted behaviors and see that as a dynamic of the modeled system. However, a PINN can handle those issues if the amount of data is low. The PINN should enable better models when using other types of sources of data, like rig tests or data collected from field test vehicles.

5.3 Reduced order Modeling

AI can be used to start reducing the order of the current model within the drive line. The model that is created by the script is compatible with Simulink, which also gives it an upper hand over the Nerve framework. More components could be modeled in the same network as long as the data is available. However, in that case, it is much easier to build a completely data-driven model instead of implementing physics, both from a development- and training aspect. In the future, if the Nerve framework models become compatible with Simulink, it would be a feasible approach to initiate ROM. This can effectively reduce the calculation time for each simulation, and allow for the representation of real components using data from diverse tests, including rig tests. However, creating a solely data-driven model would result in longer training time when attempting to represent multiple components within a single network.

Issues could emerge from only using data-driven models, extensive models where many components are integrated, and the main one is that it is nearly impossible to interpret the model since it will be a black box model. This means that the physical knowledge will be limited, and the work done to the current virtual drive line will go to waste. If PINNs were used instead, all information is not would not be lost. However, some adjustments will be necessary in the modeling process, and a considerable amount of tuning work needs to be done. In this investigation, much time was spent tuning the network to get satisfactory results; if more components were modeled in the same network, that tuning needs to be more precise to ensure accurate results.

5.4 Nerve

The testing of Nerve was mainly done to compare the development time of creating a new framework versus using an already defined one. The development time was much faster, and generating results and evaluating network performance was easy. However, as can be seen in the results for the testing of Nerve, there are some issues with parts of the cycles and some outstanding results for other parts. Since the Nerve networks were trained on 4.6 - 5.6 million data points and had worse performance than the results from the created PINN, the network needs some physical input or more data to generate accurate outputs. The PINN was trained on fewer data points and performed better, showing that a purely data-driven model is not always the best way to reduce the order of the components within the drive line.

One problem with the Nerve framework is that currently, it is not possible to test the model created for an entire drive cycle, the test is instead generated from random batches within the data set. Where 80 % is training data and 10 % is validation and test data. This may create some problems with the reset between each cycle and the network may try to catch this behavior and that could generate the oscillations that can be seen in the predictions made.

5.5 Future Work

The current data consists of one-hour-long drive cycles, meaning each cycle has the same length. Future work should include testing longer cycles (more data) and training on cycles with different amounts of data points. Currently, the program is compatible with data where the batch size can be a factor of the drive cycle length, and if the drive cycles have different lengths, the batch size parameter can be difficult to tune.

Other types of physical constraints can be added to this problem, which could increase the model's accuracy. Knowing which constraints are helpful is challenging, so testing is required to see what constraints help and which makes the network perform worse than without constraints. In this paper, a gradient constraint was put on the winding temperature. It was determined that since the power loss is so closely connected with the current, voltage, torque, and speed, there was no need for further physical implementation. However, some constraints can be put on the power loss that will increase the prediction accuracy.

As previously stated, finding the optimal parameters for this application requires extensive tuning and testing of the network. While some tuning and testing have been conducted in this thesis, a more comprehensive process on all hyperparameters is necessary to achieve the optimal network. And not just the traditional hyperparameters like the number of neurons or learning rate, but also hyperparameters in the physics implementations. For example how big the calculated input should influence the result related to the actual input.

Testing with vehicle data and rig data could further prove the advantages of using PINNs for dynamic problem-solving. When doing this, filtering the data is required to a greater extent than using GSP data, which barely has any noise or errors. One first step here is to try with data from the MDS rig and when GSP drive cycles are fed to that rig since that data will be the most "clean" and as close as you get to reality when training the network. The program should already be compatible with this type of data. It is a matter of acquiring, filtering, and cutting it to size.

In the case of Volvo the Nerve framework is a clear way forward when building and evaluating data-driven models. While the development of Nerve requires in-depth knowledge of machine learning and transformers, users do not require such knowledge. As a result, building and evaluating models is highly efficient, paving the way for more data-driven components in the virtual drive line.

This thesis focuses on analyzing the thermal model of an electric motor, which is considered one of the most intricate parts to model. Further investigation is required to determine if Physical Informed Neural Network models are compatible with other components of the motor. A hybrid solution could be explored by combining previous physical models with PINN models to create a more comprehensive model of the entire motor.

5.6 Ethics

This investigation's most crucial part of ethics is connected to the data collection. If field test vehicle data is collected, the integrity of the driver of that vehicle must be protected so that the driver can not be connected with the data that has been generated. Obtaining the driver's consent is crucial as they must be informed about the data being collected and its intended use.

5.7 Sustainability

One aim of this project was to decrease the computational time of simulating electric drive lines, thus decreasing the power usage of the computer performing the simulations. If more of the now physics-based models can be turned into reliable data-driven models, the energy consummation of the simulations could be decreased. However, looking at the whole process of creating these data-driven models is essential. It requires much computational power for training and storing data. So from an energy-saving point of view, the amount of times new models are created impacts the result.

Another aim of this project was to make the model more accessible and more usable with fewer resources. This could open the possibility of having a digital twin in the vehicle, which could continuously simulate the vehicle's life span. The driver can make informed decisions that positively impact the vehicle and its surroundings

by providing relevant information. This includes determining the optimal battery charge time and identifying potential issues requiring attention.

Further, this project aims to create more accurate models. More accurate models will, in turn, create more efficient electric drive systems that will utilize its resources better.

6
Conclusion

For this thesis, a Physically Informed Neural Network was developed to predict the output of the current thermal model in the virtual drive line. The network underwent testing both with and without physics and gradient constraints. The results indicate that the physics-informed neural network outperforms the regular GRU network. Previous studies have shown that PINNs have the potential to outperform normal NNs, particularly when there is limited available data. This has also been demonstrated in this case. The development time for a PINN is heavily limited to the amount of physical knowledge about the evaluated component and the available data. In order to create the perfect PINN, much time has to be put into tuning the hyperparameters. The created PINN is performing well, estimating both the winding temperature and the power loss with high accuracy and outputting predictions with a realistic shape. From the best-performing PINN, the results show an increase in training time by a factor of two but a reduction in MAE losses of 80 % and 53 % for power loss and winding temperature, respectively, compared with a regular GRU.

Based on the results of this investigation, it has been demonstrated that a PINN can serve as a thermal model. However, more tuning is needed in order to get more accurate estimations. The tests have also revealed that PINNs outperform purely data-driven models, indicating that they could be an effective approach for Volvo, facilitating fast simulation times with acceptable accuracy. Nonetheless, additional testing with more data and diverse gradient constraints must confirm that PINNs are the way forward for virtual simulations.

After evaluating the Nerve framework, it has been found to possess great potential. However, it may face some difficulties in predicting temperature with limited data. To work effectively with this framework, gathering a significant amount of data before initiating the modeling process is crucial.

References

[1] *Transportation emissions worldwide - statistics & facts | Statista*. [Online]. Available: https://www.statista.com/topics/7476/transportation-emissions-worldwide/#topicOverview.

[2] L. Balkanyi and R. Cornet, "The Interplay of Knowledge Representation with Various Fields of Artificial Intelligence in Medicine," *Yearbook of Medical Informatics*, vol. 28, no. 01, pp. 027–034, Aug. 2019, ISSN: 0943-4747. DOI: 10.1055/s-0039-1677899.

[3] CB Insights and PwC, *Artificial intelligence (AI) funding investment in the United States from 2011 to 2019 (in million U.S. dollars) [Graph]*, 2020. [Online]. Available: https://www.statista.com/statistics/672712/ai-funding-united-states/?locale=en.

[4] A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, Sep. 2018, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2870052.

[5] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019, ISSN: 0021-9991. DOI: 10.1016/J.JCP.2018.10.045.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," Nov. 2017. [Online]. Available: https://arxiv.org/abs/1711.10561v1.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[8] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," *Journal of Heat Transfer*, vol. 143, no. 6, Jun. 2021, ISSN: 15288943. DOI: 10.1115/1.4050542/1104439. [Online]. Available: https://asmedigitalcollection.asme.org/heattransfer/article/143/6/060801/1104439/Physics-Informed-Neural-Networks-for-Heat-Transfer.

[9] G. Rebala, A. Ravi, and S. Churiwala, *An Introduction to Machine Learning*, 1st ed. Springer eBooks, 2019, ISBN: 9783030157296.

[10] C. C. Aggarwal, "Neural Networks and Deep Learning," *Neural Networks and Deep Learning*, 2018. DOI: 10.1007/978-3-319-94463-0.

[11] T. Wang, Z. Chen, Q. Shang, C. Ma, X. Chen, and E. Xiao, "A Promising and Challenging Approach: Radiologists' Perspective on Deep Learning and Artificial Intelligence for Fighting COVID-19," *Diagnostics 2021, Vol. 11, Page 1924*, vol. 11, no. 10, p. 1924, Oct. 2021, ISSN: 2075-4418. DOI: 10.3390/DIAGNOSTICS11101924. [Online]. Available: https://www.mdpi.com/2075-4418/11/10/1924/htm%20https://www.mdpi.com/2075-4418/11/10/1924.

[12] *Recurrent Neural Network and it's variants. . . . | by Shujaat Hasan | Analytics Vidhya | Medium*. [Online]. Available: https://medium.com/analytics-vidhya/recurrent-neural-network-and-its-variants-de75f9ee063.

[13] D. Hendrycks, K. Lee, and M. Mazeika, "Using Pre-Training Can Improve Model Robustness and Uncertainty,"

[14] X. Han, Z. Zhang, N. Ding, *et al.*, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, Jan. 2021, ISSN: 26666510. DOI: 10.1016/J.AIOPEN.2021.08.002.

[15] F. Biesinger and M. Weyrich, "The Facets of Digital Twins in Production and the Automotive Industry," in *2019 23rd International Conference on Mechatronics Technology (ICMT)*, IEEE, Oct. 2019, pp. 1–6, ISBN: 978-1-7281-3998-2. DOI: 10.1109/ICMECT.2019.8932101.

[16] M. Shafto, M. C. Rich, D. E. Glaessgen, C. Kemp, J. Lemoigne, and L. Wang, "DRAFT MoDeling, SiMulATion, inFoRMATion Technology & PRoceSSing RoADMAP Technology Area 11," 2010.

[17] H. Guo, Q. Ding, Y. Song, H. Tang, L. Wang, and J. Zhao, "Predicting temperature of permanent magnet synchronous motor based on deep neural network," *Energies*, vol. 13, no. 18, Sep. 2020, ISSN: 19961073. DOI: 10.3390/EN13184782.

[18] W. Li, J. Zhang, F. Ringbeck, *et al.*, "Physics-informed neural networks for electrode-level state estimation in lithium-ion batteries," *Journal of Power Sources*, vol. 506, Sep. 2021, ISSN: 03787753. DOI: 10.1016/j.jpowsour.2021.230034.

[19] H. S. Khan, I. S. Mohamed, K. Kauhaniemi, and L. Liu, "Artificial Neural Network-Based Voltage Control of DC/DC Converter for DC Microgrid Applications,"

[20] S. Cuomo, V. Schiano, D. Cola, *et al.*, "Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next," 2022.

[21] H. Wu, S. Niu, Y. Zhang, and W. Fu, "Physics-Informed Generative Adversarial Network-Based Modeling and Simulation of Linear Electric Machines," *Applied Sciences 2022, Vol. 12, Page 10426*, vol. 12, no. 20, p. 10 426, Oct. 2022, ISSN: 2076-3417. DOI: 10.3390/APP122010426. [Online]. Available: https://www.mdpi.com/2076-3417/12/20/10426.

[22] G. D. Demetriades, H. Z. De La Parra, E. Andersson, and H. Olsson, "A real-time thermal model of a permanent-magnet synchronous motor," *IEEE Transactions on Power Electronics*, vol. 25, no. 2, pp. 463–474, 2010, ISSN: 08858993. DOI: 10.1109/TPEL.2009.2027905.

[23] *1.2: Forward Euler method - Mathematics LibreTexts*. [Online]. Available: https://math.libretexts.org/Bookshelves/Differential_Equations/Numerically_

Solving_Ordinary_Differential_Equations_(Brorson)/01%3A_Chapters/1.02%3A_Forward_Euler_method.

[24] *Welcome to Python.org*. [Online]. Available: https://www.python.org/.

[25] *PyTorch*. [Online]. Available: https://pytorch.org/.

[26] *Feature Engineering: Scaling, Normalization and Standardization*. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/.

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

CHALMERS
UNIVERSITY OF TECHNOLOGY