FACULTY OF ENGINEERING — LUND UNIVERSITY

CENTRE FOR MATHEMATICAL SCIENCES

DEGREE PROJECT IN MATHEMATICAL STATISTICS FOR ENGINEERS

Creating a Virtual Tyre Temperature Sensor

Axel Tevell
ax5754te-s@student.lu.se

Oskar Zetterberg
os4858ze-s@student.lu.se

June 15, 2023

Master's thesis work carried out at Volvo Cars Sverige Aktiebolag.

Supervisors: Johan Lindquist Holmberg, johan.holmberg4@volvocars.com
Anders Lindman, anders.lindman@volvocars.com
Andreas Jakobsson, andreas.jakobsson@matstat.lu.se

Examiner: Maria Sandsten, maria.sandsten@matstat.lu.se

Abstract

To accurately determine the efficiency and range of an electric vehicle, one must be able to estimate the rolling resistance of the car. This is currently done using standardized methods developed under laboratory settings, where transient aspects and effects of varying temperatures are excluded. Given the strong correlation between tyre temperatures and the rolling resistance, determining this temperature is of great interest. This Master's thesis investigates the development of a virtual sensor for predicting the tyre temperature during dynamic driving using recurrent neural networks (RNN). The primary goal is to examine if a virtual sensor can predict the tyre temperature within ±2 °C of the actual temperature using on-board vehicle signals. The study also evaluates the significance of these signals in the model. Two different RNN architectures, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were trained and evaluated. The LSTM performed slightly better and results indicated that the model can predict the tyre temperature within the ±2 °C interval around 90% of the time. The most crucial features contributing to model performance were identified as vehicle speed, ambient temperature, brake pedal position, accelerator pedal position, and road inclination. To improve on this results, a few interesting future research areas were suggested.

Keywords: Tyre temperature, Virtual sensor, Soft sensor, Time series, Recurrent neural networks, LSTM, GRU.

Acknowledgements

First and foremost we would like to express our gratitude to Volvo Cars for giving us the opportunity to write our thesis work with them and providing us with adequate equipment. Also, a special thanks to our supervisors at Volvo Cars, Johan Lindquist Holmberg and Anders Lindman. Your expertise, thoughtful feedback, and encouragement throughout this project have been instrumental for the success of this work. It has been a true pleasure working this closely with you during our 20-week stay.

We also give our warmest thanks to our supervisor at LTH, Andreas Jakobsson, for always being available, invaluable guidance, and interesting discussions. You were the first one we thought of when searching for a supervisor, and we are deeply grateful you accepted our request.

Finally, we would like to thank Chandra Sekhar Rachannagari, Mattias Samuelsson, and everyone else at the Vehicle Energy Efficiency team at Volvo Cars for interesting discussions and appreciated insights during our stay.

Contents

1. Introduction **9**
1.1 Background . . . 9
1.2 Research goals . . . 10
1.3 Limitations . . . 10
1.4 Structure . . . 11

2. Background on Neural Networks **12**
2.1 Artificial Neural Networks (ANN) . . . 12
2.2 Multilayer perceptron (MLP) . . . 13
2.2.1 Backpropagation . . . 14
2.2.2 Learning rate . . . 15
2.2.3 Batch and batch size . . . 15
2.3 Recurrent Neural Networks (RNN) . . . 16
2.4 Long-short term memory (LSTM) . . . 17
2.5 Gated Recurrent Units (GRU) . . . 18
2.6 Feature scaling . . . 19
2.7 Overfitting . . . 19
2.7.1 Data split . . . 20
2.7.2 Regularization . . . 20
2.8 Hyperparameter tuning . . . 21
2.9 Feature importance . . . 22
2.9.1 Shapley additive explanations (SHAP) . . . 22

3. Data **24**
3.1 Data collection . . . 24
3.1.1 Collection setup . . . 24
3.1.2 Input signals and tyre temperatures . . . 24
3.1.3 Metadata . . . 24
3.1.4 Anti-aliasing and downsampling . . . 25
3.2 Feature selection . . . 27
3.2.1 Manually dropping signals . . . 27
3.2.2 Correlation analysis . . . 27
3.3 Tyre and ambient temperature . . . 27

3.4 Feature scaling 28
3.5 Reshape data 29
3.6 Selection of train, test, and validation data 31

4. Network design **33**
4.1 Performance metrics 33
4.2 Target signal 34
4.3 Baseline design and parameters 34
4.4 Finding the sequence length 36
4.5 Grid search 37
4.6 Final model structure 37
4.7 Feature importance 38

5. Prediction result **41**

6. Discussion **45**
6.1 Evaluating performance 45
6.2 Different tyres 46
6.3 Feature importance 46
6.4 Weather & Road condition 47
6.5 Dataset split 47
6.6 Choice of hyperparameters 47
6.6.1 Sequence length 48
6.7 Future work 48

7. Conclusion **50**

Bibliography **51**

1 Introduction

1.1 Background

Improving energy usage is key when developing electric vehicles. One very important factor is tyre rolling resistance (RR), whether it is for estimating remaining range, calculate environmental impact, or for calibration. Most standard methods of estimating RR, e.g. ISO 28580 [1], are performed under laboratory settings at steady state conditions. However, most people are not driving their cars at constant speed and at a constant ambient temperature. Estimating the RR in dynamic conditions has proven to be a fundamentally hard problem to solve, and while these standards do provide a systematic methodology to assess RR, they do not capture the actual RR under real-world driving conditions. In the pursuit of improving this, the relationship between RR and tyre temperature can serve as a significant piece of the puzzle. The work done by Schuring *et al.* [2] found that, all else fixed, an increment of one degree Celsius in tyre temperature leads to a 2% reduction in rolling resistance. As such, if tyre temperature under driving is known, then that might improve ones ability to assessing the dynamic RR.

Unfortunately, due to several factors, installing sensors with the ability to measure tyre temperature accurately on consumer cars can be troublesome and costly. Another approach is to, given all on-board signals such as vehicle speed, ambient temperature, and acceleration, explore if it is possible to model the tyre temperature based on those signals, creating what is called a *virtual* or *soft sensor*. Existing literature gives some insights in how different factors affect the tyre temperature. For example, it is known that higher a vehicle speed is strongly related to higher tyre temperature as shown in the analysis made by Wangs [3]. The same relationship but for trucks has also been shown [4]. Ejsmont *et al.* [5] stated that road temperature, which is partly controlled by the air temperature, and the existence of water or snow on the pavement, also influence the tyre temperature.

There are multiple different approaches to solving this problem, e.g. by defining and solving differential equations or creating an autoregressive–moving-average model with exogenous inputs model (ARMAX model) [6]. However, both mentioned methods assumes that the system can be sufficiently expressed in closed form, which might be a hard problem. Another possible approach is *Black-box* modelling, with the benefit that the model will hopefully learn the physics. But the benefit is also its biggest drawback, since it is hard to interpret the internal processes.

With that said, numerous articles and studies demonstrate how recurrent neural networks (RNNs) can be used to create black-box virtual sensors based on time series data. Alexandersson and Lönegren [7] created a virtual engine sensor by developing a LSTM model with good performance. RNN models proved again to be well suited for constructing a soft sensor as seen in the paper by Ke *et al.* [8], this time for modelling different quality measures in chemical industries. Another recent paper by Guesbaya *et al.* [9] also built a soft sensor using RNNs with the intent to help farmers regulate the conditions inside greenhouses.

1.2 Research goals

The goal of this thesis is to explore the possibility of developing a virtual sensor using RNNs for predicting tyre temperature during dynamic driving. As mentioned in the background, a one degree increase in tyre temperature leads to a 2% decrease in rolling resistance. If one wishes to use the tyre temperature to asses RR and reduce the margin of error compared to the standardized methods, it was decided to see if the virtual sensor can be within ±2 °C of the true temperature. To accomplish this, the model relied on signals that are typically implemented as on-board signals in a consumer vehicle. Additionally, an investigation was conducted to determine the significance of various input signals. In essence, the thesis attempted to answer two questions:

- Is it possible to build a virtual tyre temperature sensor with max ±2 °C of error?
- What is the relative importance of different features in the RNN model?

1.3 Limitations

There exists some very important limitations in this work. First, all the data have been gathered during the time period 2022-11-28 and 2022-12-08. This will most likely impact the generality of the models. If one wishes to create a more general model with the intent to perform well in all road and weather conditions, then one would need data captured in all possible scenarios. Another major limitation is that all data was gathered using the same car, with the same load and tyres. Naturally the model will most likely be biased towards that specific car. Also, the final model cannot be considered optimal due to our heuristic approach of hyperparameter tuning. This suggests that there is a significant potential for discovering improvements in future research.

1.4 Structure

This thesis is structured as follows: In Chapter 2, neural networks are introduced, including theory on LSTM and GRU. Chapter 3 provides details on the data collection and processing techniques. How the network was design and why is presented in Chapter 4. In Chapter 5, the outcomes of our model are presented, followed by Chapter 6, where these results and potential improvements are discussed. Finally, Chapter 7 concludes the work with a summarization of our findings.

2

Background on Neural Networks

2.1 Artificial Neural Networks (ANN)

Artificial neural networks (ANNs) stems from the principles of the human brain. In the brain, each neuron has an axon which will send an electrical impulse that is received by connected neurons. A neuron is stimulated if the cumulative strength of all the received electrical impulses is stronger than a threshold and will then in turn send an impulse to the connected neurons. Drawing from this biological model, the concept of ANNs was developed. However, before exploring ANNs, it is crucial to first understand the concept of an artificial neuron.

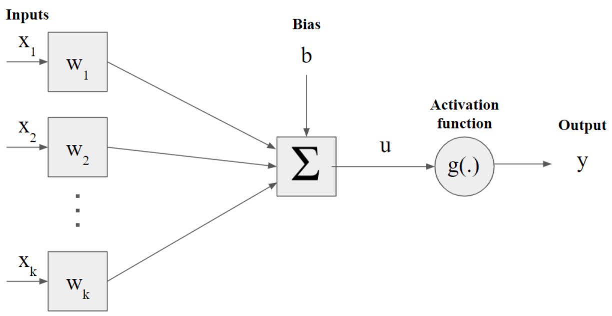


Figure 2.1: Artificial neuron.

In fig. 2.1, inputs x_1, x_2, ..., x_k and corresponding weights w_1, w_2, ..., w_k are multiplied and summed up plus an additional bias b to form the activation potential u, such that

$$u = \sum_{i=1}^{k} w_i x_i + b \tag{2.1}$$

An activation function g is then applied, often with the goal of limiting the output from the neuron within some desired range and capture nonlinearities. The output y is then

$$y = g(u) \tag{2.2}$$

There exists a multitude of different activation functions [10]. One common activation function is the hyperbolic tangent function (tanh), which maps u to the range -1 to 1, such that

$$g(u) = \frac{e^{2u} - 1}{e^{2u} + 1} \tag{2.3}$$

Another one is the sigmoid activation function which maps u to the range 0 to 1, with

$$g(u) = \frac{1}{1 + e^{-u}} \tag{2.4}$$

The simplest is just a linear activation function where the output is the potential activation u itself, i.e.,

$$g(u) = u \tag{2.5}$$

These neurons can then be connected to each-other in different ways to create various types of networks.

2.2 Multilayer perceptron (MLP)

Perhaps the most characteristic neural network is the Multilayer perceptron (MLP). It is a feedforward neural network, where the connections only flows forward and do not create cycles. It consists of one input layer, at least one intermediate (hidden) layer, and one output layer. In fig. 2.2, one can see a MLP with one input layer, two hidden layers, and one output layer. The input layer consists of two neurons, the first hidden layer consist of three neurons, second hidden layer two neurons, and the output layer only one neuron. All neurons in one layer connects to all neurons in the next layer, thus all layers are *fully connected* (FC).

For the specific network in fig. 2.2, let $\mathbf{x} = [x_1, x_2]^\mathrm{T}$ be inputs to the network, $\mathbf{W}^l$ be the l-th layer's weight matrix, with elements $w_{j,k}^{(l)}$ representing the connection between neuron j in layer l to neuron k in layer $l+1$. The bias vector for layer l is $\mathbf{b}^l$, the output of neuron i in layer l is a_i^l and y is final output of the network. All of

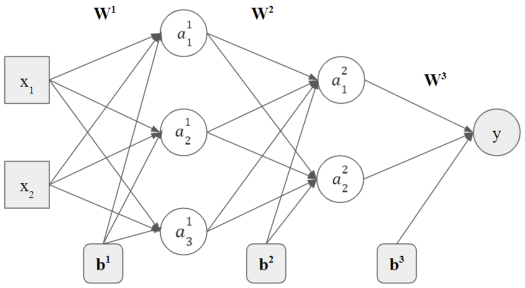


Figure 2.2: Example of a MLP network with one input layer, two hidden layers and one output layer.

the parameters can be expressed in matrix form:

$$\mathbf{W}^1 = \begin{bmatrix} w^{(1)}_{1,1} & w^{(1)}_{1,2} & w^{(1)}_{1,3} \\ w^{(1)}_{2,1} & w^{(1)}_{2,2} & w^{(1)}_{2,3} \end{bmatrix}, \quad \mathbf{W}^2 = \begin{bmatrix} w^{(2)}_{1,1} & w^{(2)}_{1,2} \\ w^{(2)}_{2,1} & w^{(2)}_{2,2} \\ w^{(3)}_{3,1} & w^{(3)}_{3,2} \end{bmatrix}, \quad \mathbf{W}^3 = \begin{bmatrix} w^{(3)}_{1,1} \\ w^{(3)}_{2,1} \end{bmatrix}$$
$$\mathbf{b}^1 = \begin{bmatrix} b^{(1)}_1 \\ b^{(1)}_2 \\ b^{(1)}_3 \end{bmatrix}, \quad \mathbf{b}^2 = \begin{bmatrix} b^{(2)}_1 \\ b^{(2)}_2 \end{bmatrix}, \quad \mathbf{b}^3 = \begin{bmatrix} b^{(3)}_1 \end{bmatrix} \tag{2.6}$$

Then, the output of all neurons and the final output of the entire network is

$$\mathbf{a}^1 = \begin{bmatrix} a^1_1 \\ a^1_2 \\ a^1_3 \end{bmatrix} = g_1\Big((\mathbf{W}^1)^\mathrm{T}\mathbf{x} + \mathbf{b}^1\Big), \quad \mathbf{a}^2 = \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix} = g_2\Big((\mathbf{W}^2)^\mathrm{T}\mathbf{a}^1 + \mathbf{b}^2\Big)$$
$$y = g_3\Big((\mathbf{W}^3)^\mathrm{T}\mathbf{a}^2 + \mathbf{b}^3\Big) \tag{2.7}$$

where g_l is activation function for layer l, which is applied elementwise to the input vector. The calculations, from input to output, is called a *forward pass*.

2.2.1 Backpropagation

The unknown parameters for the example model in Section 2.2 are the weights $w^{(l)}_{j,k}$ and the bias $b^{(l)}_i$ terms, and the goal is to find the optimal parameters that minimizes a predefined *loss function*. The loss L can be defined differently depending on the problem at hand. A popular choice in regression problems is the mean squared error (MSE):

$$L(\mathbf{x};\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i(x_i;\theta))^2 \tag{2.8}$$

where N is the total number of samples, y_i is the ground truth value, and $\hat{y}_i(x_i;\theta)$ is the model's prediction given input x_i and all parameters θ (weights and biases). Note that L is a function of θ, so it can be thought of as a surface in the multi-dimensional space spanned by the parameters. Finding a closed form solution here can be troublesome, if not impossible [11]. To solve this, the parameters are instead typically updated iteratively using a gradient decent algorithm during the training process by taking a step in the negative gradient direction w.r.t. θ of the loss function, i.e.,

$$\begin{aligned} \theta_{i+1} &= \theta_i - \varepsilon \nabla_\theta L(\mathbf{x};\theta), \\ \nabla_\theta L(\mathbf{x};\theta) &= \left[\frac{\partial L(\mathbf{x};\theta)}{\partial \theta_1}, \frac{\partial L(\mathbf{x};\theta)}{\partial \theta_2}, \dots, \frac{\partial L(\mathbf{x};\theta)}{\partial \theta_p}\right]^{\mathrm{T}} \end{aligned} \tag{2.9}$$

where ε is called the *learning rate* and p is the total number of weights and biases. The actual calculation of the gradient is called *backpropagation* [12]. It is worth noting in eq. (2.9) that all of the training data $\mathbf{x}$ is used to calculate one gradient descent step, which is called batch gradient descent. This requires a lot of memory if the training data is large, therefore optimization algorithms usually only use a few samples for every update. This subset of samples is called a *batch* and the number of samples in a batch is called *batch size*. Calculating the gradient for every batch is called minibatch gradient descent, commonly only referred to as stochastic gradient descent (SGD) [13]. Apart from SGD, there exists multiple different optimization algorithms that have shown promising empirical results, such as Adam [14]. After the gradients have been calculated and parameters updated for every batch, the training process is said to have finished one *epoch*.

2.2.2 Learning rate

The step size ε in gradient descent eq. (2.9) is called *learning rate*, which regulates how much the parameters should change every update and is a value between zero and one. As described by Goodfellow *et al.*, choosing the optimal learning rate is *"more of an art than science"* [13]. Too big and the minimum gets overshot, too small might lead the algorithm to a sub-optimal minimum or unnecessarily long training time. A common practise is to find an initial learning rate by trial and error, then implement a scheme which decreases the learning rate under the training process. By doing this, one quickly moves towards a minimum in the start then let it converge as the learning rate decreases. The Adam optimizer, previously mentioned in Section 2.2.1, creates one learning rate for every input, and implements this adaptive scheme for all of these [14].

2.2.3 Batch and batch size

As described in Section 2.2.1, a *batch* is defined as a subset of the total data used to update the model parameters according to eq. (2.9). The *batch size* is the number of samples in a batch, typically a power of 2 to enable faster processing on GPUs [13].

The choice of batch size can have a significant impact on the performance and is also related to the learning rate. Large batch sizes increase the accuracy of the gradient at the cost of higher memory requirement. Small batch sizes may not give totally accurate gradients, but introduces some regularization effects (more on that in Section 2.7.2). Since the gradient is more accurate for larger batches, it typically allows for higher learning rates to be used. Conversely, it might be necessary to use a relatively smaller learning rate for smaller batches to maintain stability due to high variance in the gradient estimates [13].

As with the learning rate, there is no predefined right or wrong answer. Some models get good performance with relatively small batch sizes (from 2 to 32) [15], others with very large batch sizes (up to 8192) [16]. As such, it needs to be optimized for the problem at hand, often by trial and error.

2.3 Recurrent Neural Networks (RNN)

The RNN is an extension of the traditional feed-forward neural network and is similar to the MLP described in Section 2.2. The difference is that RNNs are designed to handle sequential data and can capture dependencies from previous time steps, not only the current. This is achieved by adding a feedback loop that updates a hidden state which serves as a memory from previous states. To visualize RNNs, they are typically unfolded over time, as seen in fig. 2.3.

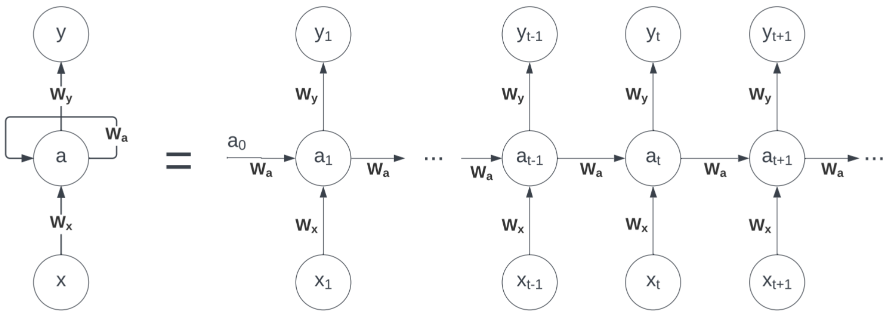


Figure 2.3: Unrolled RNN.

Calculating the forward pass is done in the same way as for a regular feed-forward network but with an additional term for the dependency of the previous state. This can be written as:

$$a_t = g_a(W_x x_t + W_a a_t) \tag{2.10}$$

$$\hat{y}_t = g_y(W_y a_t) \tag{2.11}$$

Note that all weights and biases are shared across all inputs, which also leads to the vanishing/exploding gradient problem, the main reason why these standard RNNs are not used that frequently [17]. The problem arises during weight updates as gradients from several layers back in time are included. If the weight is greater than 1, the term explodes. Conversely, if the weight is less than one, the gradients will be close to zero and lead to slow optimization. Additionally, as the gradients for steps further back become smaller, they will not contribute much to the training, making it difficult to capture long-term dependencies. There are ways around this problem, one of them is Long-Short Term Memory (LSTM).

2.4 Long-short term memory (LSTM)

Long-short term memory (LSTM) is a type of recurrent neural network that is capable of learning long term dependencies in sequential data and was first proposed in 1997 by Hochreiter and Schmidhuber [18]. The main idea is to use memory cells that stores information over time and to use different paths for long and short term memories. These cells are connected through gates that control the flow of information, and there exists three different gates: forget gate, input gate and output gate. The structure inside a LSTM cell is seen in fig. 2.4.

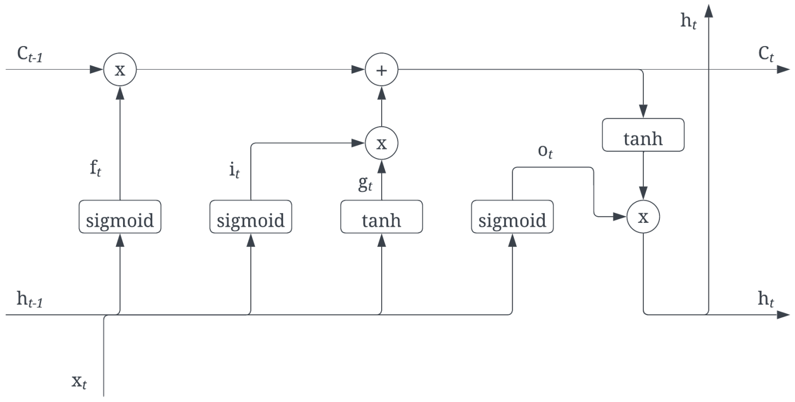


Figure 2.4: LSTM cell.

The forget gate f_t determines how much of the previous cell state C_{t-1} one should remember and is a value between 0 and 1, where 0 would completely ignore the cell state and 1 would completely accept. It is calculated as

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.12}$$

where σ is the sigmoid activation function defined in eq. (2.4). The input gate determines how one should update the cell state, and is divided into parts. First one calculates a candidate activation g_t, then input gate i_t, and finally f_t, g_t and i_t is used to update the cell state C_t.

$$g_t = \tanh(W_g x_t + U_g h_{t-1} + b_g) \tag{2.13}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.14}$$

$$C_t = f_t C_{t-1} + i_t g_t \tag{2.15}$$

The final stage of the LSTM is the output gate, basically determining what the output h_t of the LSTM cell should be, with

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.16}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{2.17}$$

In eqs. (2.12) to (2.17), the parameters (W_f, W_g, W_i, W_o) and (U_f, U_g, U_i, U_o) are trainable weights and (b_f, b_g, b_i, b_o) are trainable biases.

2.5 Gated Recurrent Units (GRU)

GRU is another approach to solve exploding/vanishing gradient and can be viewed as a simpler version of LSTM [17]. Here only two gates are used, one called update gate z_t and another called reset gate r_t. The update gate determines how much of the previous state should be passed to the current time step. The reset gate decides how much of the previous hidden state to forget. The structure inside a GRU cell can be seen in fig. 2.5.

Similar to RNNs, at every time step both the input x_t and the previous state h_{t-1} is available. Out of these two, both the update and reset gate can be calculated

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \end{aligned} \tag{2.18}$$

Here, σ represent the sigmoid activation function defined in eq. (2.4). The activation h_t of the GRU is then a linear interpolation between the previous h_{t-1} and a candidate activation $\tilde{h}_t$, such that

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t \tag{2.19}$$

where the candidate activation is defined as

$$\tilde{h}_t = \tanh(W_{\tilde{h}} x_t + U_{\tilde{h}}(r_t \cdot h_{t-1}) + b_{\tilde{h}}) \tag{2.20}$$

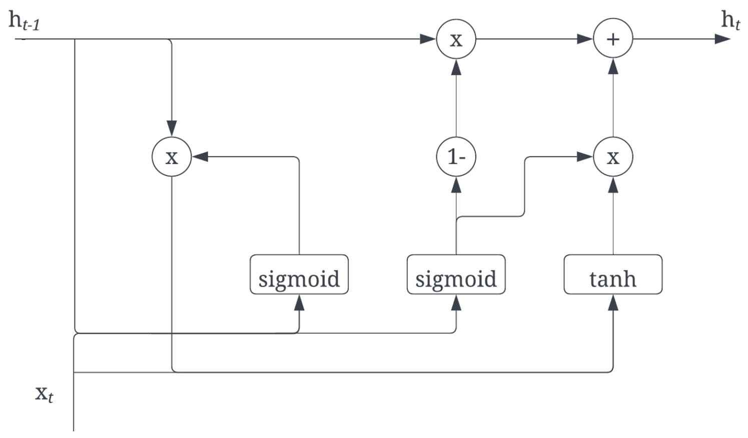


Figure 2.5: GRU cell.

2.6 Feature scaling

Machine learning algorithms struggle to perform well with different numerical scales [11]. To improve stability, feature scaling is applied and is a crucial transformation needed to be applied on the data before training a neural network. One of the most common method is *min-max scaling*.

Min-max scaling adjusts all values into a predetermined range, most often $[0,1]$. Assume a set of real values $\mathbf{X}$, then one can min-max scale the set by applying the function f in eq. (2.21) to each value:

$$x_{scaled} = f(x) = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad \forall x \in \mathbf{X} \tag{2.21}$$

where x_{max} and x_{min} is the maximum and minimum value in $\mathbf{X}$.

2.7 Overfitting

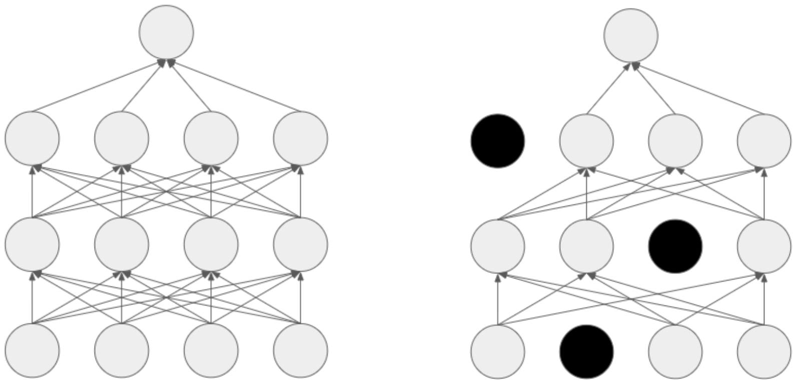
Apart from scaling, another very important phenomenon to consider is Overfitting. This is when a model learns the training data too well, capturing noise and inaccuracies rather than the underlying patterns. As a result, it performs poorly on new, unseen data. Overfitting occurs when the model is too complex or trained for too long, leading to reduced generalization capabilities [13]. To prevent this behaviour one can split the data into different parts and use different *Regularization* strategies.

2.7.1 Data split

The purpose of dividing the dataset into training, validation, and test sets is to assess the model's performance and generalization capabilities effectively. By splitting the data, the model can be trained on one subset, fine-tune its hyperparameters using another subset, and obtain an unbiased estimate of its performance on unseen data using the third subset [11]. The dataset used for training is called *Training data*. To evaluate the models performance during training one uses the *Validation data*. After model training and hyperparameter tuning, the true performance of the model is measured on the *Test data*.

2.7.2 Regularization

There exists various regularization techniques, one of them is **Dropout**. During the training process, dropout randomly ”drops” or deactivates a proportion of neurons (along with their incoming and outgoing connections) in each layer of the network at every iteration, as shown in fig. 2.6. This prevents the neurons from co-adapting or relying too heavily on each other, effectively forcing them to learn more robust and independent features from the input data [19].



(a) Without dropout **(b)** With dropout

Figure 2.6: Two neural networks, both with two hidden layers. One without dropout (a) and one with dropout applied (b).

Another regularization technique is called **early stopping**. During the training process, both the training error and validation error typically decrease with each epoch. However, there might be a point where the model starts to overfit on the training data, characterized by a continued decrease in training error but an increase in validation error. Early stopping involves monitoring the validation error and halting the training when the error fails to improve for a specified number of consecutive epochs, called *patience*. The model parameters are then reverted back to the ones that achieved the lowest validation error. By stopping the training process when the validation error has not improved for some epochs (specified by the patience), the model generalizes better to unseen data [11]. The generic implementation of the scheme is visualized in fig. 2.7.

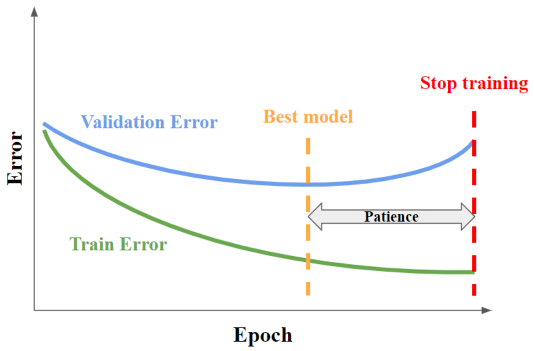


Figure 2.7: Example of how the train and validation error changes every epoch. To implement early stopping, the training is stopped when the validation error has not improved for a specific amount of epochs. How long you continue to train after minimum validation error is called patience.

2.8 Hyperparameter tuning

Fine-tuning the hyperparameters is an essential aspect of neural network development as it significantly influences the final model's performance. Hyperparameters, unlike weights and biases, are fixed before training and do not learn during the training-process. These parameters control various aspects of the model, such as its architecture (number of layers, units and dropout rate), learning rate, and batch size. A common technique used to find hyperparameters is called *grid search* [12]. In this method a set of values is selected for each parameter and every possible combination of these parameters is tested iteratively and evaluated on the validation data. In fig. 2.8, the different combinations are illustrated for a 2D problem. One drawback of grid search is that it becomes very time-consuming, as the number of iterations grows exponentially with the number of hyperparameters included. This can pose a challenge when tuning a multitude of hyperparameters or when dealing with a large set of values, making the process inefficient. Furthermore, a grid search does not explore the space between values within the chosen set, unless an additional search is carried out for the space between the best performing parameters.

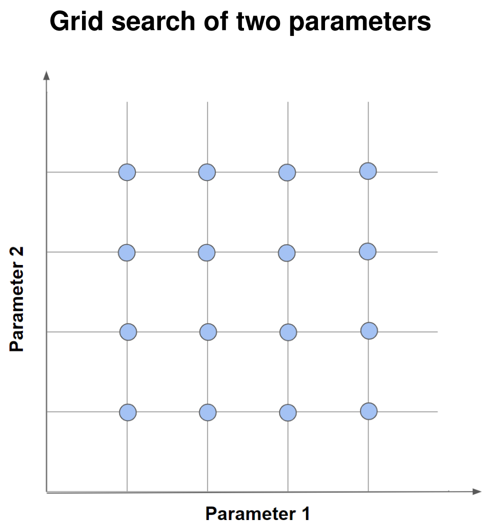


Figure 2.8: Different combinations of parameters 1 and 2 that are evaluated using grid search.

2.9 Feature importance

A drawback with using neural networks is the limited understanding offered on the logic behind the models decision making. Especially when compared to simpler and more interpretable models such as linear regressions or physic-based models. However, complex and nonlinear relationships over time in turn calls for more complex models to explain them, trading off interpretability for accuracy. Investigating the importance of each input signal for the decision making of a neural network can therefore be important and yield great insight into what otherwise is considered a ”black box”.

2.9.1 Shapley additive explanations (SHAP)

SHapley Additive exPlanations (SHAP), originally proposed by Lundberg and Lee [20], is a unified approach to interpreting the output of any machine learning model. It assigns each feature an importance score (SHAP value) for a particular prediction based on Shapley values from cooperative game theory. Given a prediction model, for a particular instance, a SHAP value quantifies the contribution of a feature towards changing the model's output from its base (expected) output. It guarantees that the sum of the SHAP values for all features will exactly equal to the difference between the model prediction and the base output. The formula for calculating the SHAP value ϕ for a feature i is as follows:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(N-|S|-1)!}{N!} (f_S(x_{S \cup \{i\}}) - f_S(x_S)) \tag{2.22}$$

where:

- N is the set of all input features.
- S is a subset of N not containing feature i.
- f_S is the model function conditioned on the subset S.

However, this formula has a computational complexity that grows exponentionally with the number of input features, which is not feasible for high-dimensional data. Hence, approximation methods are used to calculate the SHAP values. One such method is the Gradient Explainer method, which is an extenstion of the Integrated Gradient algorithm proposed by Sundararajan *et al.* [21]. For a given input x to the model, the Gradient Explainer estimates SHAP values by using the gradients of the model predictions at x, essentially providing a linear approximation of the model in the neighborhood of x.

The overall importance of a feature can then be seen as the mean of the absolute SHAP values for each feature. This can be calculated as in eq. (2.23) below:

$$\mathrm{FI}_i = \sum_{j=1}^{m} \frac{|\phi_{i,j}|}{m} \tag{2.23}$$

where:

- FI_i is the Feature Importance for feature i.
- m is the number of instances.

3 Data

The performance of a model ultimately hinges on the quality of the data it is trained on. It is essential to collect, clean, and process the data in a proper manner to achieve any meaningful results.

3.1 Data collection

3.1.1 Collection setup

The data was collected by Volvo Cars using a single car with the same load and tyres for each drive. A total of 55 drives of various routes and lengths in the Gothenburg area were carried out by 13 different drivers in different weather conditions. The time window of the data collection spanned between late November and early December in 2022.

3.1.2 Input signals and tyre temperatures

The on-board signals were obtained from control units of the car. Some signals were directly measured, while others were derived from these measurements.

Wheel-mounted sensors, specifically the Izze-Racing Tire Temperature & Pressure Monitoring System (TTPMS) [22], were used for measuring the tyre temperatures. Each sensor measured 16 different temperatures through different channels. The signals were not continuous, rather the sensor made a reading every 5 seconds and between the readings the signals were constant. As seen in fig. 3.1, it is clearly visible that there exists a large spread between them. The same sensors were used by Ydrefors [23] and that study concluded that the spread was expected, and it was not analyzed further.

3.1.3 Metadata

Some metadata was also collected in addition to the collection of data from control units and sensors. After each drive, the driver answered two questions:

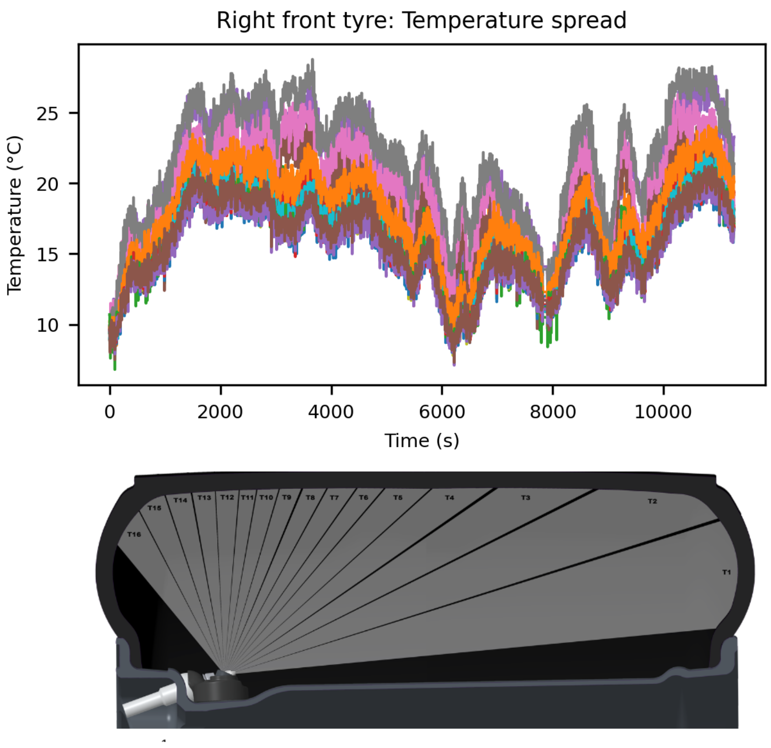


Figure 3.1: How the temperature measurement is spread among all channels (top), where different colors represents different channels. The bottom figure shows how the sensor is mounted in the tyre and each channels respective measurement area.

1. Which option is best suited to describe the road conditions during the drive?
   - Options: *Dry* / *Mixed* / *Wet*
2. Which option is best suited to describe the weather during the drive?
   - Options: *Sunny* / *Cloudy* / *Rainy*

The summary of the answers to the questions above can be seen in Table 3.1

As seen, the largest amount of data was collected when it was cloudy and when the state of the road was mixed, i.e., sometimes dry, other times wet.

3.1.4 Anti-aliasing and downsampling

Four different sampling rates were used to gather the different signals in the raw data. The tyre temperature measurements were sampled with 60ms interval, most

	Sunny	Cloudy	Rainy	Total
Dry	0	14	0	14
Mixed	4	31	2	37
Wet	0	3	1	4
Total	4	48	3	**55**

Table 3.1: Summary of the answers.

of the internal signals with 1ms interval, then only a few with either 10ms or 100ms. Depending on whether or not a signal was deemed to be noisy (determined by visual inspection), it was either downsampled after applying an anti-aliasing filter or only downsampled by interpolating the signal to a common sampling frequency. After trying a few different, it was decided to downsample the data to 1Hz. In fig. 3.2, one can see two examples of downsampled signals compared to the raw data.

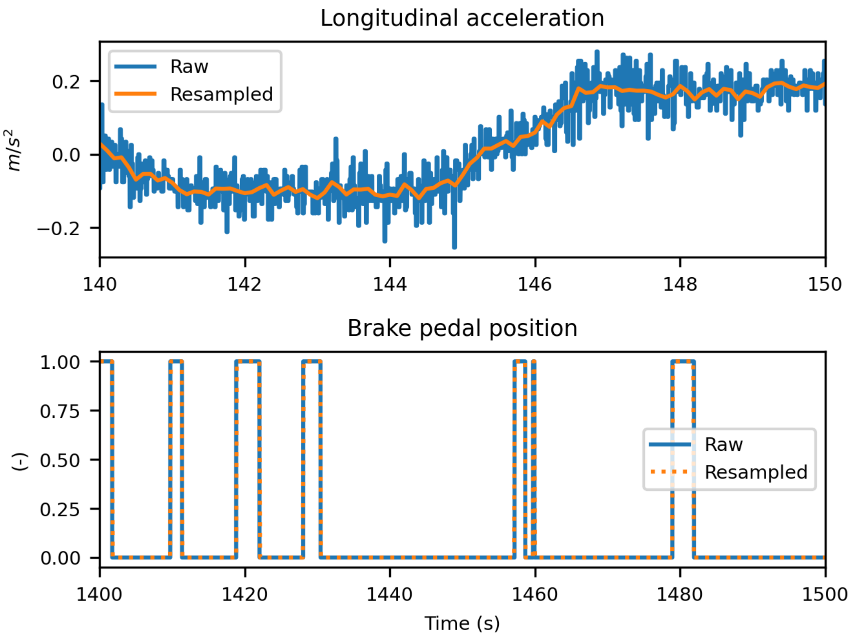


Figure 3.2: Comparison between the raw and resampled signals for longitudinal acceleration (top) and brake pedal position (bottom). Different time scales have been used to visualize the prevalence of high frequency noise in longitudinal acceleration but not in brake pedal position.

3.2 Feature selection

3.2.1 Manually dropping signals

In the raw data that was provided, 126 signals were sampled in 55 different drives. To make the data useful in other potential subject areas, all possible signals were sampled. That also meant that a lot of signals had to be dropped due to clear irrelevance for this particular work. An initial screening of signals was done where the reason was one of either:

1. Redundant signals (e.g. multiple signals measuring speed).
2. Constant signals.

If signals did fall into one of above categories, they were dropped and not analysed further. Also, *Vertical acceleration* and *Battery temperature* was dropped. The former due to having too high frequency components and cannot be accurately represented when downsampled. The latter was deemed irrelevant for predicting tyre temperature, so it was removed to reduce both training time and memory requirement.

3.2.2 Correlation analysis

To investigate the relation between the remaining input the correlation between all pairs of signals was calculated. In fig. 3.3 the correlation matrix before removing highly correlated signal is presented, and in fig. 3.4 the matrix after removing highly correlated signals can be seen.

It was observed that a few signals are highly correlated. As no information is added by including truly redundant signals [24], some where dropped. *Actual* and *requested front axle propulsion* had an almost 100% correlation with *longitudinal acceleration*. It was decided to keep the acceleration signal and drop the two others. *Requested brake friction* and *total vehicle drag inc. friction brakes* also correlated totally, thus it was decided to keep the latter. After manual screening and this correlation analysis, the resulting features that are available to use as inputs are listed in Table 3.2.

3.3 Tyre and ambient temperature

To handle the spread among the different channels seen in fig. 3.1, one signal was created by taking the average over all 16 channels. This was also done to mitigate potential noise. To make the signal continuous, a low-pass filter [25] was applied to the average signal where the cutoff frequency was set to 5 mHz. The ambient

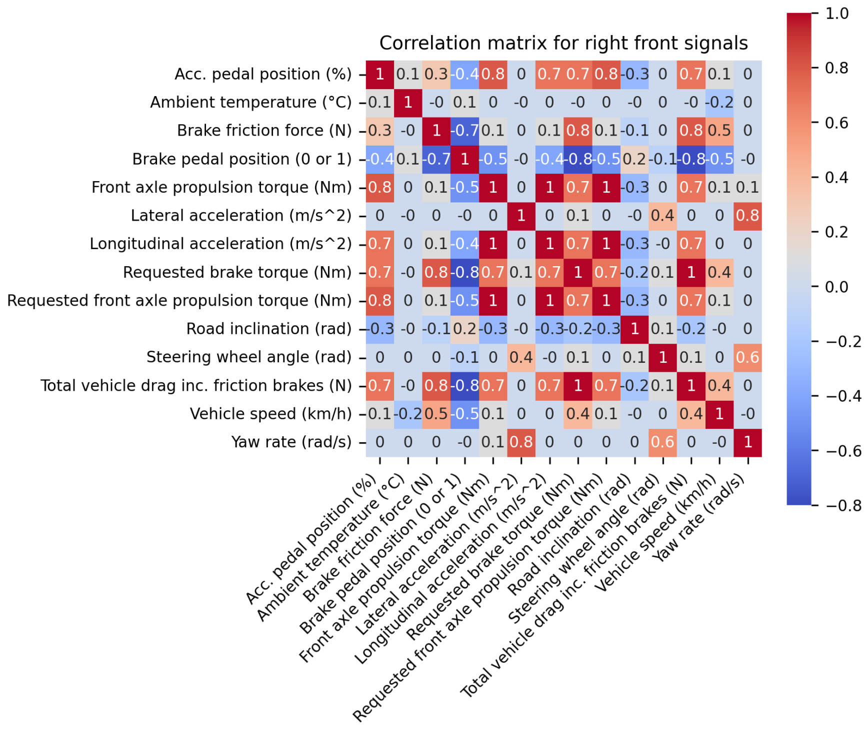


Figure 3.3: Correlation matrix before dropping highly correlated signals.

temperature signal shared the similar discontinuous characteristic as the tyre temperature, thus exactly the same filter and cutoff frequency was applied. The filtered and unfiltered signals can be seen in fig. 3.5.

3.4 Feature scaling

It was mentioned in Section 2.6 that machine learning algorithm might perform poorly if the input features are of different magnitudes. It can be seen in Table 3.3 that the input signals are indeed of different scales, and thus scaling is required.

All inputs, except Brake pedal position since it is binary, was Min-max scaled according to eq. (2.21). The scaling was first done on the training data set and the scalars x_{min} and x_{max} for each input feature were saved. These where later used to scale the validation and test data.

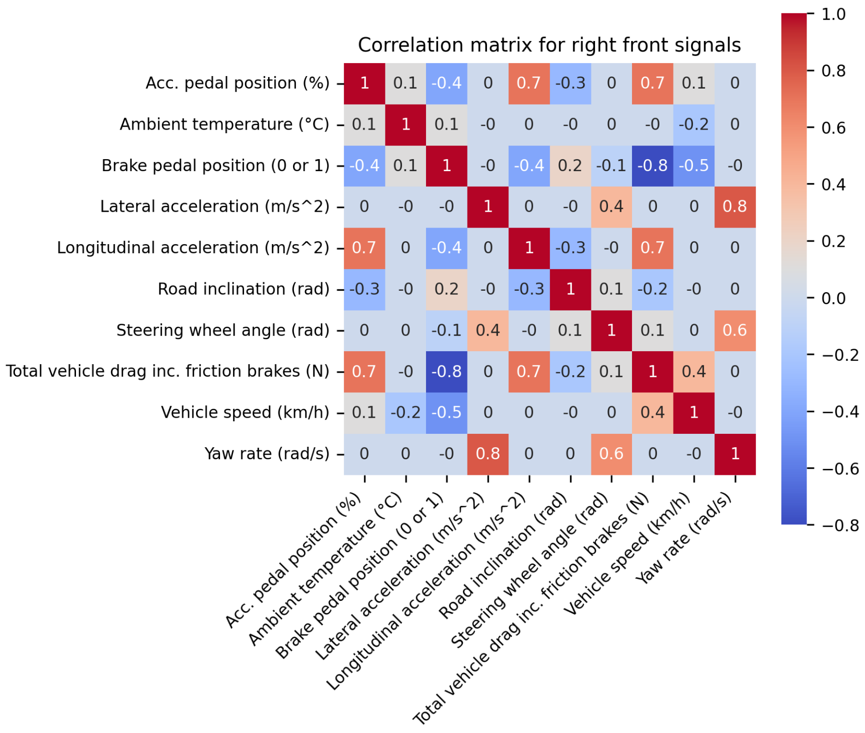


Figure 3.4: Correlation matrix after dropping highly correlated signals.

3.5 Reshape data

In order to input not only the current samples, but previous history, the data was reshaped in the form of a 3-dimensional matrix with the following dimensions: (samples, time steps, features):

- **Samples:** The total number of sequences in the dataset.
- **Time steps:** The sequence length, which is the number of time steps (observations) in each sequence.
- **Features:** The number of input features in each time step.

In mathematical terms, let n be the number of samples in one dataset, k be the number of past time steps one wants the model to take into consideration when making predictions and i be one specific feature, here labeled from 1 to p. Let $\mathbf{x}^i_{t:t+k}$

Signal Name	Description
Accelerator pedal position (%)	Accelerator pedal pressed as a percentage of max position.
Ambient temperature (°C)	Outside temperature of the car.
Brake pedal position	Binary signal, 1 if brake pedal is pushed in, 0 otherwise.
Lateral acceleration (m/s^2)	Lateral acceleration of the car.
Longitudinal acceleration (m/s^2)	Longitudinal acceleration of the car.
Road inclination (rad)	Road inclination, negative for uphill and positive for downhill.
Steering wheel angle (rad)	Steering wheel angle relative to center. Positive for left turns, negative for right turns.
Total vehicle drag including friction brakes (N)	Total vehicle drag force including friction brakes.
Vehicle speed (km/h)	Vehicle speed over ground in the direction of travel.
Yaw rate (rad/s)	Rate of change of the vehicle heading angle.

Table 3.2: Selected signals that are used as features in the model, and their description.

be the sequence of values from time t to $t+k$ for one feature i. Then, the reshaped data $\mathbf{X}$ is constructed as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^1_{1:k} & \mathbf{x}^2_{1:k} & \cdots & \mathbf{x}^p_{1:k} \\ \mathbf{x}^1_{2:k+1} & \mathbf{x}^2_{2:k+1} & \cdots & \mathbf{x}^p_{2:k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}^1_{n-k+1:n} & \mathbf{x}^2_{n-k+1:n} & \cdots & \mathbf{x}^p_{n-k+1:n} \end{bmatrix} \tag{3.1}$$

Let y_t be the temperature for one tyre at time t. Then, the corresponding target values $\mathbf{y}$ are

$$\mathbf{y} = \begin{bmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_n \end{bmatrix} \tag{3.2}$$

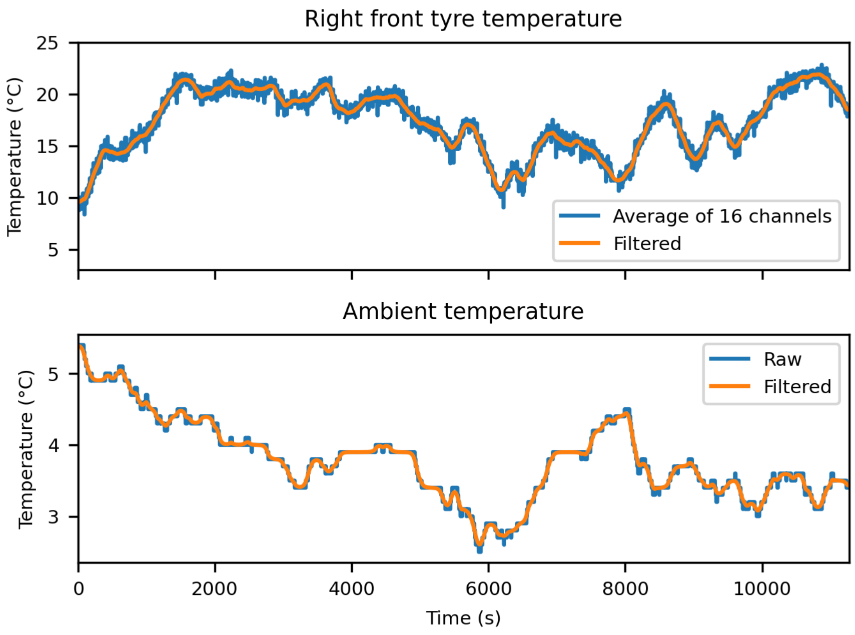


Figure 3.5: In the top plot, the blue signal is the average temperature over all 16 channels for the left rear tyre. The orange signal the filtered average signal. In the bottom plot, the blue signal is the raw ambient temperature, whereas the orange is the filtered signal.

Finally, let the m:th row in **X** be defined as $\mathbf{X}_m$, then the problem is to find the function f that satisfactory can predict y_m, i.e.,

$$f(\mathbf{X}_m) \approx y_m \tag{3.3}$$

As shown in eq. (3.1), the rows of X may have significant overlap. To accelerate the training process, every other row in **X** was dropped.

3.6 Selection of train, test, and validation data

Out of all the data, 80% was used to train the model, 10% as validation data, and the remaining 10% as test data. Drives from different conditions was included in all the datasets in the pursuit of making the different sets as general as possible. As such, the drives to be included in the validation and test sets were picked randomly but with two constraints:

Signal Name	**Mean**	**Std.Dev**	**Min**	**Max**
Vehicle speed (km/h)	49.5	24.0	-0.090	92.0
Lateral acceleration (m/s^2)	0.127	0.693	-5.58	5.80
Road inclination (rad)	0.004	0.026	-0.104	0.114
Yaw rate (rad/s)	0.008	0.073	-0.707	0.680
Longitudinal acceleration (m/s^2)	-0.046	0.635	-8.11	4.17
Total vehicle drag inc. friction brakes (N)	59.1	2060	-19200	10600
Steering wheel angle (rad)	0.028	0.907	-8.45	8.55
Acc. pedal position (%)	15.1	15.6	0	100
Ambient temperature (°C)	3.78	0.552	2.57	5.38
Brake pedal position (0 or 1)	0.145	0.352	0	1

Table 3.3: The mean, standard deviation, min and max values for each input feature before scaling.

- All conditions needs to be represented. Specifically, one drive when it was sunny, one when the pavement was dry, and one when the pavement was wet.
- The size of the validation and test sets should each be approximately 10% of the total amount of data available.

This resulted in 45 drives in the test data and 5 drives in both validation and test data. Note that the relative number of drives is not 10% but the size is.

4

Network design

4.1 Performance metrics

In order to work out a good network design, it is necessary to use performance metrics to compare one design to one another. There are different ways of evaluating the performance of a machine learning model as described in Section 2.2.1. In addition to the mean squared error (MSE) defined in eq. (2.8), a related metric is the root mean squared error (RMSE). Mathematically, it is the standard deviation of the residuals.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{4.1}$$

Another popular metric when it comes to regression problems is the coefficient of determination, often denoted as R-squared or R^2. It is defined as the proportion of the variance in the dependent variable that is explained by the predicted values from the regression model [26]. It is a value between 0 and 1, with higher values indicating a better fit, defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.2}$$

As the goal of the model (stated in Section 1.2) is to be within ±2 °C, a custom accuracy score was created. It is defined as the proportion of predicted values that fall within a certain temperature threshold T of the true temperature:

$$\begin{aligned} \mathbb{1}_{\{|y_i - \hat{y}_i| \leq T\}} &= \begin{cases} 1, & \text{if } |y_i - \hat{y}_i| \leq T \\ 0, & \text{otherwise} \end{cases} \\ \text{Accuracy}(T) &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{|y_i - \hat{y}_i| \leq T\}} \end{aligned} \tag{4.3}$$

4.2 Target signal

As mentioned in Section 3.1.2, temperature data for all four tyres was available. Furthermore, out of all available input features specified in Section 3.2.2, none is axle or tyre specific. Hypothetically, if the temperatures for each tyre during driving were somewhat close to each other, only one model would be necessary. However as seen in fig. 4.1, the front wheels are consistently higher than the rear wheels (due to the test car being a front wheel driven car), but they do follow the same patterns. To limit the extent of this thesis, the network design will be built by having the right front tyre temperatures as target signal. To check if this design can be used for another tyre, the final model design will be retrained on left rear temperatures, hopefully achieving equal performance.

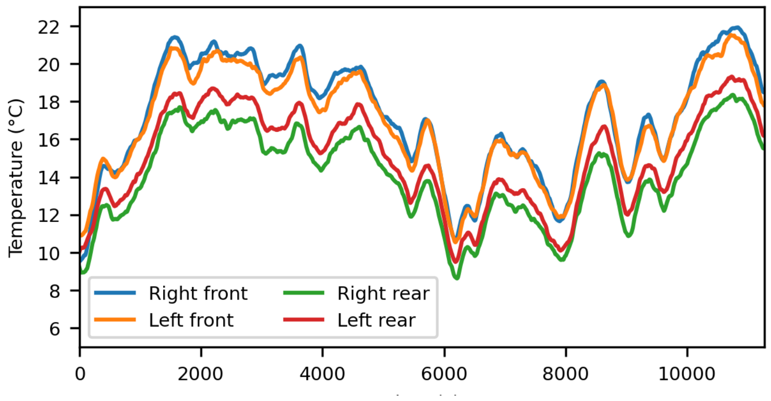


Figure 4.1: Temperature for each tyre under one drive.

4.3 Baseline design and parameters

The number of possible hyperparameter combinations were too many for all to be explored. To limit the scope of this project a heuristic approach was taken where not all hyperparameters were tuned, and only a few values for those chosen. The decisions were taken by evaluating the resulting mean squared error of the predictions on the validation data (validation loss).

Our process of finding a good combination can be though of as traversing down a directed rooted tree, where the root node is the baseline model configuration and the height is the number of hyperparameter decisions. A decision can be based on trial and error or grid search. Every leaf is a unique combination of parameters and one or more parameter is changed at every level. This saves time, but leaves lots of combinations unexplored. For example, in fig. 4.2 the process for a model with five parameter, each with six different options, is visualized.

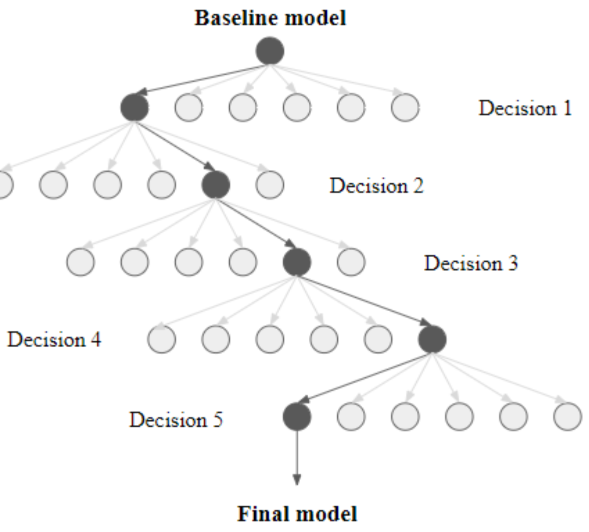


Figure 4.2: Process of finding good hyperparameters for a model with five parameters and six options for every parameter. The root node is the starting combination of hyperparameters and at every level one or more parameters are optimized and then fixed to these values.

The baseline combination of hyperparameters was found by trial and error and used as a starting point for further optimization, the specific values for each parameter can be seen in Table 4.1. Note that both the number of epochs and patience is defined, this is because the training process stopped after 40 epochs or if the validation loss fails to decrease over 5 consecutive epochs.

Hyperparameter	**GRU/LSTM**
Batch size	256
Learning rate	10^{-4}
Layer	1
Units/Layer	128
Dropout	0.2
Optimizer	Adam
Loss function	MSE
Epochs	40
Patience	5

Table 4.1: Initial model configuration used as a starting point for further optimization.

The hyperparameters that will be tuned in this work are

- Sequence length
- Dropout
- Units

4.4 Finding the sequence length

The sequence length is the number of historical datapoints that will be used as inputs to the model. During training, the LSTM/GRU network learns to incorporate historical information, so current tyre temperatures relies not only on the immediate inputs but also on multiple prior data points. As a result, selecting an optimal sequence length is crucial.

However, increasing the sequence length to achieve better performance is in conflict with the goal of maintaining model simplicity, and with the fact that the model needs to wait longer before any predictions can be made. For example, if a sequence length of 600 seconds is used, then 600 seconds of driving history is required to make a tyre temperature prediction. The model will therefore not be able to predict the tyre temperature for the first 10 minutes of every drive. Thus, not only the performance of each sequence length was considered, but also the waiting time required to generate predictions.

To find a suitable sequence length, the baseline GRU and LSTM models where trained using different sequence lengths. During training the models were evaluated after each epoch by calculating the validation loss. The minimum validation loss for every sequence length, for both models, was also determined. In fig. 4.3, the validation loss for each epoch and sequence length is illustrated for both LSTM and GRU models. The lowest validation loss attained by each model at different sequence lengths can be observed in fig. 4.4.

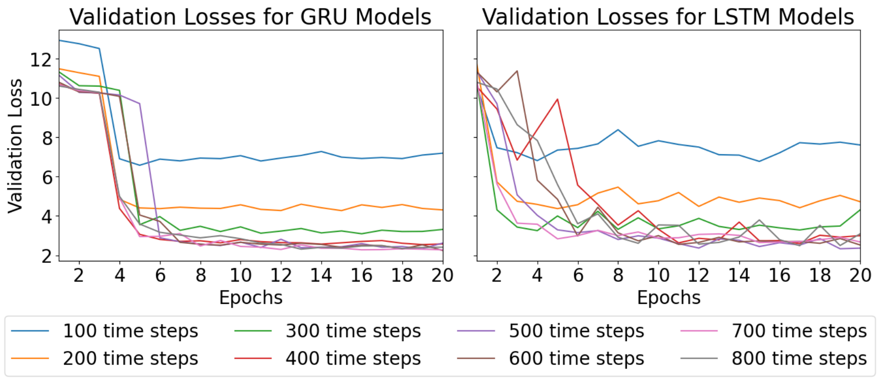


Figure 4.3: Validation losses (MSE) for sequence lengths varying between 100-800 time steps.

Looking at fig. 4.4, the GRU model seems to perform best with a sequence length of 600, with 500 being ever so slightly worse. The small performance increase by using more history was deemed less important than model simplicity and longer waiting time, thus it was decided to use 500 seconds of history. For the LSTM hovever, it is clear from the same figure that a sequence length of 500 time steps is optimal. As such, a sequence length of 500 time steps was used for both models.

4.5 Grid search

Four different values for the number of units and five different dropout rates was tested for both LSTM and GRU, resulting in 40 different models. The performances of each model represented by the MSE on the validation data, are summarized in Table 4.2.

The best performing model-combination of units, dropout and model type seems to be a LSTM model with 256 units and 0.3 in dropout. With this combination, the model reached a MSE of 2.082 on the validation data.

4.6 Final model structure

The final model is then a LSTM with 256 units and a dropout rate of 0.3 followed by a fully connected layer to create the final output. A more detailed view of the pa-

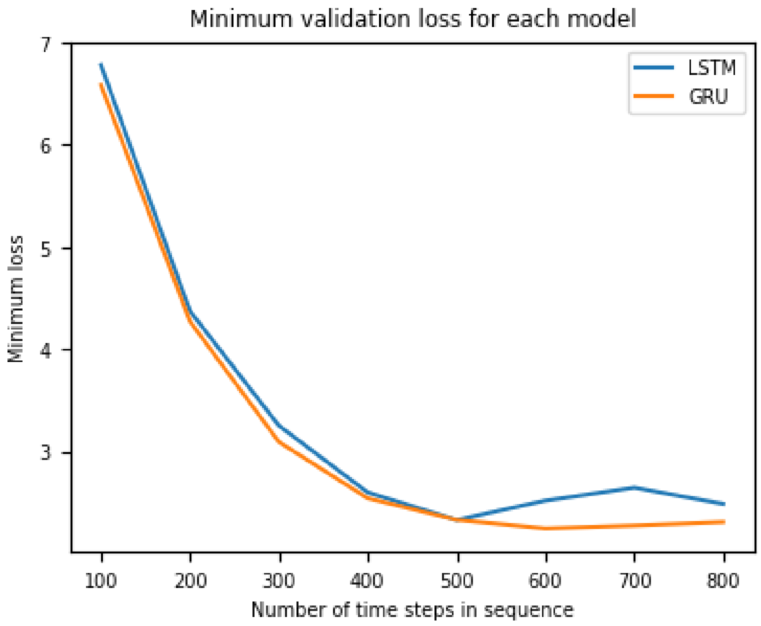


Figure 4.4: Minimum validation loss (MSE) for both models for each sequence length.

rameters is presented in Table 4.3, and a view over the network structure in fig. 4.5.

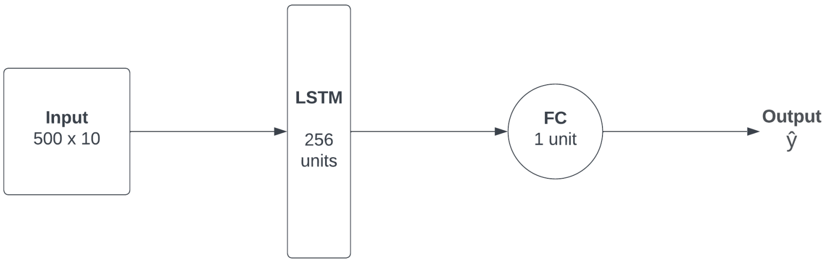


Figure 4.5: Structure of the final LSTM model.

4.7 Feature importance

After training the final model the importance of each feature was calculated. This was done using the gradient explainer method to estimate the SHAP values for 1000

Units	Dropout	MSE-LSTM	MSE-GRU
64	0	2.382	2.377
64	0.1	2.522	2.281
64	0.2	2.502	2.337
64	0.3	2.279	2.467
64	0.4	4.962	2.866
128	0	2.513	2.656
128	0.1	2.303	2.287
128	0.2	2.205	2.283
128	0.3	2.117	2.335
128	0.4	2.785	2.509
256	0	2.206	2.527
256	0.1	2.276	2.308
256	0.2	2.327	2.310
256	0.3	**2.082**	2.328
256	0.4	2.393	2.381
512	0	2.530	2.443
512	0.1	2.295	2.307
512	0.2	2.455	2.374
512	0.3	2.274	2.298
512	0.4	2.448	2.371

Table 4.2: The MSE on the validation dataset for different combinations of units, dropout, and model type. The lowest MSE is hightlighted in bold symbols.

Hyperparameter	Choice
Model type	LSTM
Batch size	256
Learning rate	10^{-4}
Layer	1
Units/Layer	256
Dropout	0.3
Optimizer	Adam
Loss function	MSE
Patience	10

Table 4.3: Final model design.

randomly drawn instances of the validation data. The mean of the absolute SHAP values represent the importance of the corresponding feature. This result can be seen plotted in fig. 4.6. It is clear that some signals have a very large impact on the output, whilst others are almost insignificant.

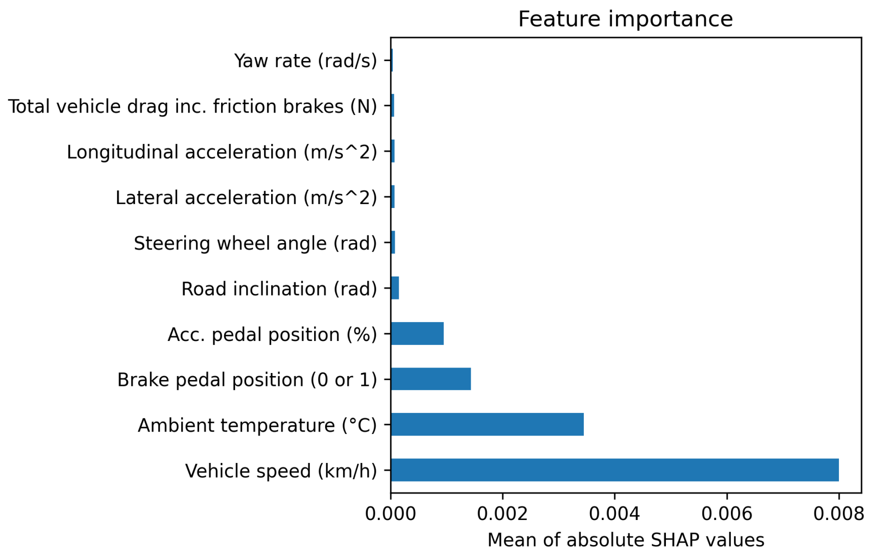


Figure 4.6: Mean of the absolute SHAP values for each input feature, using 1000 randomly drawn instances. The features are plotted in terms of importance in ascending order.

To test if the features with the lowest importance score were contributing to the output, a new model was trained and compared to the previous optimal model. This model was trained using the same optimal parameters, but without the following features with the lowest importance:

- Lateral acceleration
- Steering wheel angle
- Longitudinal acceleration
- Total vehicle drag inc. friction brakes
- Yaw rate

5 Prediction result

Three different models were created and evaluated, all with the same model design specified in Table 4.3.

- **Model 1** used all 10 inputs specified in Table 3.2 and had the right front tyre temperatures as target signal.
- **Model 2** used the same inputs as model 1, but had the left rear tyre temperatures as target signal.
- **Model 3** used only the 5 most important features seen in fig. 4.6, i.e., road inclination, accelerator pedal position, brake pedal position, ambient temperature, and vehicle speed. The target signal for model 3 was the right front tyre temperatures.

These models were then evaluated on the test dataset based on their respective RMSE, accuracy, and R-squared values, defined in eqs. (4.1) to (4.3). The results are presented in Table 5.1.

	RMSE	**Accuracy($T = 2$)**	**R-squared**
Model 1	1.322±0.028	0.894±0.007	0.831±0.009
Model 2	1.318±0.030	0.902±0.007	0.824±0.011
Model 3	1.262±0.027	0.910±0.006	0.846±0.008

Table 5.1: Performance metrics calculated on all drives included in the test set for the three models, including the 99% confidence interval.

In figs. 5.1 to 5.3 the predicted temperatures are compared with the ground truth temperatures for model 1, model 2, and model 3 respectively.

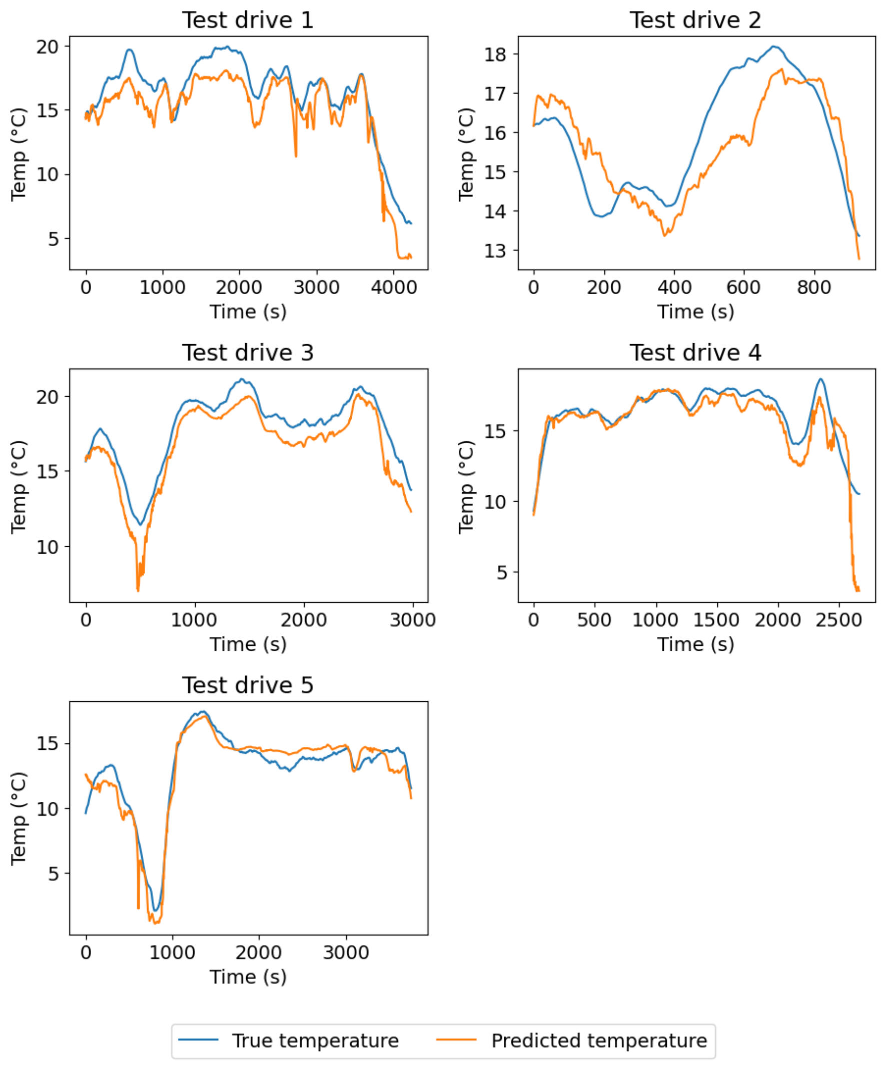


Figure 5.1: Predicted right front tyre temperatures (orange) for each drive included in the test data using model 1, and actual tyre temperature (blue).

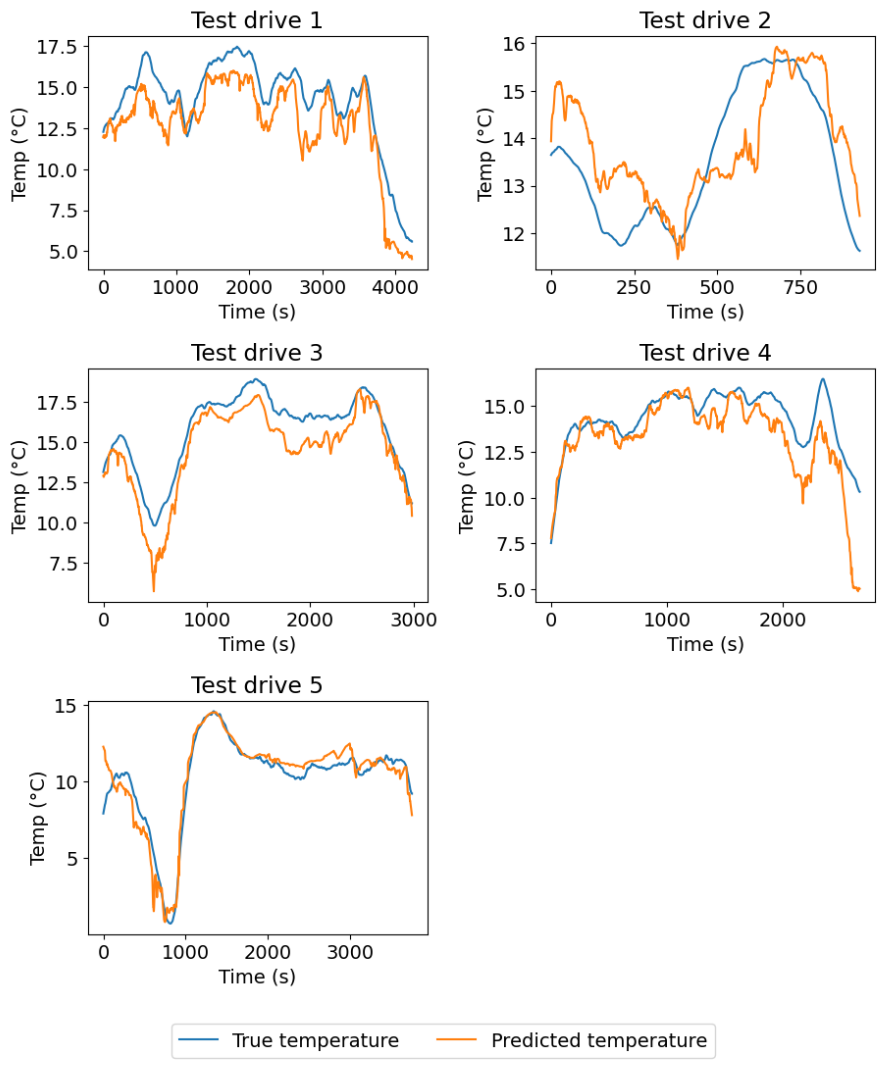


Figure 5.2: Predicted left rear tyre temperatures (orange) for each drive included in the test data using model 2, and actual tyre temperature (blue).

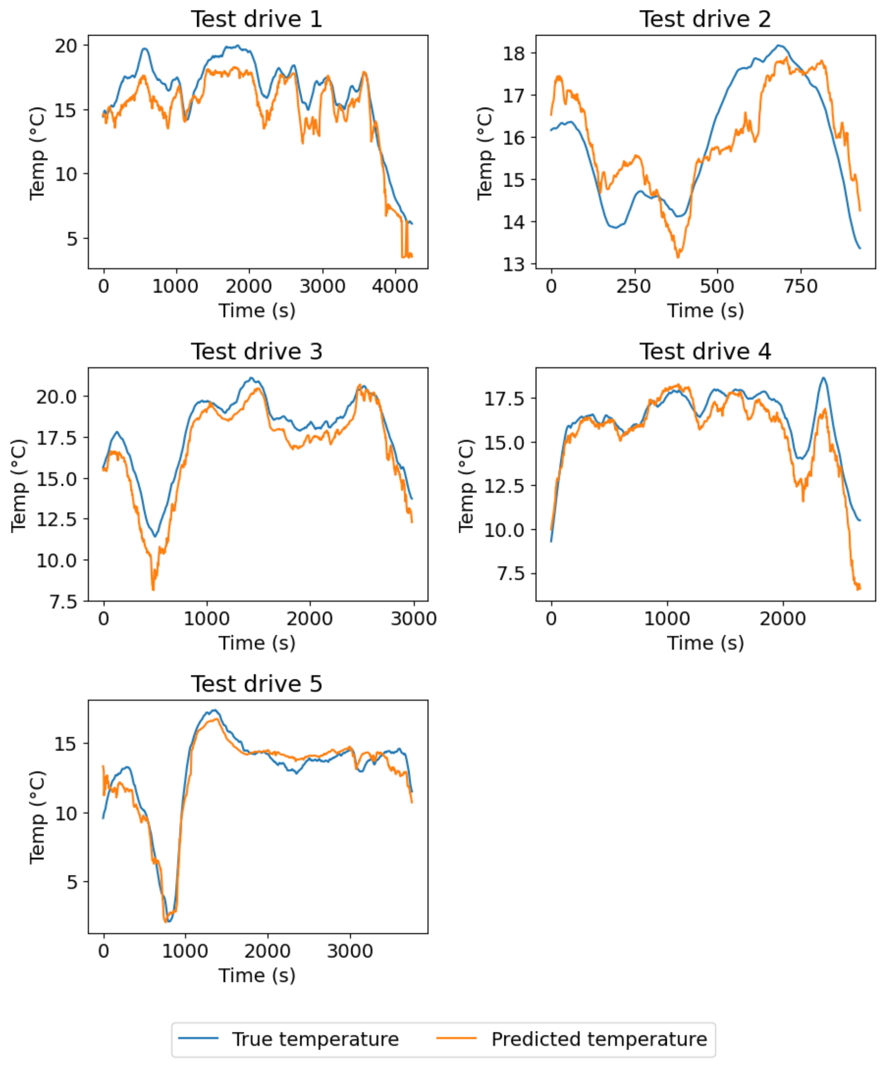


Figure 5.3: Predicted right front tyre temperatures (orange) for each drive included in the test data using model 3, and actual tyre temperature (blue).

6 Discussion

6.1 Evaluating performance

As previously stated, RMSE is the standard deviation of the residuals. Since this performance metric is dependent on the magnitude of the target signal, it is only possible to compare the RMSE between model 1 and model 3 (see fig. 4.1). So, when comparing these two it is evident that the RMSE for model 3 is lower, and the 99% confidence interval is nonoverlapping.

Contrary to RMSE, it is easier to compare different models based on their R-squared values since it is scale-invariant. The measure is commonly interpreted as the degree to which the regression model accurately represents the observed data. As seen in Table 5.1, the R-squared was around 0.82-0.85 for all models, where model 3 had the highest value. Therefore, one can say that the models are able to explain around $82\% - 85\%$ of the tyre temperature variability. However, it is unclear whether this could have been improved or if limitations exist due to the data, model design, or a potential inherent uncertainty involved in the physics of tyre temperature modeling.

Also, it can be seen from the results in Table 5.1 that the overall accuracy for all different models were around 90%. Thus, it is evident that our model falls a bit short of our desired target accuracy (100%).

However, the comparisons in figs. 5.1 to 5.3 indicates that the performance varies a bit depending on the specific drive. On the data from test drive 4 and 5 the predictions are very close to the true values. However, on test drive 1 and 2 the performance is relatively poor. On drive 1, it seems like the predictions consistently underestimates the true tyre temperature, whereas on drive 2 the predictions have a considerably higher variability compared to the true temperatures. It was tried to see if these behaviours could be explained by the drives metadata (see Table 3.1), however no clear relationship could be drawn.

6.2 Different tyres

The focuse lied on tuning the LSTM model's hyperparameters and network structure specifically for predicting right front tyre temperatures. Despite this, after retraining our model was able to predict the left rear tyre temperatures equally good. This can be seen by comparing metrics presented in Table 5.1, where many of the performance metrics fall within each-others 99% confidence intervals. This result suggests that the model, after re-training, would be capable of performing equally well on the remaining two tyres.

6.3 Feature importance

Apart from the accuracy goal, our thesis aimed to analyze the importance of different on-board signals for prediction tyre temperature. According to our model, the five most important signals are vehicle speed, ambient temperature, brake pedal position, acc. pedal position, and road inclination in that order (see fig. 4.6). One can also see that the importance does not decrease linearly, rather there are a few very important features and the rest do only marginally contribute to the predictions. By comparing results in Table 5.1, the values indicates that the performance increases after discarding insignificant signals.

The exact reason for why removing signals improves performance is not clear, however there are a few possible explanations. One reason may be that when all features are included the model overfits to the training data. By removing the least important ones the model complexity is reduced and might increase its ability to generalize.

Another possible explanation is that by reducing the least important features, multicollinearity is reduced. In Section 3.2.2, it was presented that even after dropping signals, some pairs still had a very high correlation coefficient. By removing one of them, the redundancy in the data is reduced. This reduction in redundancy might enhance the model's ability to capture independent and meaningful relationships between the remaining features, without disturbance from non-important highly correlated inputs.

Also, when reducing the dimensionality of the dataset, the sparsity in the data reduces since less volume is needed to cover an adequate part of the feature space. This phenomenon is called *Curse of dimensionality*, which often leads to over-fitting and lack of generality. Since our dataset is relatively limited, reducing the number of features might have mitigated this potential effect.

6.4 Weather & Road condition

Earlier the potential impact of a wet or dry asphalt on tyre temperatures was discussed. No conclusions could be drawn regarding the weather and/or state of the road, most likely due to the lack of sufficient data representing different conditions. As seen in Table 3.1, there are only four drives with sunny weather, three drives with rainy weather, and four drives with wet asphalt. Too analyze how these parameters impact the model, more data is needed for each condition.

If the data collection could be redone, one way of increasing the amount of data for different conditions could be to remove the option to classify the pavement under a drive as *Mixed* and just use *Dry* or *Wet*. Additionally, the driver should be allowed to change the condition under driving. Theoretically, the road can be very wet in the beginning of a drive and completely dry at the end. In light of this, if a drive is classified as mixed and the state of the road significantly impacts tyre temperature, the model will struggle to capture this effect. However, if the mixed condition is removed, one also needs to define when the transition from wet to dry occurs (or vice versa), which can be hard to keep consistent. How to retrieve the weather and state of the road live in the car is also an obstacle to be solved.

6.5 Dataset split

Since the validation and test data split was based on the different weather and road conditions and not the actual driving dynamics, there might be a bias in the model. If, for example, the majority of the validation data consists of highway driving and the test data of city driving, the model will most likely perform much better on the validation data than the test data. It can also be so that the train data and test data are more similar to each-other compared to train and validation. It is not presented in this work, but indications of this behaviour have been observed since the validation loss have sometimes been higher than the test loss. To reduce this bias in future research, one should characterize and split drives by also looking at actual driving dynamics, rather than solely on weather and road conditions.

6.6 Choice of hyperparameters

As discussed in Section 2.8 the choice of hyperparameters have a big impact on a model's performance, and in this thesis only a small portion of the hyperparameter space was explored. For example, if one wants to test three different values for all hyperparameters in Table 4.1 (except epochs), the total number of unique combinations are $3^7 = 2187$. However, the goal of this work was not to create the optimal model so searching extensively through the hyperparameter was deemed unnecessary.

With that said, one could have chosen to test other hyperparameters. Additional layers and/or more units, a smaller batch size, and a different learning rate might have improved the performance of the model. Other choices of optimizers or loss functions might also have had an impact of the performance. Adam and MSE were used due to them being used in similar problems [7, 8, 9] and they are simple to implement, but again there are plenty more options to test. Therefore, the remaining hyperparameter space is left to be explored by future research.

6.6.1 Sequence length

Our analysis showed that for the baseline GRU model, 600 seconds was actually marginally better than 500 seconds which was used. But as discussed in Section 4.4, it was opted to use 500 seconds to reduce history required to predict. For the LSTM model, 500 seconds was best and the validation loss actually increased for longer sequence lengths.

The drawback of the model design is that to make predictions, 500 seconds of history is required. It was discussed in Section 4.4 that even though a longer sequence might result in better results, it is problematic due to the increase of waiting time before predictions can be made. This will limit the model's real-time applicability and hinder its usefulness in short drives scenarios. Thus if one wants to predict the tyre temperatures in such scenarios, one needs to make a simpler assumption on how the tyre temperature behaves for the first 500 seconds, or another approach is required.

6.7 Future work

Apart from exploring the remaining hyperparameter space as previously mentioned, it would be valuable to explore how the LSTM model performs with an extended dataset that encompasses a wider range of weather and road conditions. This enhanced dataset could potentially help provide insight into how the model's performance is affected by different environmental factors. Also, data from other cars, tyres, and loads would be interesting to incorporate as well. By doing so, one can start to work towards developing a more robust tyre temperature prediction model that can accurately forecast tyre temperatures across a broader spectrum of driving scenarios, conditions, and cars.

Also, further research could explore the impact of varying sequence lengths on the model's performance in more detail, as well as investigate alternative methods for capturing the necessary historical information more efficiently. This could include techniques that feed in the initial ground truth tyre temperature at the start and apply a online learning approach, with the ambition of removing the necessity historical data. However, if such an approach was to be analyzed, one needs to implement a method of prediction the initial tyre temperature when the car is turned off and standing still. This problem might prove to not be as complex, but still one additional obstacle is introduced.

Additional research could also encompass the possible effects of other features that are not included in the onboard-signals analyzed in this work. Such signals could be the type of road surface (e.g. asphalt, concrete, or gravel), the sun intensity, and the temperature of the road. However, some of these signals may be hard to capture without manual input. Therefore further research is suggested within these areas to see if it is possible to, for example, automatically detect the type of road surface.

7 Conclusion

The aim of this thesis was to explore the possibility of constructing a virtual sensor using on-board signals and explore the importance of these. To solve this, different LSTM and GRU models were tried, where an LSTM model with one layer of 256 units, dropout of 0.3, and a sequence length of 500 seconds performed the best. Even-though it was shown that creating such a sensor is possible, the overall performance falls a bit short of our desired target. Despite this, valuable information about feature importance could be extracted. It was shown that vehicle speed, ambient temperature, brake pedal position, acc. pedal position, and road inclination was the five most important inputs to the model. At the end, it was discussed how the model performance can be improved by suggesting different interesting future research areas, e.g., by including weather and road surface information.

Bibliography

[1] *Passenger car, truck and bus tyre rolling resistance measurement method — Single point test and correlation of measurement results*. Standard. International Organization for Standardization, Geneva, CH, 2018. URL: https://www.iso.org/standard/67531.html.

[2] D. J. Schuring, J. F. Siegfried, and G. L. Hall. “Transient Speed and Temperature Effects on Rolling Loss of Passenger Car Tires”. In: *SAE Technical Paper Series*. SAE International, 1985. DOI: 10.4271/850463.

[3] T. Wangs. *Analysis on Tyre Wear: Modelling and Simulations*. MA thesis 2017:96. KTH, Vehicle Dynamics, 2017, p. 76. URL: http://kth.diva-portal.org/smash/get/diva2:1416550/FULLTEXT01.pdf.

[4] J. Hyttinen, M. Ussner, R. Österlöf, J. Jerrelind, and L. Drugge. “Truck tyre transient rolling resistance and temperature at varying vehicle velocities - measurements and simulations”. *Polymer Testing* (2023), p. 108004. ISSN: 0142-9418. DOI: 10.1016/j.polymertesting.2023.108004. URL: https://www.sciencedirect.com/science/article/pii/S0142941823000843.

[5] J. Ejsmont, S. Taryma, G. Ronowski, and B. Swieczko-Zurek. “Influence of temperature on the tyre rolling resistance”. *International Journal of Automotive Technology* **19**:1 (2017), pp. 45–54. DOI: 10.1007/s12239-018-0005-4.

[6] A. Jakobsson. *An introduction to time series modeling*. Third. Studentlitteratur AB, 2021. ISBN: 9789144134031.

[7] J. Alexandersson and E. S. Lönegren. *Neural Networks for modelling of a virtual sensor in an engine*. MA thesis. Chalmers University of Technology, 2019. URL: https://publications.lib.chalmers.se/records/fulltext/257203/257203.pdf.

[8] W. Ke, D. Huang, F. Yang, and Y. Jiang. "Soft sensor development and applications based on LSTM in deep neural networks". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2017, pp. 1–6. DOI: 10.1109/SSCI.2017.8280954.

[9] M. Guesbaya, F. Garcia-Manas, F. Rodriguez, and H. Megherbi. "A soft sensor to estimate the opening of greenhouse vents based on an lstm-rnn neural network". *Sensors* **23** (2023), p. 1250. DOI: 10.3390/s23031250.

[10] I. N. da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves. *Artificial Neural Networks*. Springer, 2017. ISBN: 9783319431611.

[11] A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*. Second. O'Reilly Media, Incorporated, Sebastopol, UNITED STATES, 2019. URL: http://ebookcentral.proquest.com/lib/lund/detail.action?docID=5892320.

[12] C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-94463-0.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[14] D. P. Kingma and J. Ba. *Adam: a method for stochastic optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[15] D. Masters and C. Luschi. "Revisiting small batch training for deep neural networks" (2018). DOI: 10.48550/ARXIV.1804.07612. arXiv: 1804.07612 [cs.LG].

[16] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. "Accurate, large minibatch sgd: training imagenet in 1 hour" (2017). DOI: 10.48550/ARXIV.1706.02677. arXiv: 1706.02677 [cs.CV].

[17] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. *Empirical evaluation of gated recurrent neural networks on sequence modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[18] S. Hochreiter and J. Schmidhuber. "Long short-term memory". *Neural Computation* **9**:8 (1997), pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". *Journal of Machine Learning Research* **15** (2014), pp. 1929–1958. ISSN: 15324435. URL: `https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edselc&AN=edselc.2-52.0-84904163933&site=eds-live&scope=site`.

[20] S. Lundberg and S.-I. Lee. *A unified approach to interpreting model predictions*. 2017. arXiv: `1705.07874 [cs.AI]`.

[21] M. Sundararajan, A. Taly, and Q. Yan. *Axiomatic attribution for deep networks*. 2017. arXiv: `1703.01365 [cs.LG]`.

[22] Izze-Racing LLC. *Tire temperature & pressure monitoring system*. Accessed 2023, May 15. URL: `https://www.izzeracing.com/products/tire-temperature-pressure-monitoring-system.html`.

[23] L. Ydrefors, A. Stensson Trigell, and J. Jerrelind. *The relationship between rolling resistance and tyre operating conditions, with a focus on tyre temperature*. eng. KTH, Farkostteknik och Solidmekanik, 2022. URL: `http://urn.kb.se/resolve?urn=urn:nbn:se:vti:diva-18776`.

[24] I. Guyon and A. Elisseeff. "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research* **3**:7/8 (2003), pp. 1157–1182. ISSN: 15324435. URL: `https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=bth&AN=11939737&site=eds-live&scope=site`.

[25] S. Butterworth. "On the Theory of Filter Amplifiers". *Experimental Wireless & the Wireless Engineer* **7** (1930), pp. 536–541.

[26] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole, Cengage Learning, 2012. ISBN: 9780538733526.