Linköping University | Department of Vehicular Systems

Master's thesis, 30 ECTS | Elektroteknik

2022 | LITH-ISY-EX– 22/5494-SE

Artificial Neural Network in Exhaust Temperature Modelling

– Viability of ANN Usage in Gasoline Engine Modelling

Linus Roos
Nibras Musa

Supervisor : Olov Holmer
Examiner : Jan Åslund

External supervisor : Ali Ghanaati

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

Developing and improving upon a good empirical model for an engine can be time-consuming and costly. The goal of this thesis has been to evaluate data-driven modelling, specifically neural networks, to see how well it can handle training for some static models like the mass flow of air into the cylinder, mean effective pressure and pump mean effective pressure but also for transient modelling, specifically the exhaust gas temperature. These models are evaluated against the classical empirical models to see if neural networks are a viable modelling option. This is done with five different types of neural networks which are trained. These are the feed-forward neural network, Nonlinear autoregressive exogenous model network, layer recurrent network, long short term memory network and gated recurrent network. The inputs were determined by looking at more simple physical models but also looking at the covariance to determine the usefulness of the input. If the calculation time is small for the specific network, the neural network structure is tested and optimized by training many networks and finding the median/mean result for that specific test. The result has shown that the static models are handled very well by the most simple feed-forward network. For the exhaust temperature, both NARX and Layer recurrent network could predict and handle it well giving results very close to the empirical models and could be a viable option for transient modelling, on the other hand, Long short term memory, gated recurrent network and the feed-forward network had trouble predicting the exhaust gas temperature and returned bad results while training.

Acknowledgments

We want to extend our gratitude to those who helped make this thesis possible. First on this list are our supervisors: Ali Ghanaati at Aurobay, who help with providing knowledge and experience of both company-related tool and procedures as well as data-driven methods, and Olov Holmer at Linköping University, who provided ideas and strategies that helped us improve on the work done in this thesis.

We also want to thank the engineers at Aurobay who helped in different ways throughout the project. In particular we wish to thank Bohan Liang for playing a large role both the organisation and execution of the thesis work, as well as Venkatraman Nagaraj who spent a considerable amount of time assisting in the ECU compilation of our neural networks.

Contents

Abstract iii

Acknowledgments iv

Contents v

List of Figures vii

List of Tables x

1 Introduction 4
- 1.1 Motivation . . . 4
- 1.2 Aim . . . 5
- 1.3 Research questions . . . 5
- 1.4 Literature Review . . . 5
- 1.5 Delimitations . . . 6

2 Engine Modelling 7
- 2.1 Mean Value Engine Models . . . 7
- 2.2 Cylinder Air Flow . . . 7
- 2.3 Engine Power and Efficiency . . . 8
- 2.4 Heat Transfer . . . 10

3 Artificial Neural Networks 11
- 3.1 Artificial Neural Networks . . . 11

4 Available Data 20

5 Method 22
- 5.1 Choice of programming tools . . . 23
- 5.2 Pre-processing and Training . . . 23
- 5.3 Network training Process . . . 26

6 Results 30
- 6.1 Cylinder Flow: FNN . . . 31
- 6.2 PMEP: FNN . . . 36
- 6.3 $IMEP_g$: FNN . . . 39
- 6.4 Exhaust Gas Temperature Model: FNN . . . 43
- 6.5 Exhaust Gas Temperature Model: NARX . . . 47
- 6.6 Exhaust Gas Temperature Model: LSTM . . . 64
- 6.7 Exhaust Gas Temperature Model: GRU . . . 65
- 6.8 Exhaust Gas Temperature Model: LRN . . . 66
- 6.9 Summation of Results . . . 73

7 Discussion **75**
7.1 Results 75
7.2 Method 78
7.3 Future Work 78
7.4 The work in a wider context 79

8 Conclusion **80**

Whole bibliography **84**

List of Figures

2.1 The air does not flow freely from the intake manifold through the cylinder but is controlled by the inlet valve and exhaust valve. When these valves open and close affects the efficiency of the engine. . . . 8
2.2 A pressure to volume diagram of the engine cycle. . . . 9

3.1 Sigmoid neuron . . . 12
3.2 A Feed forward neural network with two hidden layers. . . . 12
3.3 An example of a recurrent network. Feedback is introduced between the layers, enabling the network to better capture the behaviour of dynamic systems. . . . 13
3.4 A NARX network with one hidden layer, weights W, activation function $g(x)$ and internal delay of 5 samples. . . . 14
3.5 A layer recurrent network with two hidden layers, weights W, activation function $g(x)$ and internal delay 5. . . . 14
3.6 Here the input gate is the forget gate . . . 15
3.7 Long short term memory architecture . . . 16

4.1 Vehicle speed for each data set. . . . 21
4.2 Engine load to Engine Speed for the steady-state data sets . . . 21

5.1 Black-Box Exhaust Temperature Model . . . 22
5.2 [XXX] representing a vector with the inputs in a timestep and T_{eo} the output at the same timestep . . . 25
5.3 When using the neural network for prediction the output is used as feedback to generate time-series data. . . . 28

6.1 Cylinder flow FFNN performance dependence on number of neurons. . . . 31
6.2 The performance of different networks. The number of neurons in the first hidden layer is six and the number of neurons in the second hidden layer changes as seen in the figure. . . . 32
6.3 The performance of different activation functions. . . . 32
6.4 Validation of the cylinder flow model on a part-load data set . . . 33
6.5 Cylinder flow model on the data-set with cold start. . . . 34
6.6 With the engine on/off input, the model correctly predicts no flow when the engine is off. . . . 34
6.7 Without the engine on/off input, the model incorrectly predicts negative values when the engine is off. . . . 35
6.8 The performance of a trained network depending on the number of neurons. . . . 36
6.9 The performance of different networks. The number of neurons in the first hidden layer is six and the number of neurons in the second hidden layer changes as seen in the figure. . . . 36
6.10 The performance of different activation functions in a one hidden layer network. . 37
6.11 Validation of the PMEP model on a part-load data set . . . 37
6.12 Validation on warm start transient data set. . . . 38

6.13 Validation on cold start transient data set. 38
6.14 The performance of different activation functions in a one hidden layer network. . 39
6.15 The performance of different networks. The number of neurons in the first hidden layer is six and the number of neurons in the second hidden layer changes as seen in the figure. 40
6.16 The performance of different activation functions in a one hidden layer network. . 40
6.17 Validation of the IMEP model on a part-load data set 41
6.18 Validation on warm start transient data set. 41
6.19 Validation on warm start transient data set. 42
6.20 Validation on cold start transient data set. 42
6.21 The performance of a trained network depending on the number of neurons. . . . 43
6.22 The performance of different networks. The number of neurons in the first hidden layer is four and the number of neurons in the second hidden layer changes as seen in the figure. 43
6.23 The performance of different activation functions in a one hidden layer network. . 44
6.24 Validation of the exhaust gas temperature model on a part-load data set 44
6.25 Validation on warm start transient data set. 45
6.26 Validation on warm start transient data set. Prediction is low-pass filtered 45
6.27 Activation function test with median value, training data 47
6.28 Activation function test with median value, validation data Kiruna 2 48
6.29 Activation function test with median value, validation data Gibraltar 2 48
6.30 Training data NRMSE, 95% confidence interval 49
6.31 Gibraltar 6, validation results 95% confidence interval 50
6.32 Kiruna 2, validation results 95% confidence interval 50
6.33 Test on number of neurons with two hidden layers, training data 51
6.34 Test on number of neurons with two hidden layers, validation data Kiruna 2 51
6.35 Test on number of neurons with two hidden layers, validation data Gibraltar 6 . . 52
6.36 Network performance as a function of early stopping ratio. Trained on Gibraltar 3 53
6.37 Network performance as a function of early stopping ratio. Trained on Gibraltar 3 and evaluated on Gibraltar 6 53
6.38 Autocorrelation to 150 sample delays 54
6.39 Training result of Kiruna two 54
6.40 Validation on Kiruna two data set, orange: ANN, blue:original 55
6.41 Training data on delay test 55
6.42 Kiruna 2 validation data on delay test 56
6.43 Gibraltar 6 validation data on delay test 56
6.44 cross-covariance between the inputs and the exhaust temperature up to 2 min sample delay 57
6.45 cross-covariance between the inputs and the exhaust temperature up to 2 min sample delay 57
6.46 Training data input delay test 58
6.47 Validation data Kiruna 2 58
6.48 Validation data Gibraltar 6 59
6.49 Validation on Gibraltar 6 data set, orange: ANN, blue:original 60
6.50 Validation on Kiruna 2 data set, orange: ANN, blue:original 61
6.51 NARX model validated against steady-state data. The green line indicates the size of the error for each measurement. 62
6.52 Validation on Kiruna 2 data set using ANN-models for cylinder flow, $IMEP_g$ and PMEP inputs. 63
6.53 Validation on Cold 2 data set, orange: ANN, blue:original 64
6.54 Validation on Cold two data set, orange: ANN, blue:original 65
6.55 Median from training data 66
6.56 Median Result with the validation data set Kiruna 2 66

6.57 MedianResult with the validation data set Gibraltar 6 67
6.58 Result with training data 68
6.59 Result with the validation data Kiruna 2 68
6.60 Result with the validation data Gibraltar 6 69
6.61 Feedback test with training data 70
6.62 Feedback test with validation data Kiruna 2 70
6.63 Feedback test with validation data Gibraltar 6 71
6.64 Validation on Kiruna 2 data set, orange: ANN, blue:original 72
6.65 Results of the mass flow into the engine, in real-time 74
6.66 Results of the exhaust gas temperature, in real-time 74

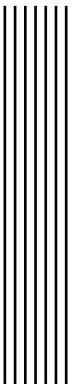
List of Tables

3.1 Long short term variable description 15
3.2 Gated recurrent unit, variable description 16
3.3 Some of the most commonly used activation functions. 17

4.1 Data sets for transient data 20

5.1 Normalisation Ranges 23
5.2 Input used for the static neural network models 26
5.3 Input used for the temperature neural network models 26

6.1 Performance of the cylinder flow model on different transient data sets. 33
6.2 Performance of the filtered and unfiltered feed-forward exhaust gas temperature network 46
6.3 Narx performance on validation data against Aurobays physical model with no cold start. 60
6.4 Narx performance on training data against Aurobays physical model with no cold start. 60
6.5 Narx performance on validation data against Aurobays physical model. 61
6.6 Narx performance on training data against Aurobays physical model. 61
6.7 Performance of a full ANN simulation of the exhaust gas temperature. Data from table **??** is repeated here for readability. 63
6.8 LSTM performance on validation data against Aurobays physical model. 64
6.9 LSTM performance on training data against Aurobays physical model. 64
6.10 GRU performance on validation data against Aurobays physical model. 65
6.11 GRU performance on training data against Aurobays physical model. 65
6.12 Layer Recurrent performance on validation data against Aurobays physical model. 72
6.13 Layer Recurrent performance on training data against Aurobays physical model. . 72
6.14 Summation of model performance on validation data. 73



Notations

Artificial Neural Networks

Nomenclature	**Description**
t	Time index
$y(t)$	System output
$\hat{y}(t)$	Estimated system output
$x(t)$	System input
$\sigma(z)$	Sigmoid function
$\hat{w}$	neuron weights
b	neuron bias

Engine Modelling

Nomenclatur	**Description**
$W_{i,g}$	Gross indicated work $[kgm^2/s^2]$
$W_{i,p}$	pumping work losses $[kgm^2/s^2]$
$W_{i,n}$	Net indicated work $[kgm^2/s^2]$
$PMEP$	Pump mean effective pressure $[Pa]$
$IMEP_g$	Indicated mean effective pressure $[Pa]$
$\dot{m}_{ac}$	Mass flow of air through the cylinders $[kg/s]$
$\dot{m}_f$	Mass flow of fuel into cylinder $[kg/s]$
$\dot{m}_{em}$	Mass flow of in exhaust manifold $[kg/s]$
η_i	Heat efficiency [-]
η_λ	fuel energy release coefficient
V_D	Engine Displacement $[m^3]$
N	Engine Speed [RPM]
n_r	Number of crank revolutions in a cycle [-]
p_{im}	Pressure at intake manifold [Pa]
p_{eo}	Pressure at cylinder exit [Pa]
T_{im}	Temperature at intake manifold [K]
T_{eo}	Temperature at cylinder exit [K]
T_{em}	Temperature at exhaust manifold [K]
T_{amb}	Ambient temperature [K]
η_{vol}	Volumetric efficiency[-]
η_{vvt}	Efficiency of VVT [-]
R	Specific Gas Constant $[Jkg^{-1}K^{-1}]$
θ_{ign}	Ignition angle [rad]
$\theta_{ign,opt}$	Optimal ignition angle [rad]
C_i	Ignition efficiency polynomial [-]
λ_{cyl}	Air to fuel equivalence ratio in the cylinder[-]]
$\dot{Q}_{ht}$	Heat transfer from gas in pipe to surrounding $[kgm^2/s^3]$
c_p	Specific heat capacity $[J/kgK]$
q_{lhv}	Lower heating value $[MJ/Kg]$
r_c	Compression ratio [-]
γ	ratio of specific heats [-]

Abbreviations

Abbreviation	Description
ANN	Artificial Neural Network
VVT	Variable Valve Timing
NARX	Nonlinear auto-regressive network with external inputs
FNN	feedforward neural network
LSTM	Long short term memory

1 Introduction

1.1 Motivation

Knowledge of the internal engine-states are useful for many different things, such as diagnostics, control and development. The internal states of the engine can be predicted in three ways:

- Measurements
- Physical modelling
- Data-driven modelling

Measuring all different physical properties of the engine is expensive and can not affordably be done in commercially available passenger cars. Traditionally, the car industry has relied mainly on physical modelling due to its predictability, but also on data-driven modelling to solve more complex problems. These data-driven models can, for example, be curve-fitting, mapping the variables or by using machine learning.

The car industry has existed for some time and has developed complex and well-tuned engine models of both the physical and data-driven types. These models are expensive to make, however, as they take a lot of development time. As new components are developed in order to increase engine performance and comply with environmental laws, these models will have to be expanded.

The engine developer and manufacturer Aurobay want to look into the possibilities of using data-driven model development to speed up the process of creating new models. Specifically if using artificial neural networks can be used to capture complex, dynamic and non-linear behaviour where, currently, curve-fitting and map-based approaches are commonly used.

The temperature of the exhaust gas has a complicated behaviour as it is not only affected by the combustion in the engine but also by the cooling of the engine components as heat is transferred to the ambient air. It is therefore a good candidate for a sub-component in an

engine model to evaluate the performance of artificial neural networks as a modelling tool. If the neural network does predict the exhaust temperature with very good results this could in turn prove that a data-driven approach is a viable option for replacing or complementing physical modelling. The method could then potentially lower development time and cost as you could simply put a sensor for a needed variable, then use a data-driven method to extract a model, and later remove that sensor if the performance of the data-driven model is good.

1.2 Aim

The underlying purpose of this thesis is to work as a proof of concept for the use of artificial neural networks as a way to decrease model development time. This will be done by attempting to achieve a model of the exhaust gas temperature. The exhaust gas temperature has been chosen partly because of the non-linear and dynamic aspects of the system.

1.3 Research questions

The question of checking the viability of using artificial neural networks for the purpose of engine modelling, specifically for the exhaust gas temperature and its component can be divided into four research questions:

- Is artificial neural network modelling a viable method for estimating the exhaust gas temperature?
- How does the performance of the artificial neural network model compare to existing empirical models?
- How can the structure of a neural network model of exhaust gas temperature be optimised with respect to the results and computational effort?
- Which network structure is best suited for modelling the exhaust gas temperature?

1.4 Literature Review

A state-of-the-art review done by Bhatt and Shrivastava shows great potential in using artificial neural networks to predict complex engine performance [1]. Some of the areas where research on artificial neural networks has been carried out are engine performance, heat and temperature problems, exhaust emission composition, diagnostics and maintenance. The authors noted that the most common training method used is the Levenberg–Marquardt method. The networks employed to solve the problems investigated usually contain between 10 and 20 neurons per hidden layer and use either logarithmic sigmoid or tangent sigmoid activation functions. Researchers at West Virginia University have managed to use different types of machine learning principles to accurately estimate the exhaust gas temperature [2]. This was however done on a heavy-duty natural gas spark ignition engine converted from a diesel engine. Their conclusion was that the most appropriate machine learning principle was, if well trained, an artificial neural network.

A study using artificial neural networks as soft sensors for predicting exhaust emissions in gasoline engines came to the conclusion that predicting different properties of the exhaust emission using artificial neural networks is possible. The network used a series of local linear models. The authors claimed however that it was difficult to capture the transient behaviour of the temperature using their steady-state model [3].

A different study that attempted to model a heavy-duty diesel engine using layered artificial neural networks also succeeded by using mainly static feed-forward networks. They solved

the problem of modelling the more complex dynamic behaviour by using a non-linear autoregressive network with exogenous inputs (NARX) [4].

The conclusion from the work previously done in this area is that artificial neural networks are well suited for engine modelling. Relying only on shallow feed-forward structures appears to be sufficient for simpler systems, such as several of the components used for calculating the exhaust gas temperature. For more dynamic systems, such as the exhaust temperature itself, a different approach needs to be used. It seems like previous researchers have had some success using a non-linear auto-regressive network with exogenous inputs for similar dynamic systems.

1.5 Delimitations

The delimitations for this project will be defined as followed.

- Only one engine model will be considered.
- Only the exhaust temperature model should be seen as a dynamic model and its components as static models.
- The model will not separate sensor dynamics from the temperature measurements.

2 Engine Modelling

The engine used for the project is a four-cylinder Volvo Engine Petrol 4 (VEP4) with a displacement of 2.0 litres.

2.1 Mean Value Engine Models

The engine models are derived from [5] the assumption of some of the models being mean value engine models. This means that the different models are assumed either as, static, dynamic or as a constant depending on how fast the model values change during engine cycles. An example is assuming a model is expressed as constant if its change is slower than 1000 engine cycles, static if it changes during 1 cycle and dynamic if it changes between $3 - 1000$ cycles.

2.2 Cylinder Air Flow

In order to examine the exhaust gas temperature it is necessary to calculate the air flow through the engine. The total air flow $\dot{m}_{ac}$ for all cylinders at steady-state can be estimated using a volumetric efficiency η_{vol}:

$$\dot{m}_{ac}(N, p_{im}, T_{im}, \dots) = \eta_{vol}(N, p_{im}, ...)\frac{V_D N p_{im}}{n_r R T_{im}} \tag{2.1}$$

Where N is the engine speed, V_D is the engine displacement, p_{im} and T_{im} is the pressure and temperature in the intake manifold, $n_r = 2$ for a four-stroke, four-cylinder engine and the specific gas constant $R = 286$ for air. The term $\frac{p_{im}}{RT_{im}}$ is the density of the air, assuming that the ideal gas law holds for these conditions.

2.2.1 Estimation of Volumetric Efficiency

The volumetric efficiency is complex and depends on many things such as intake manifold pressure, exhaust pressure, fuel, heat transfer, engine speed and exhaust gas re-circulation

[5]. In many cases, the volumetric efficiency has to be measured empirically. There are several common models for this. A simple relation for η_{vol} from [5]

$$\eta_{vol}(N, p_{im}) = c_0 + c_1\sqrt{p_{im}} + c_2\sqrt{N} \tag{2.2}$$

It can be noted that the above model is a parameterised non-linear data-driven model. This makes the cylinder air flow model a grey-box model.

2.2.2 Variable Valve Timing

The gas flowing through the cylinder is controlled by the exhaust and inlet valves as in Figure2.1. These valves are traditionally controlled mechanically by the camshaft.

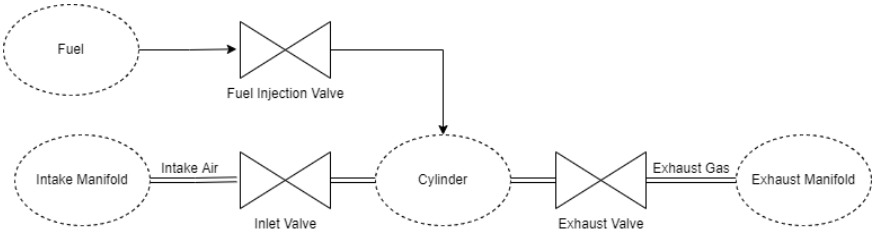


Figure 2.1: The air does not flow freely from the intake manifold through the cylinder but is controlled by the inlet valve and exhaust valve. When these valves open and close affects the efficiency of the engine.

If the exhaust valve is opened too early the pressure in the cylinder will drop, thus reducing the engine efficiency [5]. If the exhaust valve is opened too late the pressure will not decrease enough to efficiently pump new air into the cylinder. Having the valve openings overlap will affect the amount of residual exhaust gases in the cylinders. The optimal timings for opening the valves are different for different operating points. The exhaust valve for example should have an earlier opening when at high speed or high load.

In order to optimise air intake over a range of engine speeds and engine loads variable valve timing (VVT) can be used in the engine. This means that the opening and closing of the inlet and exhaust valves can be offset by a certain angle decided by the engine management system, both with respect to each other and to the combustion cycle. A simple way to model VVT is to simply add an efficiency η_{vvt} to the cylinder flow model. The valve opening and closing events need to be taken into consideration. Using only the valve overlap to model too variable valve timing is not enough [6].

$$\dot{m}_{ac} = \eta_{vvt}\eta_{vol}\frac{V_D N p_{im}}{n_r R T_{im}} \tag{2.3}$$

2.3 Engine Power and Efficiency

The gross work produced by the engine, as well as the pumping work consumed, can best be described by a graph of the cylinder pressure and volume during a combustion cycle. A typical combustion cycle can look like this in Figure2.2.

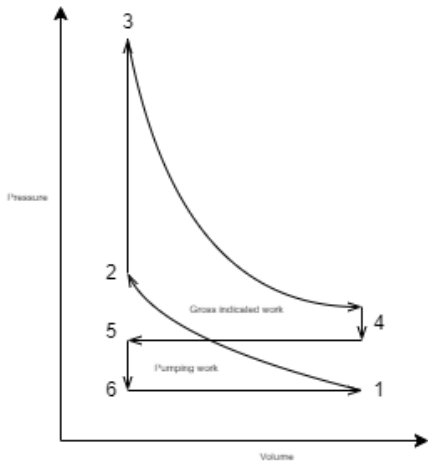


Figure 2.2: A pressure to volume diagram of the engine cycle.

2.3.1 Gross Indicated Engine Work

The gross indicated work produced $W_{i,g}$ by the engine is proportional to the amount of fuel m_f that is injected as well as q_{LHV}, how much energy per kilogram that the fuel contains.

$$W_{i,g} = m_f q_{LHV} \eta_{ig} \tag{2.4}$$

Where the efficiency η_{ig} of the process sometimes is written as a product of several efficiency loss sources.

$$\eta_{ig} = (1 - \frac{1}{r_c^{(\gamma-1)}}) \min(1, \lambda_{cyl}) \eta_{ign} \eta_{ig,ch} \tag{2.5}$$

Where:

- $\frac{1}{r_c^{(\gamma-1)}}$ are the losses for an ideal otto-cycle. r is the compression ratio and γ is the ratio of specific heats.
- $\min(1, \lambda_{cyl})$ limits the amount of energy that can be used when the gas is fuel-rich. λ_{cyl} is the air/fuel equivalence ratio in the cylinder.
- η_{ign} are the losses due to sub-optimal ignition timing.
- $\eta_{ig,ch}$ captures a range of engine-dependent losses.

The efficiency due to crank shaft ignition angle, η_{ign},is sometimes approximated as a polynomial of order n of the deviation $\theta_{ign} - \theta_{ign,opt}$ from the optimal ignition angle.

$$\eta_{ign}(N, m_f, \lambda_{cyl}, ...) = 1 - \sum_{i=2}^{n} C_i \left(\frac{\theta_{ign} - \theta_{ign,opt}(N, m_f, \lambda_{cyl}, ...)}{100} \right)^i \tag{2.6}$$

The gross indicated work can be described as a gross indicated mean effective pressure, IMEP_g.

$$\text{IMEP}_g = \frac{W_{i,g}}{V_d} \tag{2.7}$$

2.3.2 Net Indicated Work

Some of the work created by the engine is used up internally as pump work $W_{i,p}$, and does not contribute to the net indicated work $W_{i,net}$.

$$W_{i,net} = W_{i,g} - W_{i,p} \tag{2.8}$$

2.3.3 Pumping Work

A simple but common way to describe the pumping losses is as in Equation (2.9) [5]. This model is simplified and does not take pressure losses into account. It does also not take VVT behaviour into account.

$$W_{i,p} = V_d(p_{eo} - p_{im}) \tag{2.9}$$

The pumping work can also be expressed as a pumping mean effective pressure by normalising against the engine displacement.

$$\text{PMEP} = \frac{W_{i,p}}{V_d} \tag{2.10}$$

2.3.4 Grill shutter

A grill shutter controls the radiator inlet to adjust the airflow into the radiator which in turn regulates the coolant temperature.

2.4 Heat Transfer

2.4.1 Engine Out Temperature

The engine out temperature T_{eo} can be estimated by looking at the energy balance of an open system representation of the engine [5].

$$c_p(\dot{m}_{ac} + \dot{m}_f)T_{eo} + \dot{m}_f x_e h_{fg} + \dot{W}_{ig} + \dot{Q}_{ht} = \dot{m}_{ac} c_p T_{im} + \dot{m}_f c_p T_f + \dot{m}_f q_{LHV} \eta_\lambda \tag{2.11}$$

Where $\dot{Q}_{ht}$ is the heat transfer to the coolant system, T_f is the temperature of the fuel and $x_e h_{fg}$ is the energy required to evaporate a fraction, x_e, of the fuel.

This can be rewritten to estimate T_{eo}

$$T_{eo} = \frac{\dot{m}_{ac} c_p T_{im} + \dot{m}_f(c_p T_f + q_{LHV}\eta_\lambda - x_e h_{fg}) - \dot{W}_{ig} - \dot{Q}_{ht}}{c_p(\dot{m}_{ac} + \dot{m}_f)} \tag{2.12}$$

This method requires knowing the amount of heat that is transferred to the coolant system, which can be difficult to estimate.

2.4.2 Heat Transfer in Exhaust Pipe

The engine out temperature is the temperature as the exhaust gases leave the cylinder. There are however further losses in heat as the gas travels through the exhaust manifold.

3 Artificial Neural Networks

3.1 Artificial Neural Networks

An artificial neural network (ANN) has the ability to infer rules with a given data set. A simple example of this is recognising handwritten numbers. Defining an untrained neural network with defined handwritten numbers and training the network to recognize these patterns, gives the neural network an ability to recognise and distinguish the pattern and shapes of the different numbers. The concept behind artificial neural networks is, as the name implies, is to establish a network of artificial neurons, where input data is sent into this network, and the neurons applies their own properties, in our case mathematical functions and internal parameters to reach an ideal output. An illustration of such a neuron can be seen in3.1. This mathematical function is commonly referred to as the neurons activation function.

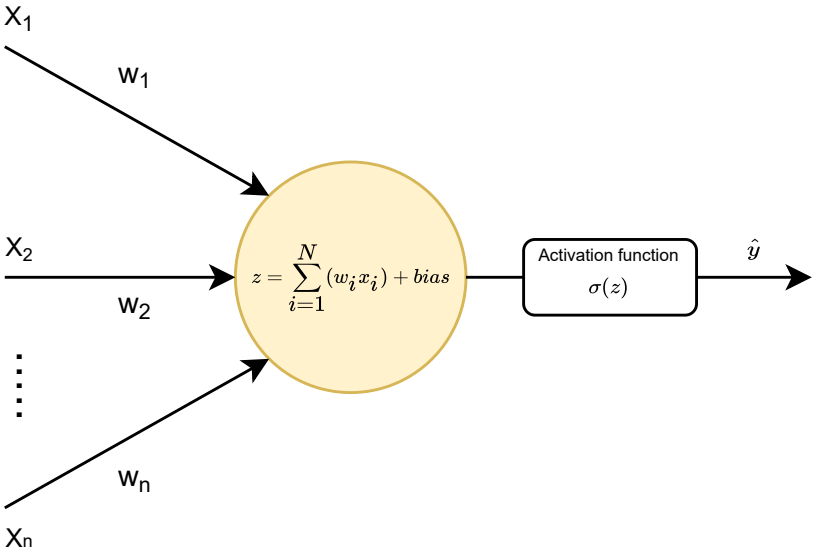


Figure 3.1: Sigmoid neuron

In this case the neuron output $\hat{y} = \sigma(z) = \sigma(\bar{w}^T \bar{x} + b)$, where $\sigma(z)$ is the Sigmoid activation function and $z = \bar{w}^T \bar{x} + b$. The internal parameters $\bar{w}$ and b are known as the weights and bias of the neuron [7].

In the illustration σ, the sigmoid function is used, but other activation functions can be used as well. A few different activation functions will be defined in this thesis, including the Sigmoid function. The network does not need to use the same activation function for every neuron.

3.1.1 Feed forward Neural Network

A simple way to explain neural networks is to look at a simple feed-forward neural network as illustrated in3.2. Here information is propagated from the input layer to the output layer, going through each of the hidden layers in order, where each of the layers consists of neurons similar to3.1.

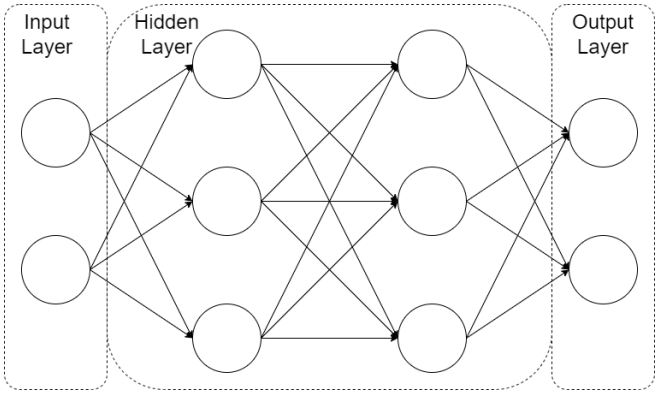


Figure 3.2: A Feed forward neural network with two hidden layers.

An important distinction to make is that a feed-forward neural network does not use any information on the time but is more focused on fitting data points if it is a simple non-linear input-output ANN and so the use of a feed-forward network in some cases, might not be optimal. This type of ANN is fine if the desired model is to be static, or if the model is to describe a dynamic system while at steady state. If dynamics are to be incorporated into a feed-forward neural network the input layer should include lagged input and output for a more optimal result [8].

3.1.2 Recurrent Neural Network

Before the other structures are looked at, recurrent neural networks need to be explained as that is what the more complex networks are defined as. In comparison to a Feed forward neural network, the recurrent network can have data travelling forward and backwards i.e. a feedback loop which can handle solving problems which the feed-forward network might not be optimal for [7].

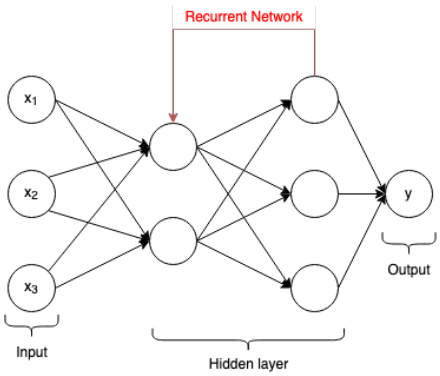


Figure 3.3: An example of a recurrent network. Feedback is introduced between the layers, enabling the network to better capture the behaviour of dynamic systems.

NARX Network

One way to describe dynamic non-linear systems, where the output $y(t)$ depends on time-series of both external inputs $u(t)$ and previous values $y(t-\Delta t)$ is as shown in (3.1). Different types of input functions $\varphi(t)$ can be used, but the most common by far is a simple vector containing previous states as in (3.2). This is known as a nonlinear autoregressive network with external inputs (NARX).

$$\hat{y}(t) = g(\varphi(t)) \tag{3.1}$$

$$\varphi(t) = g[y(t-1), \ldots, y(y-n_a), u(t-n_k), \ldots, u(t-n_k-n_b-1)] \tag{3.2}$$

Where $n_a n_k n_b$ are the chosen delays for the input and output. Because $g(t)$ is an unknown non-linear function it can be very difficult to estimate. One way is to use an artificial neural network. Such a network would be a type of recurrent network, similar to a feed-forward network, but with a delayed output of the network used as input to the network as well.

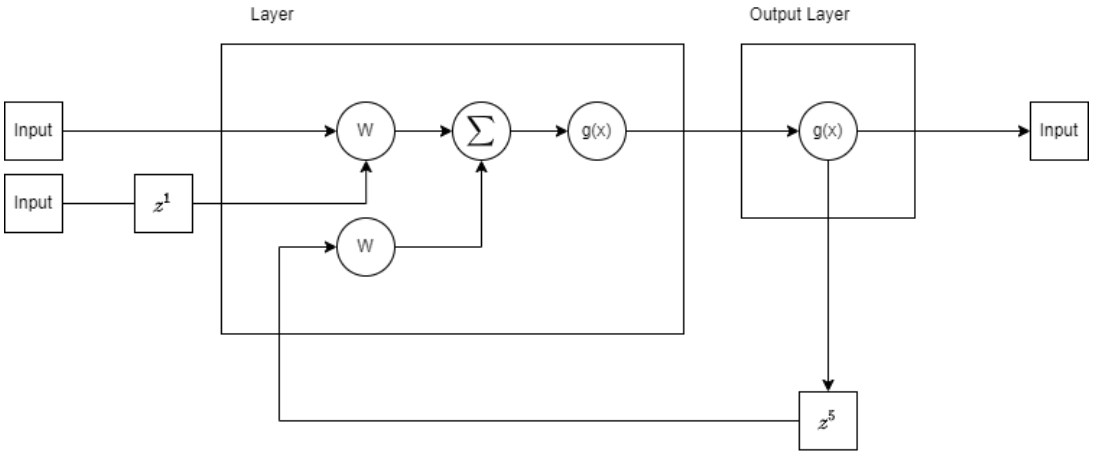


Figure 3.4: A NARX network with one hidden layer, weights W, activation function $g(x)$ and internal delay of 5 samples.

Layer Recurrent Network

Layer recurrent neural network is a simple recurrent network which is very similar to NARX. They are similar because they feedback their output with a delay into the input of the ANN making both good for transient data. The difference is that Layer recurrent feedbacks the output from the hidden layer back to the input while NARX feedbacks the output from the output layer. If the network only has one hidden layer and the output layer has a linear activation function, the layer recurrent network will be similar to a NARX network, hence the tests for layer recurrent network will use at least two hidden layers[9].

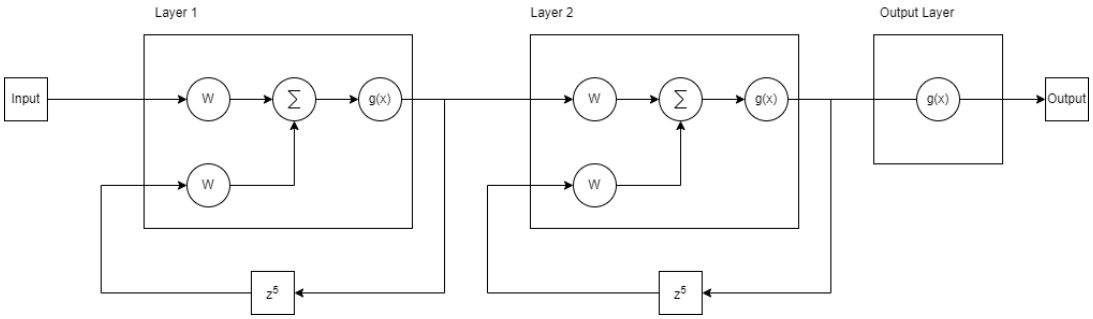


Figure 3.5: A layer recurrent network with two hidden layers, weights W, activation function $g(x)$ and internal delay 5.

Long Short Term Memory Network

The layer of a long short term memory network(LSTM) is an extension of the functionality of a recurrent network, as a recurrent neural network learns short-term relationships. The architecture of a long short term memory network includes many functionalities in comparison to a simple feed-forward network. The gates decide what values gets added or removed from the cell state which is the memory of the LSTM. The different gates use sigmoid and tanh as activation functions, this implies that the cell state either gets updated regulated or kept if the values are -1/1 or removed if the value is 0. The cell state gets updated with the input values from the input gate. The hidden state which contains information on previous inputs and is the output value that is sought after, gets updated in the output gate and then gets pointwise multiplied with the cell state which has travelled through the tanh function on the far right. these gates allow the network to learn long-term relationships in the data more effectively which gives the LSTM network the benefit to have a lower sensitivity to the time gaps, making it more suited for analyzing time series data than simple RNN [10].

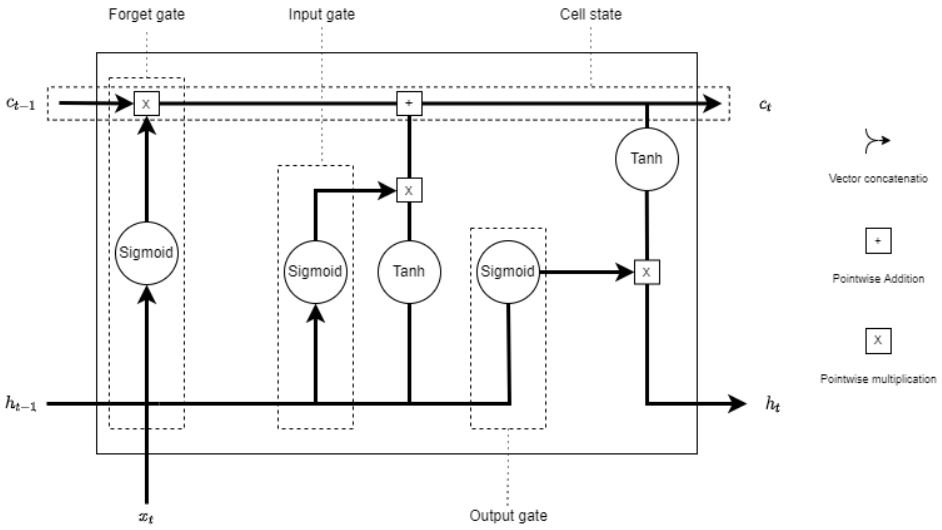


Figure 3.6: Here the input gate is the forget gate

x_t:	Input
h_t:	Hidden state
c_t :	Cell state

Table 3.1: Long short term variable description

Gated Recurrent Network

Gated recurrent network (GRU) is a more simplified version of an LSTM unit by getting rid of the cell state and using the hidden state to transfer information. The update gate acts similarly to the forget and input gates of an LSTM. it decides what information it wants to keep or throw away. Additionally GRU also has a reset gate which decides how much of the past information it should forget. It is a simplified version of LSTM, with a decrease in calculation time while trying to reach similar results to LSTM[11]

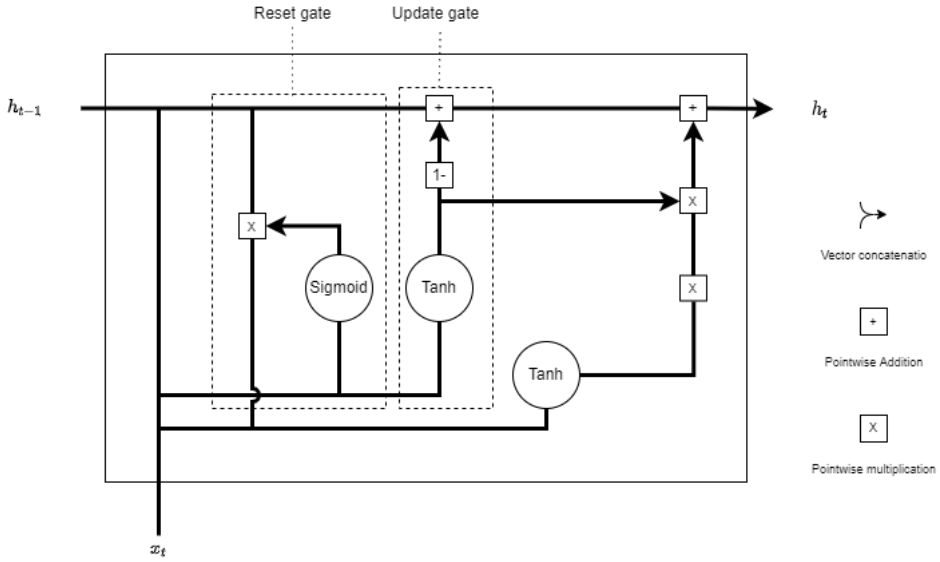


Figure 3.7: Long short term memory architecture

x_t:	Input
h_t:	Hidden state

Table 3.2: Gated recurrent unit, variable description

3.1.3 Learning in Neural network

For an ANN to learn the weights and biases of the different neurons have to be updated to accommodate and adapt to the output. This can be done in multiple ways. Three different methods will be described in this section.

Gradient descend method

This method uses the gradient of a cost function to update the weights. There are different ways of defining a cost function, in this case, the cost function can be defined as a mean squared error, as followed.

$$C(\beta, \gamma) \equiv \frac{1}{2n} \sum \|y - \hat{y}\|^2 \tag{3.3}$$

Here β denotes the collection of all weights in the network, γ all the offsets, n is the total number of training inputs, y is the output data and $\hat{y}$ defined as $\hat{y} \equiv \beta x + \gamma$. As in the name, the weights are updated to move to the lowest point in the curve meaning that the gradient

should converge to the lowest possible value i.e

$$\nabla C \equiv \left(\frac{\partial C}{\partial \beta}, \frac{\partial C}{\partial \gamma} \right) \equiv 0 \tag{3.4}$$

Defining the weight and offset parameters as $v \equiv [\beta, \gamma]$ we can calculate the new weights with the following equation.

$$v_{i+1} = v_i - \lambda \nabla C \tag{3.5}$$

where λ is a positive value called the learning rate. It is worth noting that the solution the gradient descent converges to, might not be the most optimal solution as the gradient function might have multiple minimums and even reach a maximum i.e. a bad solution. [7].

Gauss-Newton method

The Gauss-Newton method uses both the gradient of the cost function and the curvature. Using newtons method to solve $\nabla C(v) = 0$, with $v = (\beta, \gamma)$. Expanding the gradient of the cost function using Taylor series around v_0 we get.

$$\nabla C(v) = \nabla C(v_0) + (v - v_0)^T \nabla^2 C(v_0) \tag{3.6}$$

Assuming C is quadratic around v_0 and solves for minimum v we get the following update rule

$$v_{i+1} = v_i - (\nabla^2 C(v_i))^{-1} \nabla C(v_i) \tag{3.7}$$

where v_0 is replaced with v_i [12]

Levenberg-Marquardt method

The Levenberg-Marquardt algorithm varies the parameter updates between gradient descent update and Gauss-Newton Update.

$$v_{i+1} = v_i - (\nabla^2 C(v_i) + \lambda diag[\nabla^2 C(v_i)])^{-1} \nabla C(v_i) \tag{3.8}$$

where small values of the λ parameter will result in a Gauss-newton update and large values of λ result in a gradient descent update. Since the hessian is proportional to the curvature of C, 3.8 implies a large step in the direction with low curvature and a small step in the direction with high curvature [12].

3.1.4 Activation Functions

The activation function is the function that is evaluated for each neuron. $\kappa(z)$. Here z will be assumed to be $\beta^T x + \gamma$, where β is a vector of parameters (weights) and x is a vector of the input for each neuron, and γ is a parameter (bias).

Name	Function	Range
Sigmoid	$\kappa(z) = \sigma(z) = \frac{1}{1+e^{-z}}$	$\kappa(z) \in [0, 1]$
Hyperbolic Tangent	$\kappa(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$\kappa(z) \in [-1, 1]$
Linear	$\kappa(z) = z$	$\kappa(z) \in]-\infty, \infty[$
Rectified Linear Unit	$\kappa(z) = \max(z, 0)$	$\kappa(z) \in [0, \infty[$

Table 3.3: Some of the most commonly used activation functions.

The sigmoid function works well with classification problems since it squishes the numbers between 0 and 1. Another advantage of the sigmoid function is that it is continuously derivable, and the gradient is contained within reasonable limits. The hyperbolic tangent function

improves upon this by extending the output range from negative one to positive one, thus including the possibility of negative numbers. If the network is shallow, however, then there is usually little difference in performance between the two [13].

Purely linear activation functions can be used but have some disadvantages. The most obvious is that it does not introduce non-linearities in the network. Rectified linear unit (ReLU), however, is quickly becoming the most commonly used activation function [14]. The advantage of using such a function is that it is non-derivable at zero. When the neuron can be considered deactivated, which speeds up the learning process of the network. It also introduces non-linearities, compared to the purely linear which does not.

3.1.5 Cross covariance, Auto covariance

To check the linear relationship between two signals or the same signal in different time steps, cross-covariance and auto covariance are used. This will be of use to see how useful the signal is to the output one trains for or how the sample delay of the output affects the output itself [15]. Cross covariance is defined as the following equation.

$$\hat{R}_{yx}^{N} = \frac{1}{N}\sum_{t=1}^{N} y(t)x(t-\tau) \tag{3.9}$$

And auto covariance.

$$\hat{R}_{yy}^{N} = \frac{1}{N}\sum_{t=1}^{N} y(t)y(t-\tau) \tag{3.10}$$

3.1.6 Overfitting

An important aspect of neural network models is the ability to generalise the system behaviour for different data sets. If the model is overfitted to the training set then it might accidentally try to capture the behaviour of measurement noise, which should not be explainable with the given inputs. This will lead to the model fitting well to the training set but give bad predictions for new data sets. If a network is deeper than necessary it will likely lead to over-fitting [13]. It is also important to note that not only the network parameters can lead to over-fitting, but the input used as well. If multiple input contain the same information or permutations of the same information, this might also lead to overfitting [16].

3.1.7 Evaluation Methods

To evaluate the performance of the neural network two evaluation methods will be used. The first method is the coefficient of determination defined as followed

$$R^2 = 1 - \frac{\sum_{t=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{t=1}^{N}(y_i - \bar{y}_i)^2} \tag{3.11}$$

and the second method is normalized root-mean-square.

$$\text{NRMSE} = \sqrt{\frac{||y - \hat{y}||^2}{N \cdot ||y||^2}} \cdot 100 \tag{3.12}$$

where N represents the size of the test set, y represents the actual data, $\hat{y}$ is the neuron output and $\bar{y}$ is the mean of the actual data. An R^2 value equal to 1 implies a perfect linear relationship between the original data and the ANN output and for NRMSE a smaller value suggests better forecasting [17][18][19].

A more intuitive way to compare different models of the same system is to simply look at the absolute error $|\epsilon|$. This method is not normalised, so it should not be used to compare models of different systems or different inputs.

$$|\epsilon| = |y - \hat{y}| \tag{3.13}$$

It is also useful to plot the output of the neural network and the data set to get a visual understanding of the results.

4 Available Data

The data used for the system identification are collected by Aurobay and measured using the same engine type. The location and environmental conditions for the tests vary heavily. The data is divided into two categories: steady-state part load mappings and transient data. The steady-state data will use five different data sets collected from engine test cells.

The transient data is from different environments. This includes three different runs in Kiruna Sweden, six Gibraltar data sets, one data set collected in Spain and one data set collected in the more southern warmer part of Sweden. Some of these sets are with warm start and some with cold start. The starting condition for each data set follows the table 4.1. The velocity profile for each data set can be seen in Figure 4.1. The engine load to engine speed for the steady-state data can be seen in figure 4.2.

Data set	Used for	Starting condition
Spain	Training	warm
VNT	Training	warm
Gibraltar 1	Training	cold
Gibraltar 2	Training	warm
Gibraltar 3	Training	warm
Gibraltar 4	Training	cold
Gibraltar 5	Training	warm
Gibraltar 6	Validation	warm
Kiruna 1	Training	warm
Kiruna 2	Validation	cold
Kiruna 3	Training	cold

Table 4.1: Data sets for transient data

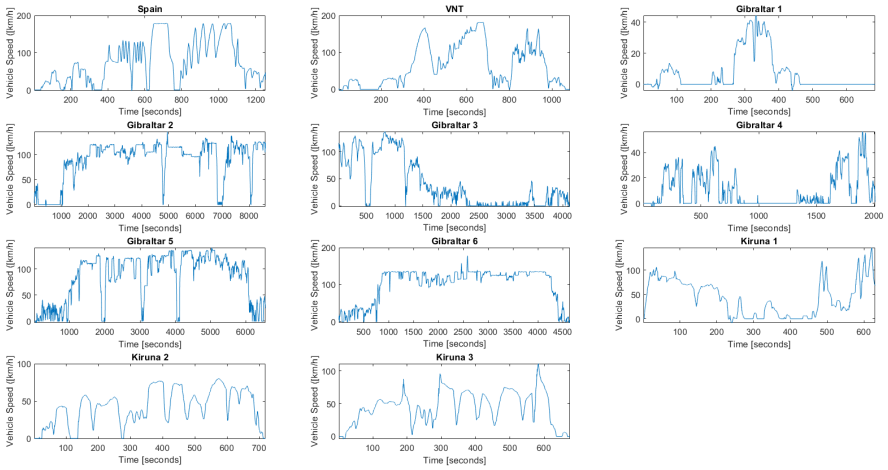


Figure 4.1: Vehicle speed for each data set.

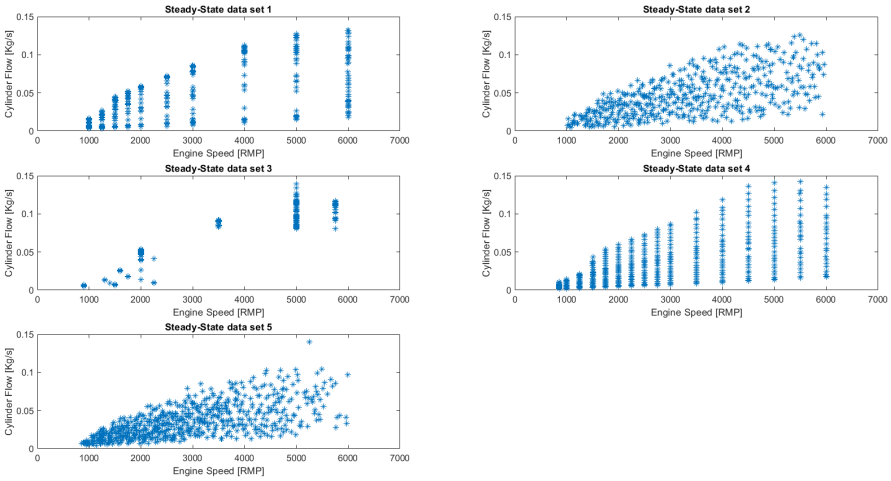


Figure 4.2: Engine load to Engine Speed for the steady-state data sets

It is also important to mention that the temperature measurements come from a thermocouple sensor. This sensor needs to be resistant to the heat in the exhaust gas, which gives a slow response time. The sensor dynamics will thus be included in the model and high frequency transients will be excluded from the model.

5 Method

This chapter explains the data pre-processing and system identification methods used in the thesis to estimate the exhaust gas temperature. A structured black-box approach is used. Simple feed-forward models are used to estimate the cylinder flow, $IMEP_g$ and PMEP. These features are then used as an input to a recurrent neural network estimating the exhaust gas temperature. The temperature in turn is then used as an input to the $IMEP_g$ and cylinder flow networks, according to Figure5.1. While a simple feed-forward network is expected to work for the systems with fast dynamics (cylinder flow, $IMEP_g$ and PMEP), different types of recurrent networks will be evaluated for the exhaust gas temperature.

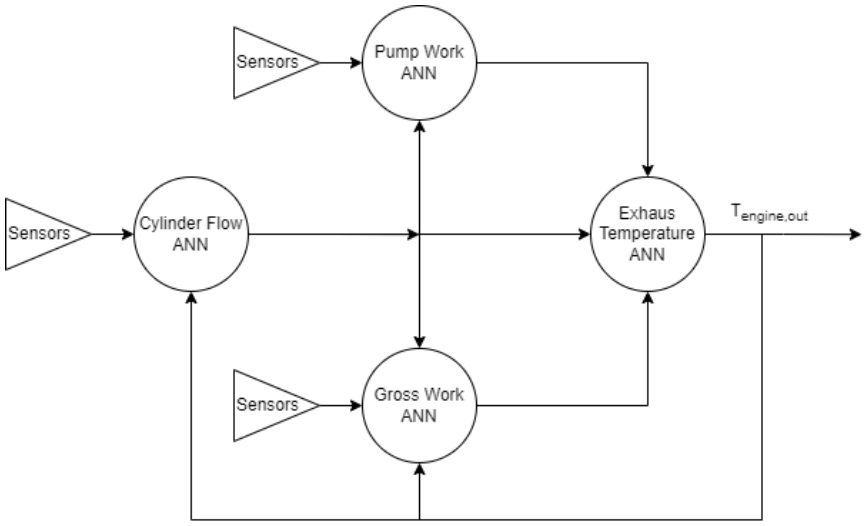


Figure 5.1: Black-Box Exhaust Temperature Model

5.1 Choice of programming tools

There are many different kinds of tools for creating and training artificial neural networks. Two of the most common choices are to use either the Pytorch or TensorFlow modules for Python. These modules are focused on creating deep networks and do therefore not include the Levenberg-Marquardt algorithm, which is more suitable for shallow networks as this algorithm can find very good results during the optimization process as the gradient step adjusts itself accordingly but has high computational complexity. Instead, the Deep Learning Toolbox in Matlab is used for this thesis.

5.2 Pre-processing and Training

Before training the data need to be pre-processed. This section describes the data and the steps necessary to pre-process the data for training, as well as the methods used for training the networks.

5.2.1 Non-reasonable Values

Some adjustments need to be made in the cases where the data set includes non-reasonable values. For the part-load mapping, the data points which include such values are simply removed. For the time-series data, all work is put to zero when the engine speed is zero.

5.2.2 Normalisation

The different inputs used for the neural network use different units, and the scale of these units varies massively. The pressures for example are usually in the size of 10^5 while the injected fuel mass is closer to 10^{-3}. This means that changes in the larger inputs affect the network more than changes in the smaller inputs. In order to counteract this, all inputs are normalised to fit in the range $[0, 1]$. This is done by establishing in which range the inputs are normally operating. These ranges can be seen in 5.1. An input $z \in [z_{min}, z_{max}]$ can be normalised to $z^* \in [0, 1]$ using (5.1).

Input	Range
Intake Pressure	0 - 4 [Bar]
Exhaust Pressure	1 - 3[Bar]
Intake Temperature	223 - 373 [K]
Exhaust Temperature	223 - 1373 [K]
Ambient Temperature	253 - 313 [K]
Coolant Temperature	258 - 363 [K]
Engine Speed	600 - 6000 [RPM]
Cylinder Flow	0 - 0.14 [kg/s]
Fuel Injection	0 - 0.01 [kg/s]
Intake VVT	0 - 70 [Degrees]
Exhaust VVT	0 - 35 [Degrees]
Ignition Angle	-20 - 45 [Degrees]
PMEP	-1.75 - 0.35 [Bar]
IMEP	0 - 20 [Bar]
Grill Shutter	0 - 100 [%]
Vehicle Speed	0 - 50 [m/s]

Table 5.1: Normalisation Ranges

$$z^* = \frac{z - z_{min}}{z_{max} - z_{min}} \tag{5.1}$$

The reasoning for using a table of pre-determined ranges rather than normalising against the highest and lowest values of the data set is two-fold. Firstly, since the output can be any value the normalisation needs to be the same for every data set, not just the training set. Secondly, the operating ranges for the different inputs are well known, and might commonly include regions not present in the training set.

A normalised value can be returned to its original size with (5.2).

$$z = z^*(z_{max} - z_{min}) + z_{min} \tag{5.2}$$

5.2.3 Sampling Frequency

In some cases, the data might not have the same sampling frequency. One of the main reasons why it is necessary to resample the data to a uniform frequency is that it can affect the results when it comes to testing different feedback delays for NARX, as the change in the output in a higher frequency will be smaller than the lower sampled data set. To solve this, the data will be resampled to the same frequency of 25 Hz. Other benefits to resampling the data set are as follows.

- Downsampling shortens the length of the data in turn reducing calculation time when training the ANN.
- Downsampling the data set reduces the high-frequency noise.
- Resampling to a uniform time difference means that some data points that seem to be missing from the measurements are interpolated instead.

5.2.4 Preparing data for ANN Training

Feed-Forward Networks

For the feed-forward networks simply using a collection of data points is enough, as the ordering of these points should not matter. Here, part load mapping data are intended to be used. That is because a data set collected in an engine test cell will collect all of the characteristics of an engine from max to min in load and engine speed. It is however also possible to use transient time-series data, as the dynamics for these systems are assumed to be fast in the first place.

Recurrent Networks

While training a recurrent network it is necessary to use only time-series data.

For training a NARX network with multiple data sets at the same time. The data will be set up as an array with the column corresponding to the data set, and the row representing the input and output at that specific timestep.

Figure 5.2: [XXX] representing a vector with the inputs in a timestep and T_{e0} the output at the same timestep

During the training, the data have to be separated and trained in indices so that the network gets its data in a time-series structure and not randomly from the data-sets.

Long short term memory/Gated Recurrent network

To train these networks, the training would have to run separately with different data sets to be able to train multiple scenarios. So all of the data will be set up individually and trained separately to the same network.

5.2.5 Input-output Correlation

To check if the input will be useful for training the ANN, it is good to check the cross-covariance of the variable against the output. Covariance might have the drawback that it only shows the linear relationship between two signals, which might not mean that the specific input does not affect the output that one uses covariance for if the covariance is low. For training an ANN, this will be useful because the ANN looks specifically at the behaviour of the different inputs and adjusts weights to accommodate the output. But it is good to consider the actual physics of the variable you are training for and test these inputs that still have an effect, to see if they bring any improvements to the results of the ANN.

5.3 Network training Process

This section mentions how the networks are trained and which data are used.

5.3.1 Training Data

The feed-forward networks are independent of time. Because of this, the neural network is trained using a data-set of singular data points, no time series are used. The inputs used for each network can be seen in table 5.2 and table5.3.

Cylinder Flow	PMEP	$IMEP_g$
Engine On/Off	Engine On/Off	Engine On/Off
Intake Pressure	Intake Pressure	Intake Pressure
Exhaust Pressure	Exhaust Pressure	Exhaust Pressure
Intake Temperature	Cylinder Flow	Cylinder Flow
Exhaust Temperature	Intake VVT Angle	Intake Temperature
Intake VVT Angle	Exhaust VVT Angle	Exhaust Temperature
Exhaust VVT Angle	Engine Speed	Intake VVT Angle
Engine Speed		Exhaust VVT Angle
		Engine Speed
		Ignition Angle
		Fuel Injection Rate

Table 5.2: Input used for the static neural network models

Exhaust Gas Temperature
Engine On/Off
Cylinder Flow
$IMEP_g$
PMEP
Vehicle Speed
Engine RPM
Fuel Injection Rate
Intake Pressure
Exhaust Pressure
Intake Temperature
Intake VVT Angle
Exhaust VVT Angle
Ignition Angle
Coolant Temperature

Table 5.3: Input used for the temperature neural network models

5.3.2 Network Training

The networks are trained using Matlabs Deep Learning Toolbox. They are trained using the Levenberg-Marquardt algorithm, excluding LSTM which does not have the Levenberg-Marquardt algorithm and uses gradient descent instead. The reason for this is because of the way the algorithm can adjust its step size if needed to find an optimum. An example of the benefit is when descending a very steep local minimum parable, it is necessary to use small step sizes to avoid passing over that minimum and vice versa for the opposite case. If in the case of using gradient descent, the step size would be fixed, so would take much longer to

find that optimum for a small step size. During training, if the gradient becomes too small or the number of iterations too large the training is deemed finished. The initial weights used for training are normally randomised. This can cause some problems when evaluating different network structures as it is difficult to see if the varying results are because of the random weights or because of the different structures. For this reason, several networks are trained. The network that performs best on a separate validation set is then chosen.

Early Stopping

In order to avoid over-fitting and improve generalisation a technique called early stopping is used. This is implemented by dividing the data set into two independent sets. One set is used for estimating the parameters of the neural network and one set for validating the network. If by further improving the performance of the training, the performance on the validation set is reduced for several iterations in a row, the training process is halted, as further training reduces generalisation. For this thesis, the number of iterations for validation checks is set as six.

For static data sets the order of the data points is irrelevant. Thus the training-validation divide can be set as random. For time-series data the set is split into two blocks. The ratio used is 85 % of the data for training and 15 % for early stopping validation. It should be noted that the early stopping validation set is still a part of the training process, and should not be confused with independent validation sets.

Initial Weights

The initial weights for the training are randomised. This can have the effect that the same training process results in networks with different performances. Ideally, the training process finds a global optimum. However, it is also likely that the training process finds a local optimum, and that the local optimum found might vary depending on the initial weights.

Open-loop training of NARX-networks

When training a NARX network it can be seen as a feed-forward neural network which uses a time series, as well as a delayed version of the time series, as an input. This is in contrast with when the network is used for prediction, where the previous estimates have to be fed back as an input to generate predicted time series, as shown in **??**. This type of feed-forward training is fast compared to other recurrent neural networks, which are usually trained by being unfolded in the time dimension.

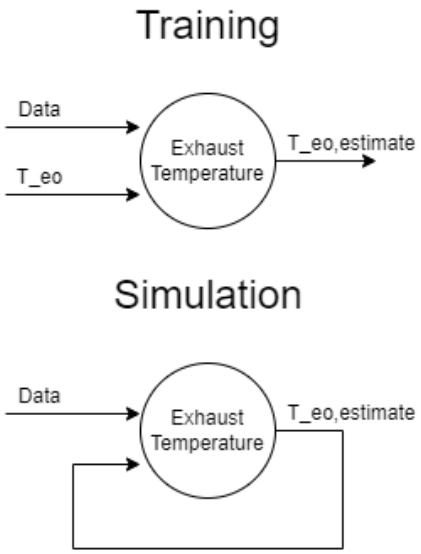


Figure 5.3: When using the neural network for prediction the output is used as feedback to generate time-series data.

5.3.3 Network Structure

The network structure can be defined in large by its number of layers and number of neurons in each layer, as well as the activation function used for the neurons. The recurrent networks might have further parameters related to the recurrent nature of the networks. For example, the number of delays, and the positioning of these delays, is an important design choice for the NARX network. The number of delays used internally for the layer recurrent network is also important.

How these parameters affect the performance of the network for each network type and system are investigated to decide an optimal structure. Due to the random initial weights, several networks need to be trained for each setup. Only one structure parameter is investigated at a time to reduce the complexity of the analysis.

Number of neurons

For finding a good structure of the network, a test on the number of neurons needs to be done. This test can be done by locking the other changeable parameter, and varying the number of neurons while training multiple networks in a number of iterations to minimize the randomness of the result, and calculating the NRMSE for each iteration.

Activation functions

To find an appropriate activation function multiple networks will be trained where the activation functions are varied between, linear ReLU, tanh and sigmoid.

Feedback Delay and input delay

This test will only apply to NARX and layer-recurrent neural networks, with NARX having the additional test on the input delay. The delay, similar to the neuron test will be varied while locking the other parameters and calculating the NRMSE for each new network created for a number of iterations.

5.3.4 Validation on steady-state data

The feed-forward ANNs will be validated simply using the steady-state data as input. For the dynamic networks, it will be necessary to generate a time series of constant values to see where the simulation ends up in steady state.

5.3.5 Validation on transient data

For the exhaust gas temperature and the cylinder flow, the validation will be done against the measured sensor value. In the case of the $IMEP_g$ and PMEP, the validation will be done against Aurobays model. The data used for validation consists of one warm start and one cold start data set.

6 Results

This chapter presents the results for each system and network type separately. Both in regards of finding the optimal structure and the validation of these networks. The structure for the result will be the following

- Cylinder Flow: FNN
- PMEP: FNN
- $IMEP_g$: FNN
- Exhaust Gas Temperature Model: FNN
- Exhaust Gas Temperature Model: NARX
- fully connected ANN with NARX
- Exhaust Gas Temperature Model: LSTM
- Exhaust Gas Temperature Model: GRU
- Exhaust Gas Temperature Model: LRN

6.1 Cylinder Flow: FNN

6.1.1 Network Structure

First, a single layer is tested with a varying amount of neurons. Several separate networks are trained for each amount of neuron. The cylinder flow network is evaluated on Kiruna 2 and Gibraltar 6 data sets and the other transient data sets are used for training. The performance as a function of the number of neurons in a single hidden layer network can be seen in Figure6.1. The results vary slightly despite having the same amount of neurons. This is because of the randomised initial weights and is to be expected. The green bar represents the 95 % confidence interval. It can be seen that adding more than six neurons has little effect, if none, on the performance of the network. Another thing to note is that the network that performs best on the training data also performs well on the validation data for all numbers of neurons.

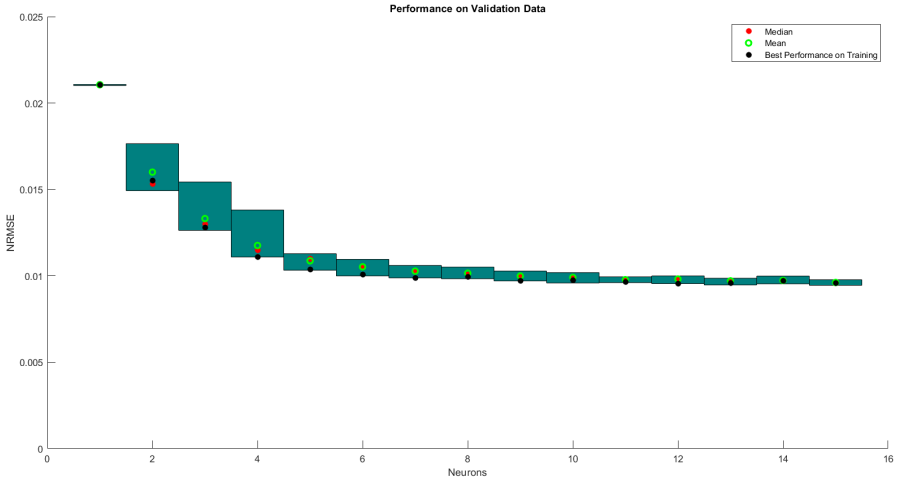


Figure 6.1: Cylinder flow FFNN performance dependence on number of neurons.

In the figure,6.2 a network with six neurons in the first layer and a varying amount of neurons in the second layer has been tested. It appears that adding a second layer does not affect the performance at all.

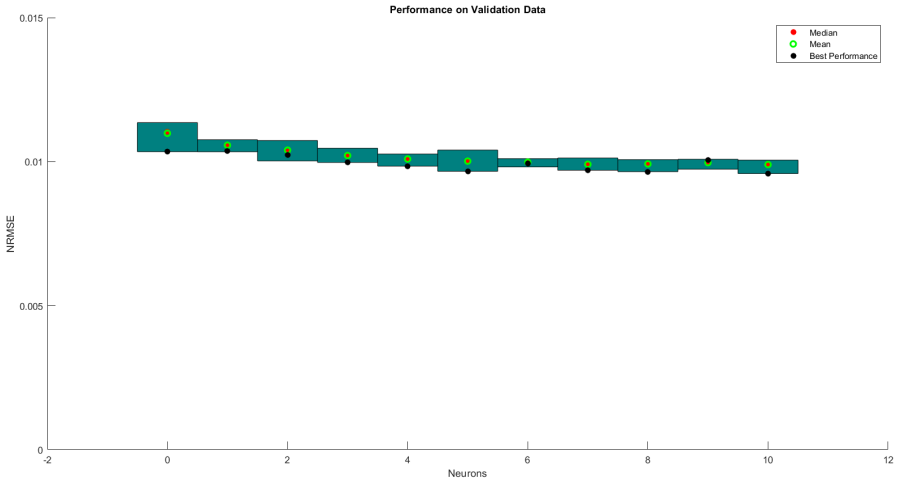


Figure 6.2: The performance of different networks. The number of neurons in the first hidden layer is six and the number of neurons in the second hidden layer changes as seen in the figure.

Figure 6.3shows the performance of a one hidden layer network with varying amounts of neurons for different activation functions in the hidden layer (for previous testing only hyperbolic tangent function has been used). It can be seen that the sigmoid and the hyperbolic tangent function is best suited for the cylinder flow model.

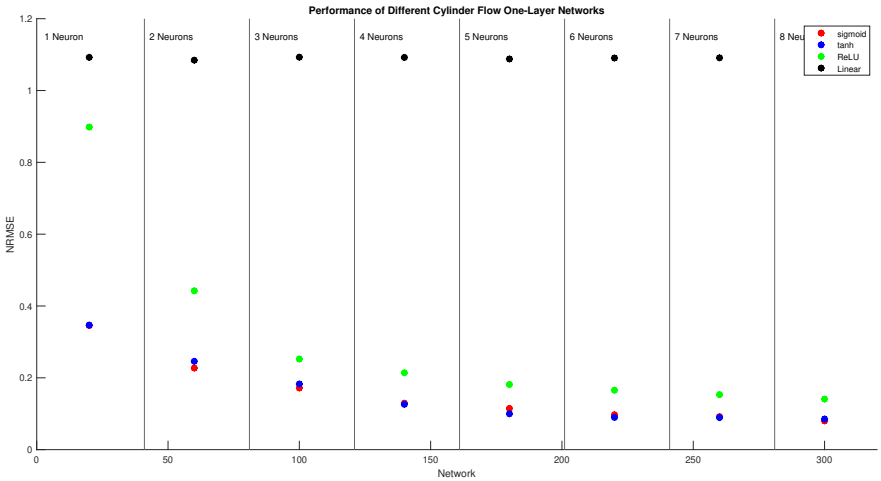


Figure 6.3: The performance of different activation functions.

6.1.2 Validation on Static Data

Figure 6.4shows the estimation of the cylinder flow model compared to the measurements for a steady-state data-set that was not involved in the training of the network. An R^2 of

0.9956 was achieved on the evaluation set. This corresponds to an average absolute error of 1.2 grams per second.

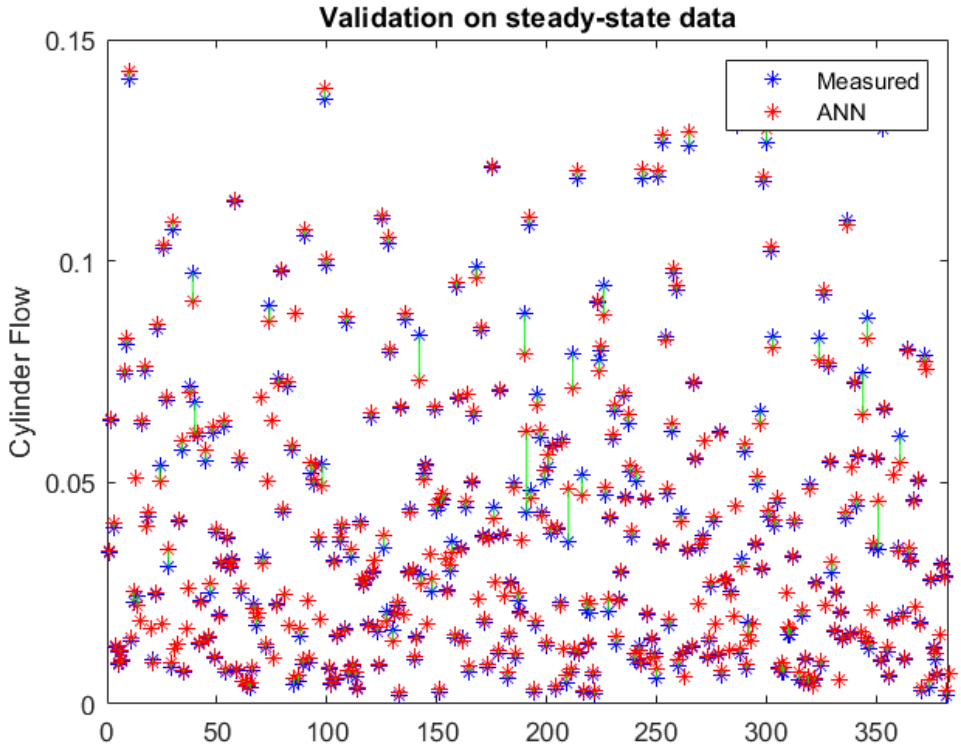


Figure 6.4: Validation of the cylinder flow model on a part-load data set

6.1.3 Validation on Transient Data

The cylinder flow is the only static model that is trained on transient data. The networks performance on the data sets not used for training can be seen in Table6.1. The cylinder flow model handles cold starts well, although not to the same degree as warm starts. In Figure,6.5 the cylinder flow model can be seen compared to the measured data from the data gathered in Kiruna.

Data set	R^2	NRMSE	Starting condition
Gibraltar 6	0.99743	0.0098971	warm
Kiruna 2	0.98832	0.053353	cold

Table 6.1: Performance of the cylinder flow model on different transient data sets.

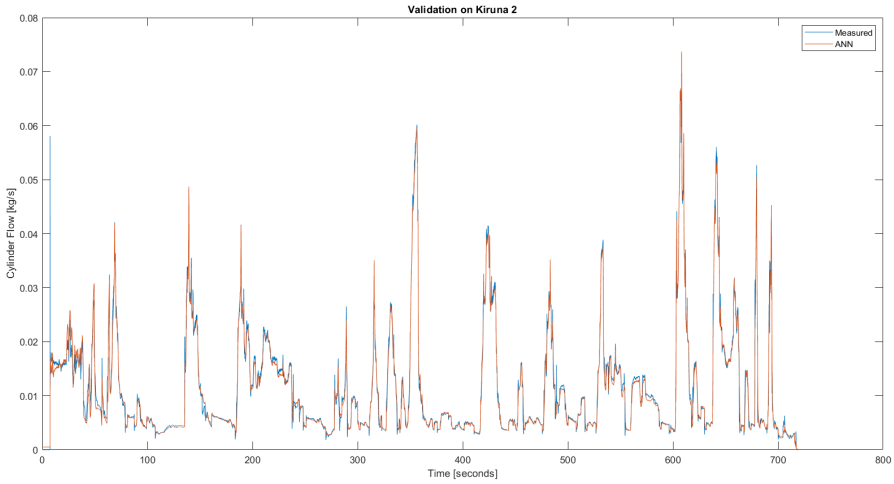


Figure 6.5: Cylinder flow model on the data-set with cold start.

6.1.4 Engine On/Off input

In Figure, 6.6it can be seen that the model correctly predicts that the cylinder flow is zero when the engine is turned off. This is due to the binary engine on/off signal provided as an input to the network. Without including such a signal the model tends to undershoot and predict a negative cylinder flow, such as in Figure6.7. While a back-flow is not impossible, it should not happen at these points. Obviously, no steady-state data has been collected when the engine is turned off, therefore it is necessary to add artificial data for the networks that are trained on steady-state. This data consists of random noise for all signals except the engine on/off and cylinder flow, which is zero.

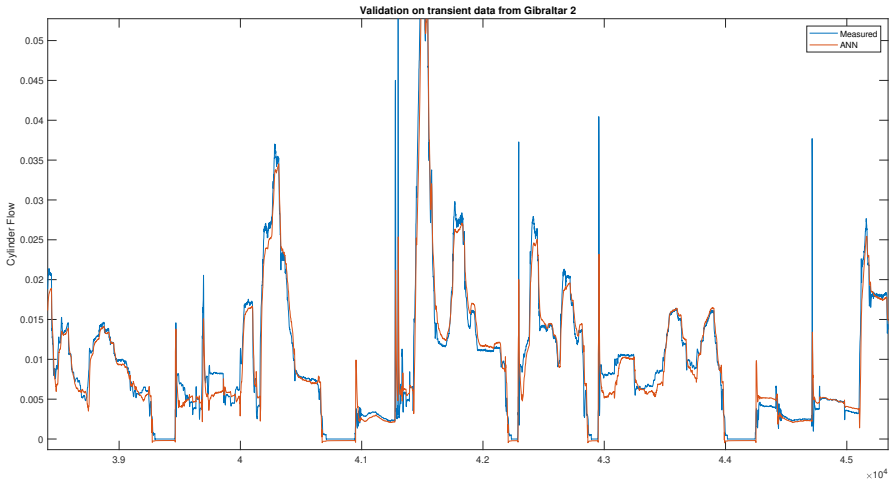


Figure 6.6: With the engine on/off input, the model correctly predicts no flow when the engine is off.

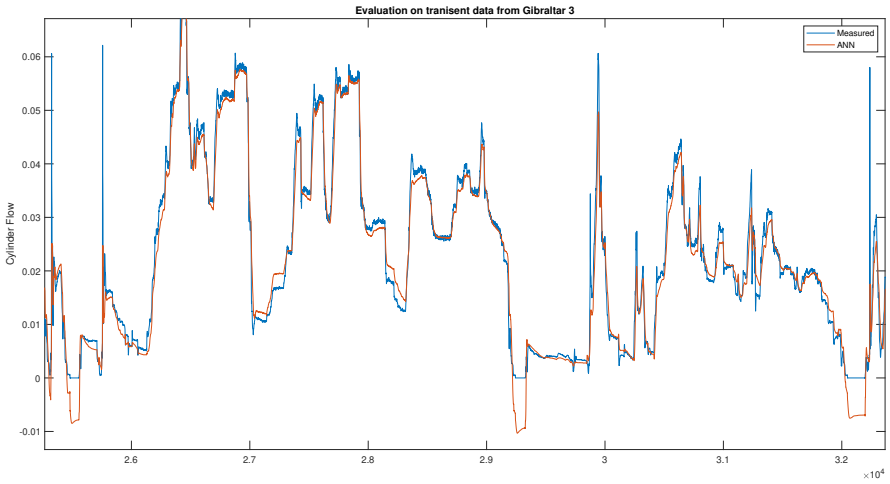


Figure 6.7: Without the engine on/off input, the model incorrectly predicts negative values when the engine is off.

6.2 PMEP: FNN

Network Structure

In Figure 6.8 it can be seen that about five neurons are sufficient to model the pump work. The networks are trained on steady-state data and evaluated on different steady-state data.

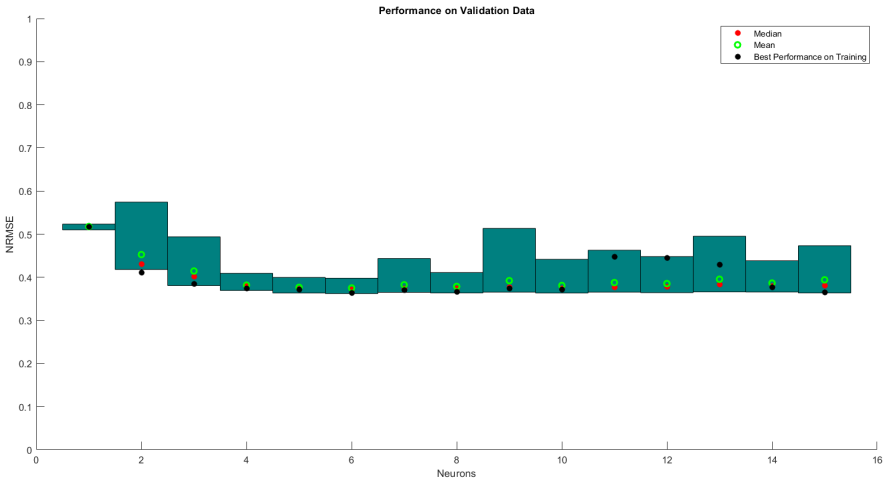


Figure 6.8: The performance of a trained network depending on the number of neurons.

Using six neurons in the first hidden layer and changing the number of neurons in the second shows that a second hidden layer does not affect the performance of the network, which can be seen in6.9.

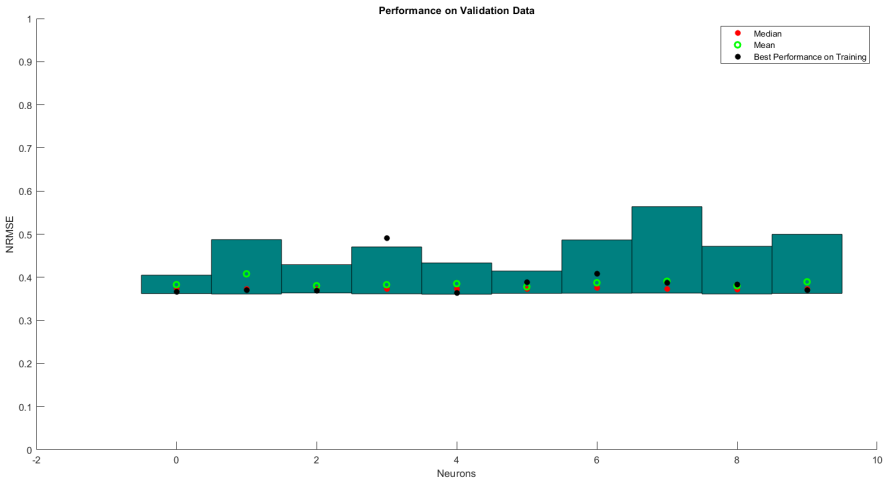


Figure 6.9: The performance of different networks. The number of neurons in the first hidden layer is six and the number of neurons in the second hidden layer changes as seen in the figure.

In Figure 6.10 it can be seen that the hyperbolic tangent function and the sigmoid function performs better than the other two types for the pump work as well. Although ReLU seems to catch up with the other two when using a higher number of neurons.

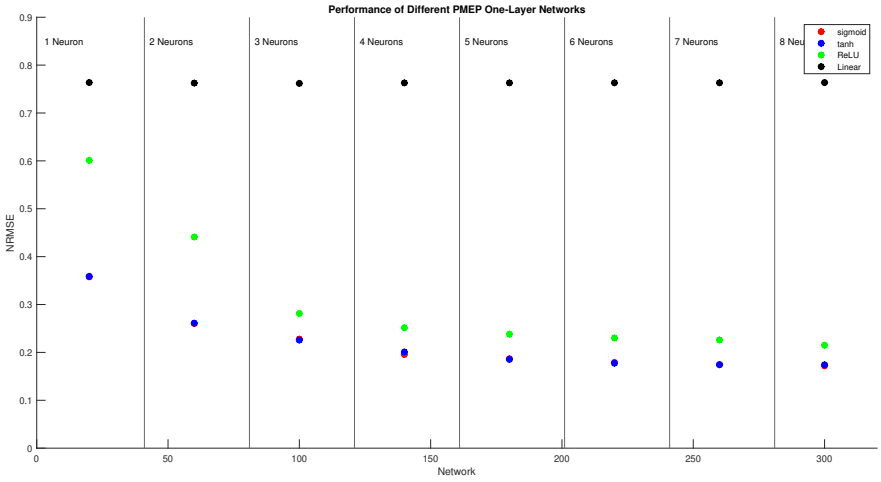


Figure 6.10: The performance of different activation functions in a one hidden layer network.

6.2.1 Validation on Static Data

In Figure 6.11 it can be seen that the network predicts pump work on steady-state data well. The performance is $R^2 = 0.995$ and an average absolute error of 1.7 kPa.

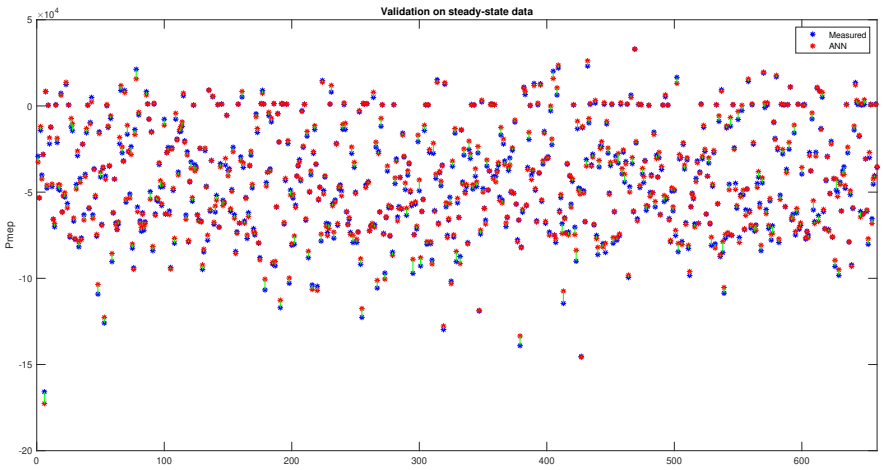


Figure 6.11: Validation of the PMEP model on a part-load data set

6.2.2 Validation on Transient Data

Just as with the IMEP_g the PMEP can not be evaluated against measured values and is instead validated against Aurobays modelled PMEP. The PMEP network follows Aurobays model

with little deviation. An example of this can be seen in Figure, 6.12. In Figure 6.13it can be seen that there is a slight static error during cold start.

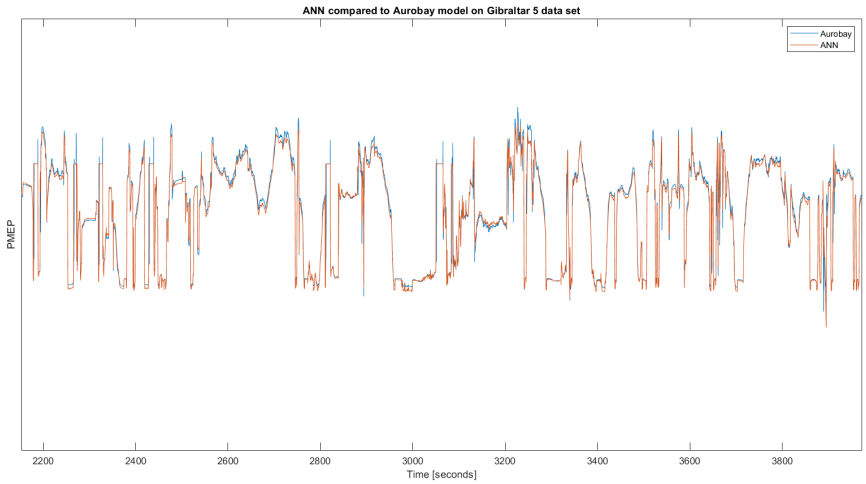


Figure 6.12: Validation on warm start transient data set.

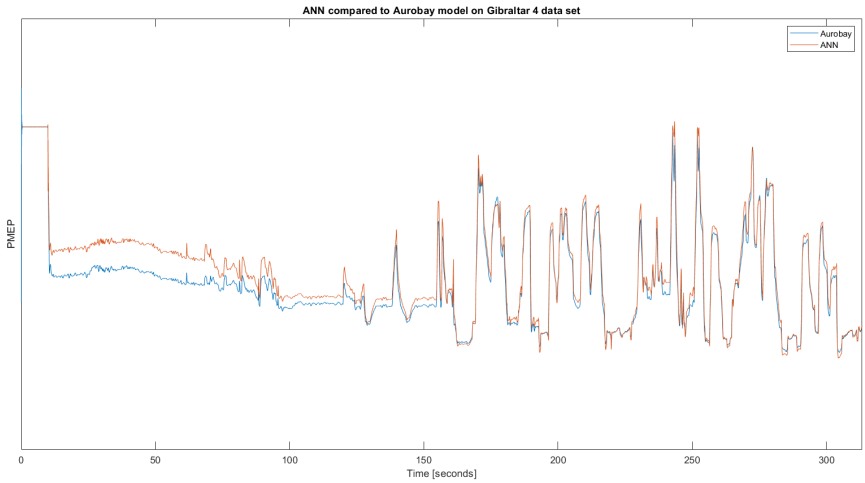


Figure 6.13: Validation on cold start transient data set.

6.3 IMEP$_g$: FNN

6.3.1 Network Structure

The results for how the number of neurons in a single hidden layer network impacts the IMEP$_g$ can be seen in Figure 6.3.1. The networks are trained on steady-state data and evaluated on different steady-state data. The performance seems to converge after about four to six neurons.

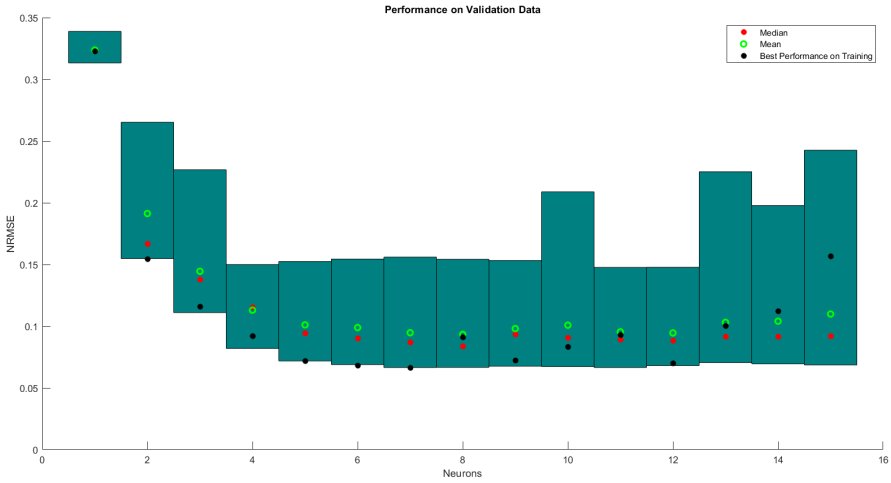


Figure 6.14: The performance of different activation functions in a one hidden layer network.

Just as with the cylinder flow and PMEP, The number of neurons in the second layer of a two hidden layer network with six neurons in the first hidden layer is evaluated. Figure 6.3.1 shows that a second layer has little effect on the performance of the network.

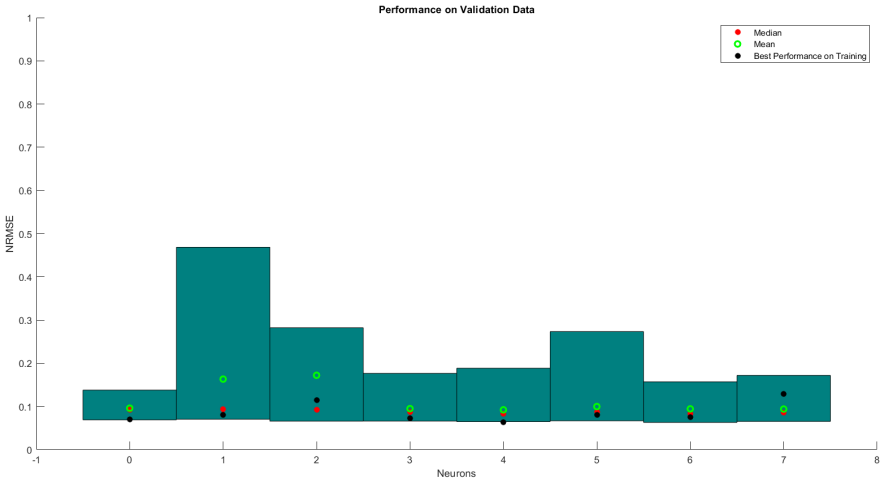


Figure 6.15: The performance of different networks. The number of neurons in the first hidden layer is six and the number of neurons in the second hidden layer changes as seen in the figure.

Figure 6.3.1 shows the performance of different activation functions. Due to the difference between the transient data set and the steady-state sets the evaluation is done only on the steady-state data set. Similarly to the cylinder flow and PMEP the hyperbolic tangent function and the sigmoid function gives better results than ReLU.

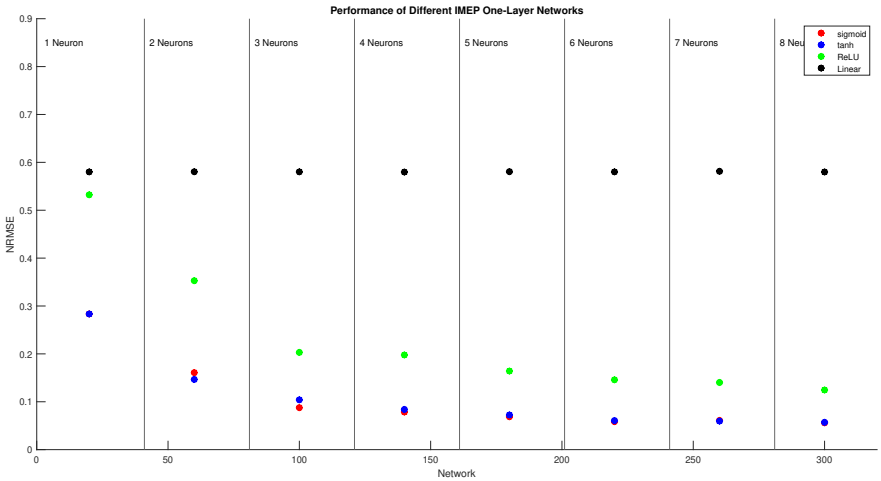


Figure 6.16: The performance of different activation functions in a one hidden layer network.

6.3.2 Validation on Static Data

The performance can be seen in Figure 6.17. R^2 is 0.9994 the NRMSE is 0.0654 and the average absolute error is 7.2 kPa.

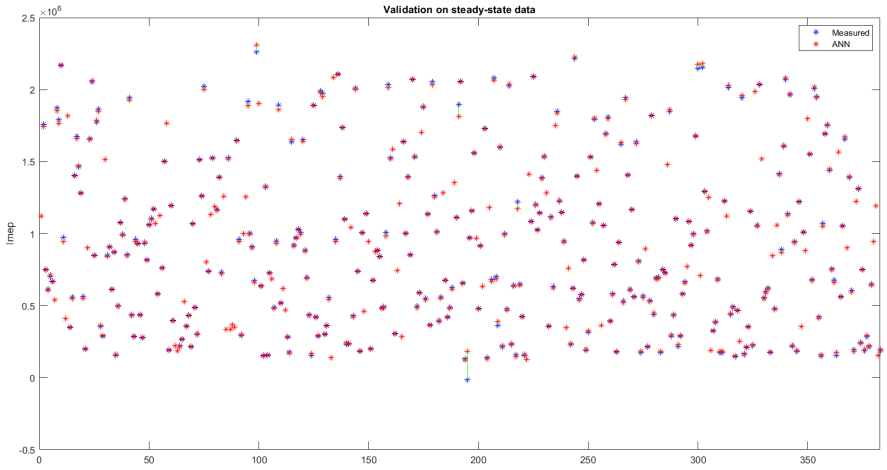


Figure 6.17: Validation of the IMEP model on a part-load data set

6.3.3 Validation on Transient Data

Validating the IMEP$_g$ against the transient data sets would require the cylinder pressure to be measured. Something that is not done. Instead, our neural network model will be validated against Aurobays model. In Figure 6.18 it can be seen that the ANN model does follow Aurobays model, however, it does have a lot of spikes. A similar result can be seen in Figure 6.19, but with fewer spikes. Figure 6.20 shows that the network handles cold starts well, with the exception of the very first couple of seconds.

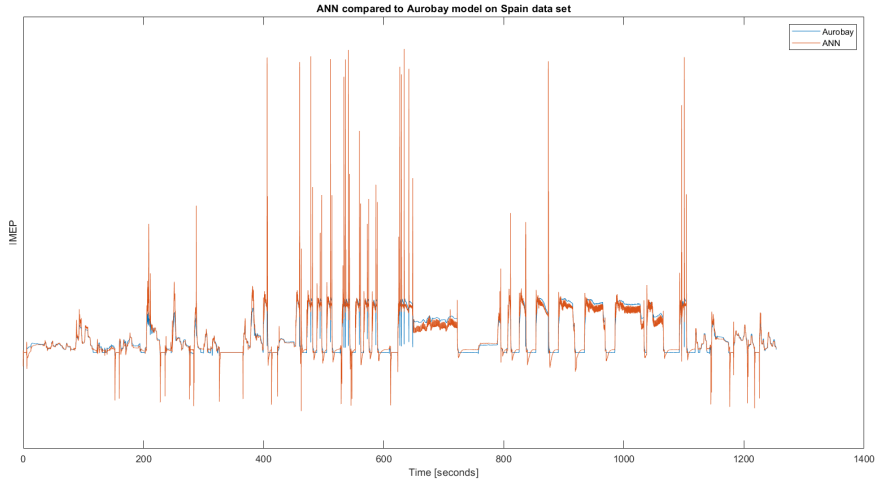


Figure 6.18: Validation on warm start transient data set.

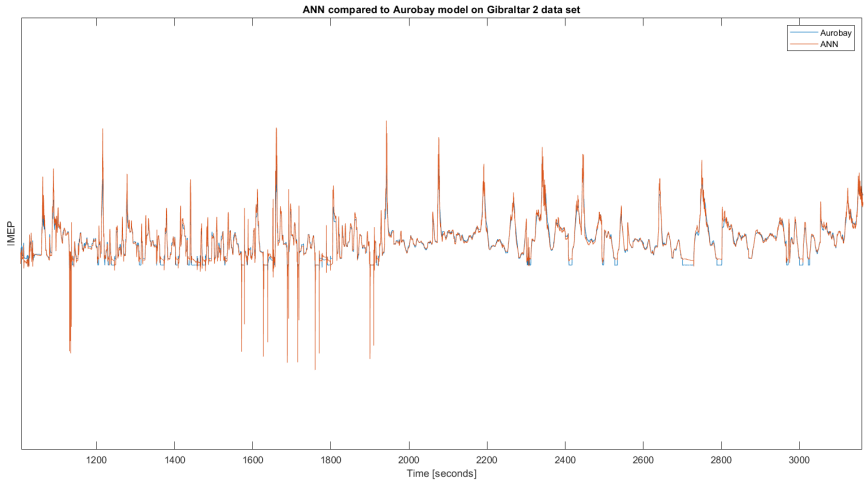


Figure 6.19: Validation on warm start transient data set.

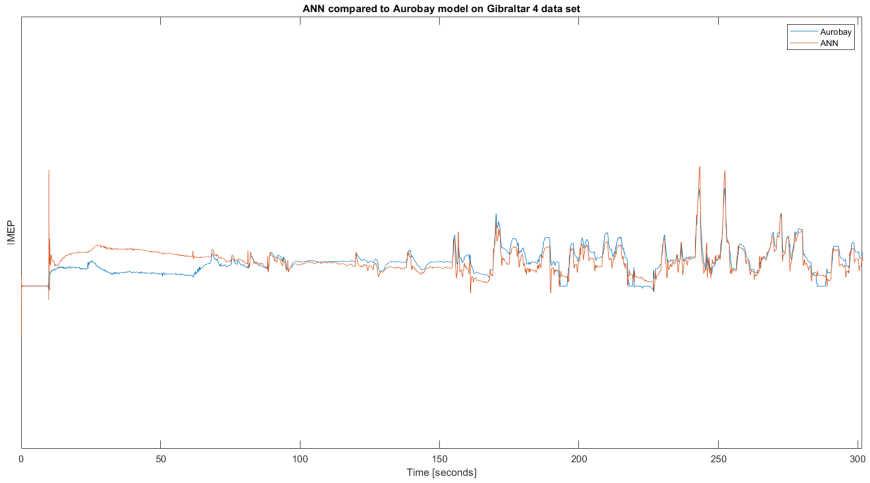


Figure 6.20: Validation on cold start transient data set.

6.4 Exhaust Gas Temperature Model: FNN

Network Structure

The performance as a function of the number of neurons for the exhaust gas temperature can be seen in Figure 6.4. It can be seen that no more than four neurons are needed.

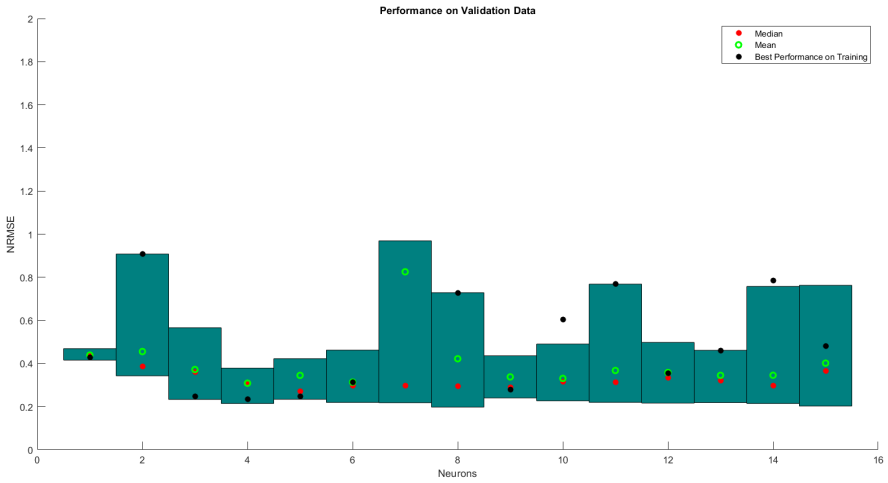


Figure 6.21: The performance of a trained network depending on the number of neurons.

The performance as a function of the number of neurons in the second layer for the exhaust gas temperature can be seen in Figure 6.4. It can be seen that adding a second layer has little effect.

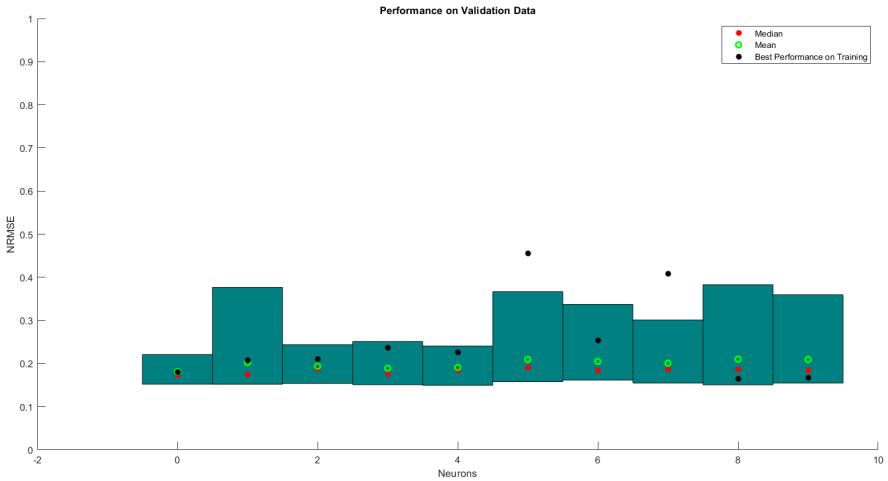


Figure 6.22: The performance of different networks. The number of neurons in the first hidden layer is four and the number of neurons in the second hidden layer changes as seen in the figure.

The performance as a function of the number of neurons and activation function can be seen in figure 6.4. The figure shows that either the logistic sigmoid function or the hyperbolic tangent function is optimal.

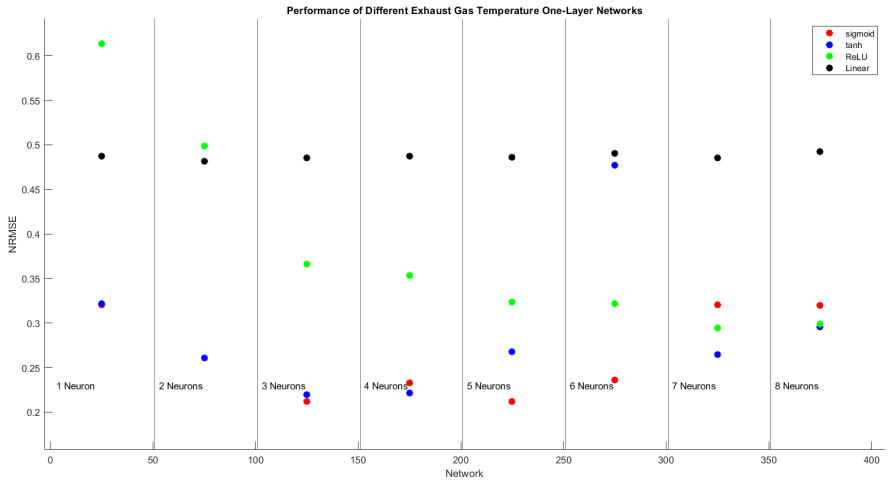


Figure 6.23: The performance of different activation functions in a one hidden layer network.

6.4.1 Validation on Static Data

Figure 6.24 shows that the feed-forward network performs decently on steady-state data. The performance is $R^2 = 0.9678$ and has an average absolute error of 14.6 K.

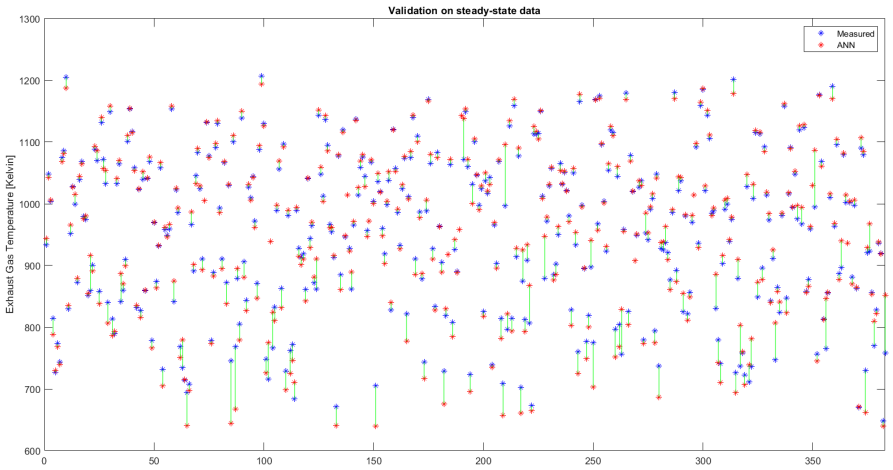


Figure 6.24: Validation of the exhaust gas temperature model on a part-load data set

6.4.2 Validation on Transient Data

The dynamic response of the feed-forward exhaust gas temperature network can be seen in figure 6.25. In order to take the slow sensor dynamics into account, the ANN prediction is filtered with a fourth-order Butterworth low-pass filter with a cut-off time of 12 seconds. The filtered response can be seen in figure 6.26. The performance of on all warm start can be seen in table 6.2. The network can not predict cold starts.

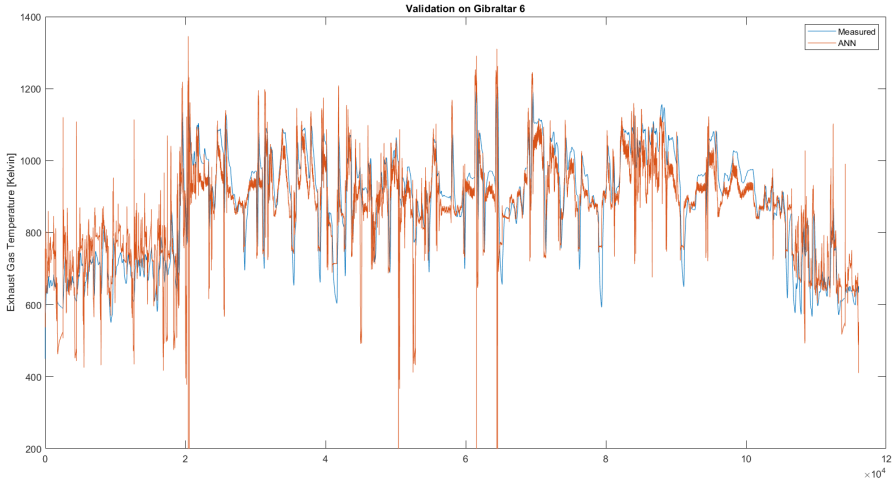


Figure 6.25: Validation on warm start transient data set.

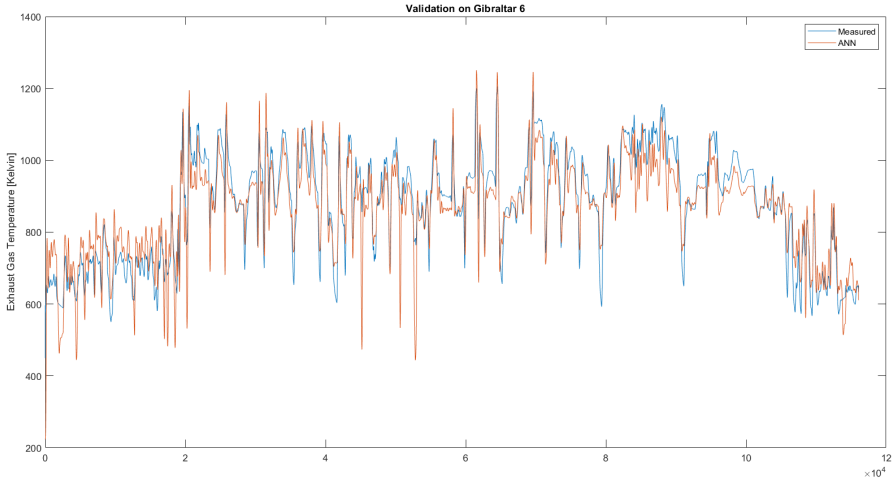


Figure 6.26: Validation on warm start transient data set. Prediction is low-pass filtered

Data set	R^2 Unfiltered	R^2 Filtered	NRMSE Unfiltered	NRMSE Filtered	starting condition
Spain	0.2525	0.59776	0.12935	0.094889	warm
VNT	0.49769	0.6449	0.059986	0.050437	warm
Gibraltar 2	0.58116	0.73326	0.032534	0.025963	warm
Gibraltar 3	0.25129	0.506	0.05935	0.048209	warm
Gibraltar 5	0.553	0.74633	0.03762	0.02834	warm
Gibraltar 6	0.66569	0.81516	0.037355	0.027776	warm
Kiruna 1	0.44759	0.57145	0.14528	0.12796	warm

Table 6.2: Performance of the filtered and unfiltered feed-forward exhaust gas temperature network

6.5 Exhaust Gas Temperature Model: NARX

6.5.1 Network Structure

What needs to be evaluated for the NARX structure, are the number of neurons, number of hidden layers, activation functions, number of output delays and choice of inputs which test will be conducted.

Choice of activation function

A test of different activation functions will be made on a single layer from 1 to 8 neurons. The test will iterate 50 times for each activation function at each neuron count. The median value will be plotted as a dot.

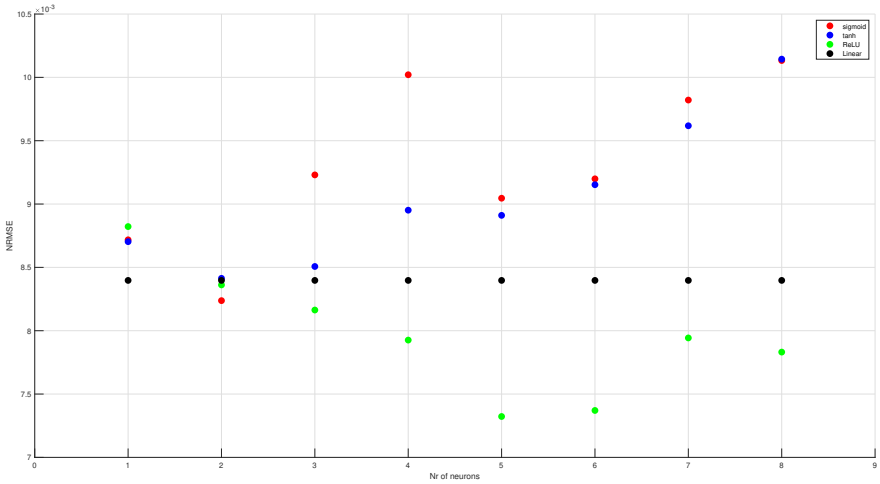


Figure 6.27: Activation function test with median value, training data

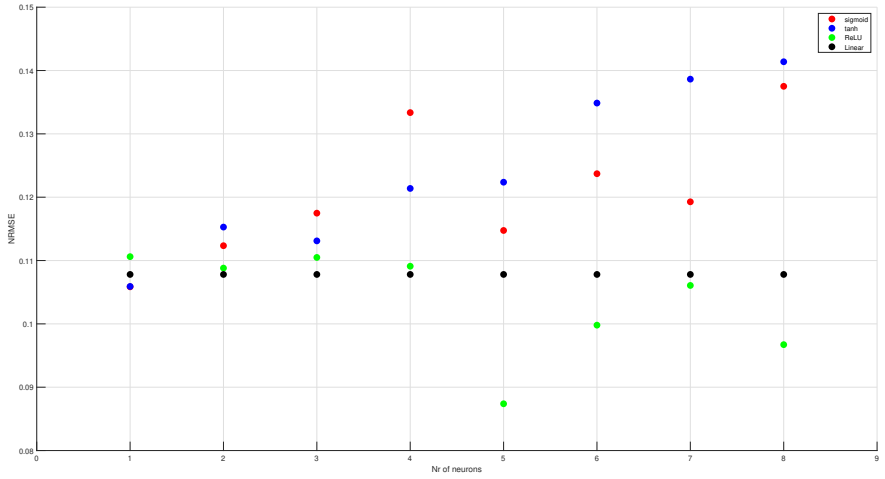


Figure 6.28: Activation function test with median value, validation data Kiruna 2

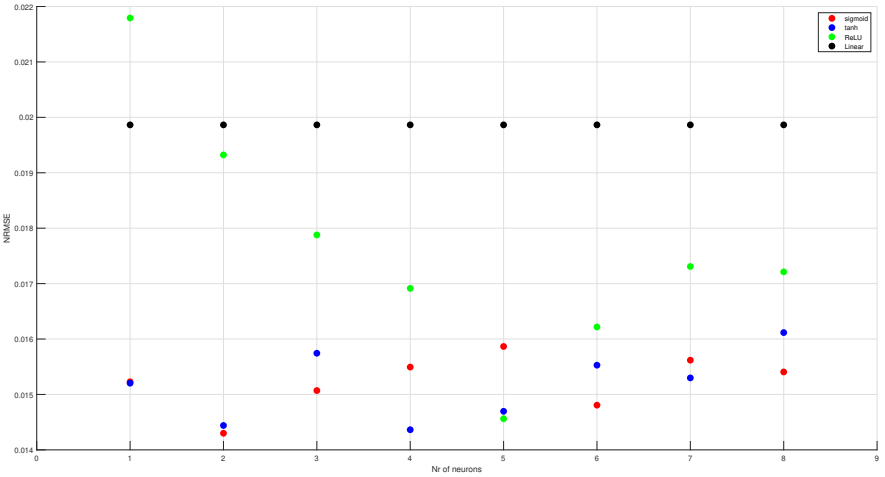


Figure 6.29: Activation function test with median value, validation data Gibraltar 2

From the figures above, it shows that the best performing activation function is ReLU while training, which will be the choice of activation function for the final network and for the continuing tests.

Number of Neurons and hidden layers

To see the appropriate number of neurons to use for the network a test had to be done to verify good performance. This is done by training multiple ANN:s while changing the number of neurons and calculating the NMRSE for each iteration. The vertical lines separate the number of neurons. This results in the following figure.

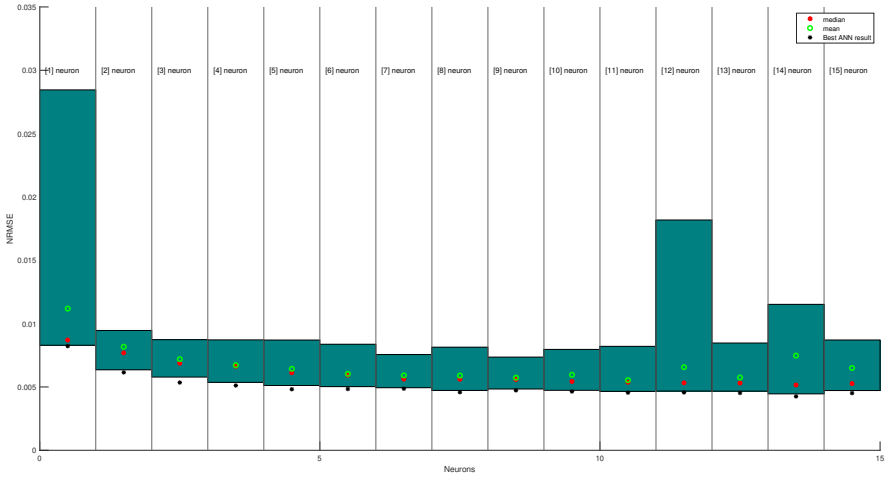


Figure 6.30: Training data NRMSE, 95% confidence interval

The test is set up so that the number of neurons changes from 1 to 15 with all of the training data sets. As can be seen in the graph, the number of neurons has a significant impact on the performance using at least two neurons, but also the results do get slightly better at around five neurons, looking at the median and mean. Having a good number of neurons does bring performance benefits at a cost of calculation performance, anything above one neuron would bring good results, but from the figure above, using five neurons and above should be recommended.

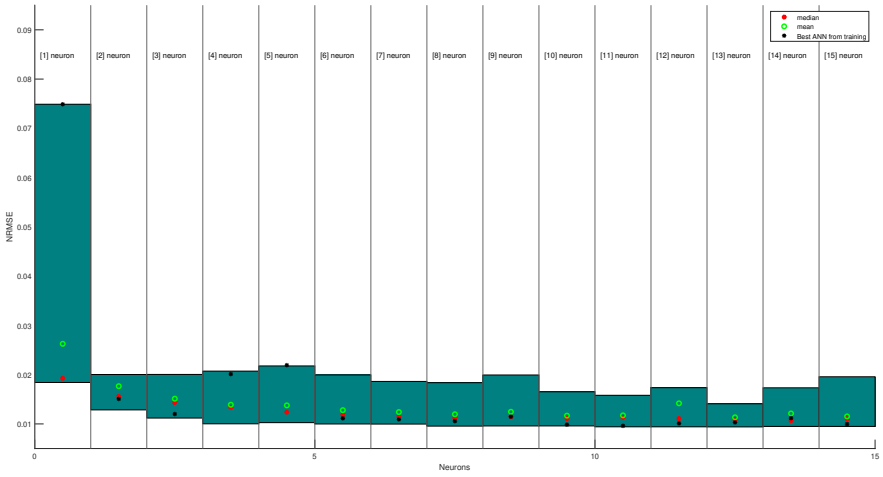


Figure 6.31: Gibraltar 6, validation results 95% confidence interval

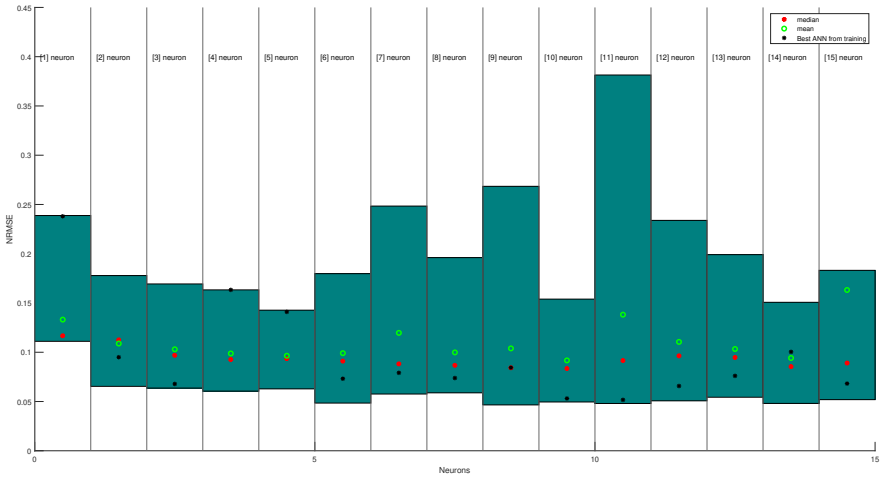


Figure 6.32: Kiruna 2, validation results 95% confidence interval

The graph for Gibraltar 6 validation data is very similar to the training data but is offset on the y-axis overall, which seems logical considering that the trained neural network has never seen the data before but Gibraltar 6 is also easier to handle as it is a warm start. Looking at Kiruna 2 mean, it looks like 11 and 15 neurons are worse than one but this is because of single bad values which worsen the mean. Looking at the median, it is clear that a good choice of neurons is around five and above.

The black dot in the graph above is using the best performing ANN in the training data and sending the validation data Gibraltar 6 into them. The dots hence show that the best performing ANN from training does not mean that the results in the validation data will necessarily be good.

For finding the appropriate amount of hidden layers a test with two hidden layers will be made. this is enough of a test, if the second layer does bring any improvements in the results of the training, then a more in-depth test has to be done with multiple hidden layers but if the second layer does not improve the performance of the ANN, then one hidden layer will be sufficient enough.

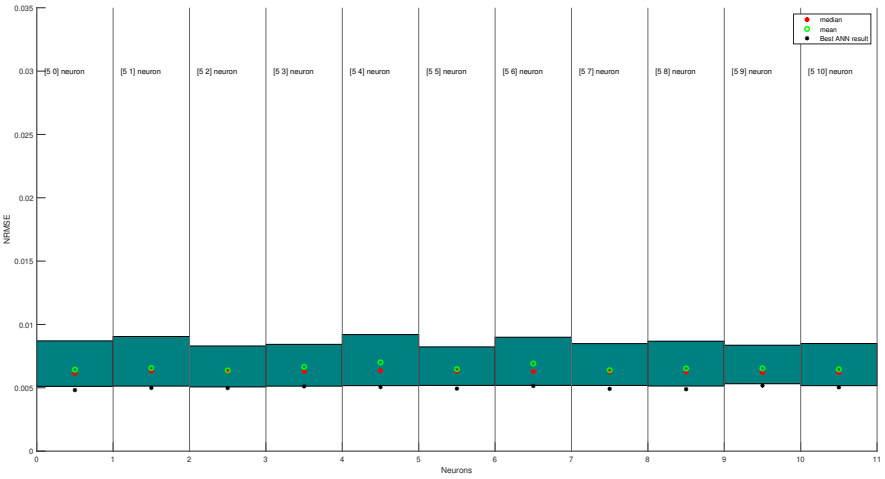


Figure 6.33: Test on number of neurons with two hidden layers, training data

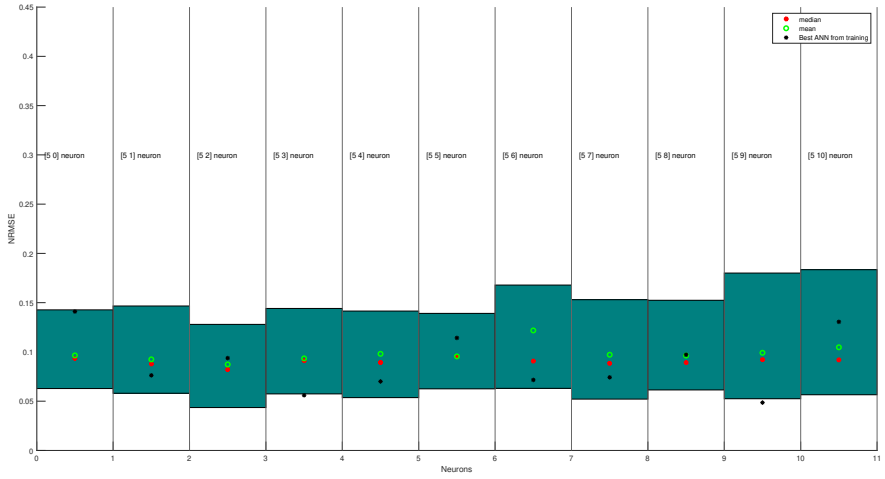


Figure 6.34: Test on number of neurons with two hidden layers, validation data Kiruna 2

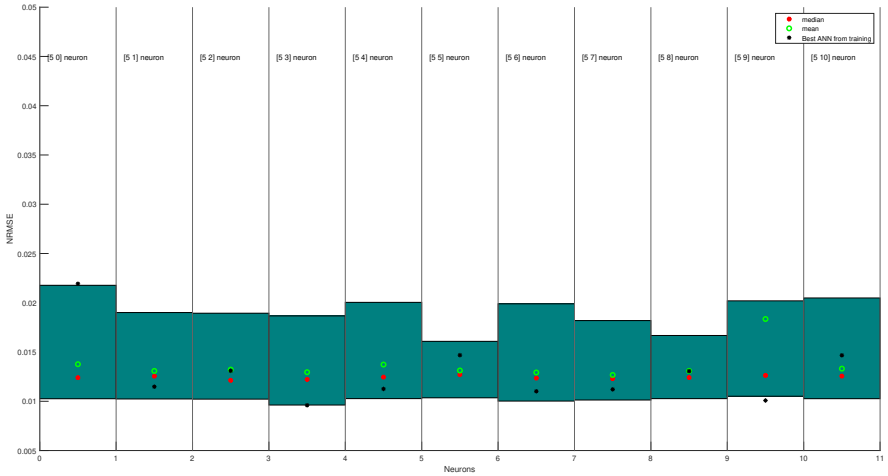


Figure 6.35: Test on number of neurons with two hidden layers, validation data Gibraltar 6

The first hidden layer is set to 5 while a second hidden layer is added, iterating from 0 to 10. The vertical line separates the number of neurons in the second hidden layer, with the first vertical line showing no neurons in the second hidden layer and the last having 10. From the figures above, it shows that the second hidden layer does not affect the performance at all and in turn adds unnecessary computation time and complexity to the model. Specifically looking at the median the value seems consistent throughout the figure, the mean in neuron [5 6] is bad but that is because of one very bad network which offsets the mean. There is also no consistency with the best neural network from the validation figures and so could end up anywhere in the green squares.

Early Stopping Ratio

As previously mentioned a part of the data is used for early stopping regularisation. For a NARX network, it is important that the input sequence is in the correct order. The data is thus divided into training and early stopping validation blocks. The impact of changing the ratio between training data and early stopping data can be seen in Figure 6.36for the training set itself and 6.37 for the independent validation set. Here Gibraltar 3 is used for training and Gibraltar 6 as an independent validation set. Both sets are warm starts. To reduce randomness three networks are trained for each ratio and only the median performance is shown. This test is based on the assumption that at least 50 % of the data is necessary for training.

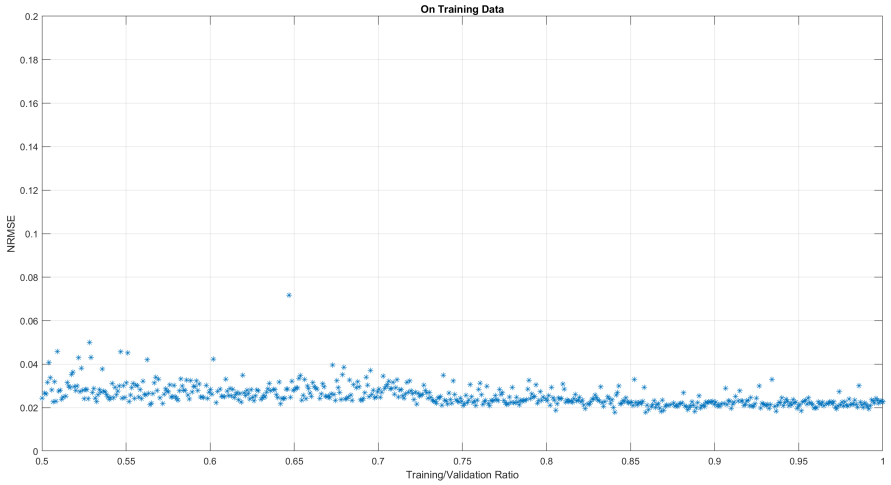


Figure 6.36: Network performance as a function of early stopping ratio. Trained on Gibraltar 3

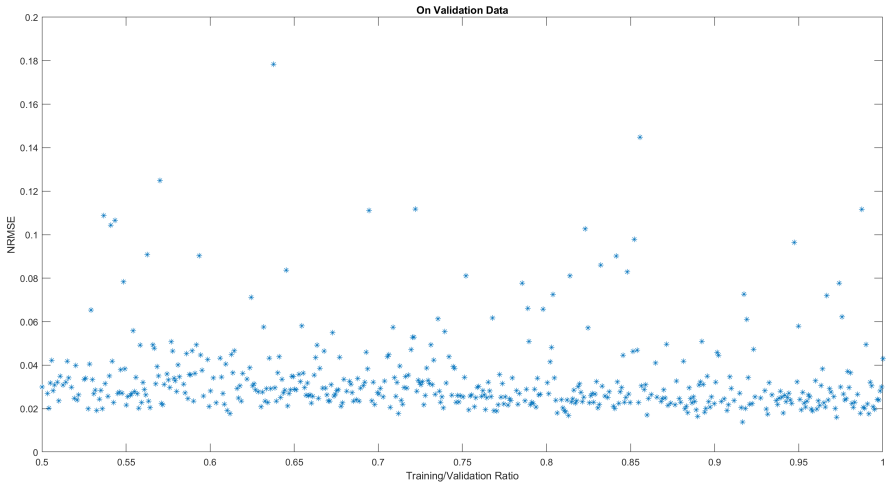


Figure 6.37: Network performance as a function of early stopping ratio. Trained on Gibraltar 3 and evaluated on Gibraltar 6

Feedback delays

Autocorrelation is calculated for every data set to see if there is a definite sample value to use as feedback for NARX.

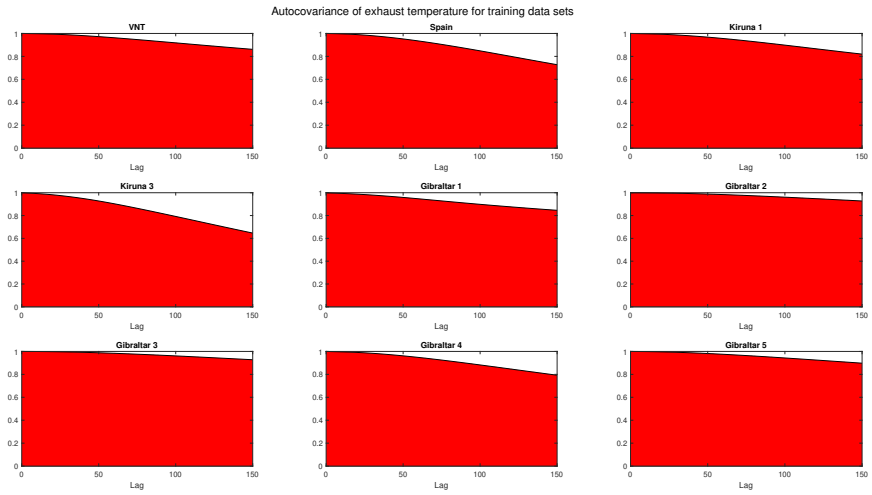


Figure 6.38: Autocorrelation to 150 sample delays

It is clear from the graph above that the performance degrades the higher the sample delay is set. with testing on higher delay counts. Using Kiruna 2 as training data while varying the feedback from 10 to 150 gives us the following result.

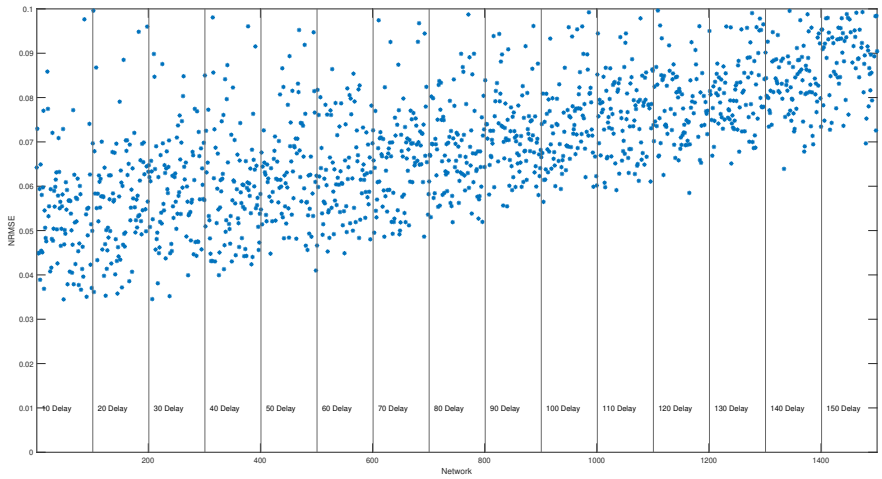


Figure 6.39: Training result of Kiruna two

Which does confirm the performance degradation with higher sample delays. Additionally, the higher delay count introduces oscillation to the behaviour of the results which might

not be ideal. A network has been trained with 75 feedback delay and this resulted in high oscillation. This can be seen in the figure below.

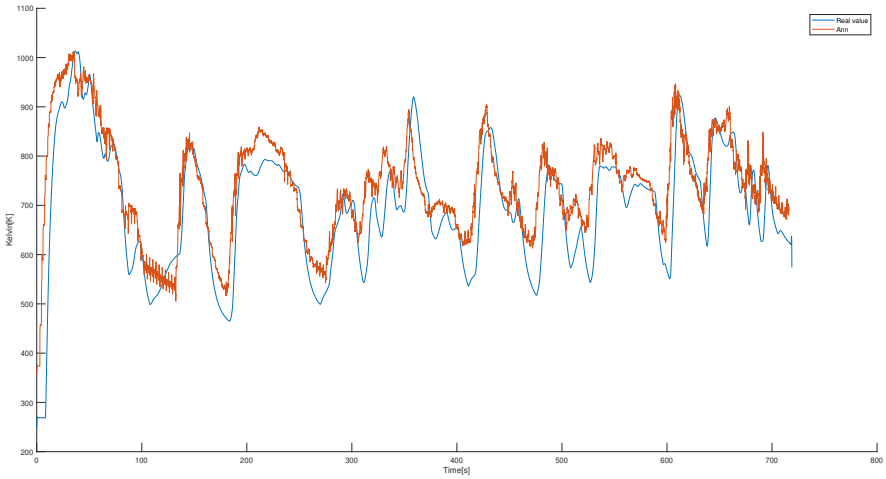


Figure 6.40: Validation on Kiruna two data set, orange: ANN, blue:original

This behaviour has been shown to occur as the feedback delay is increased hence A good choice in delay is concluded to be between 1 to 10 delays to avoid degradation in performance and oscillation.

Looking at using multiple delays, a test has been done whereas the input delay is set to zero and the feedback delay varies from 1 to [1 2 3 .... 10] to see how the added delays affect the performance. This resulted in the following graphs.

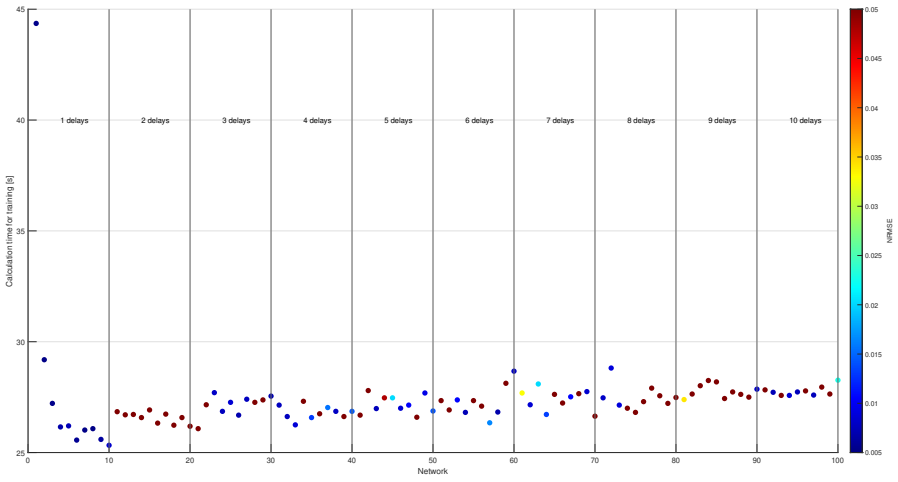


Figure 6.41: Training data on delay test

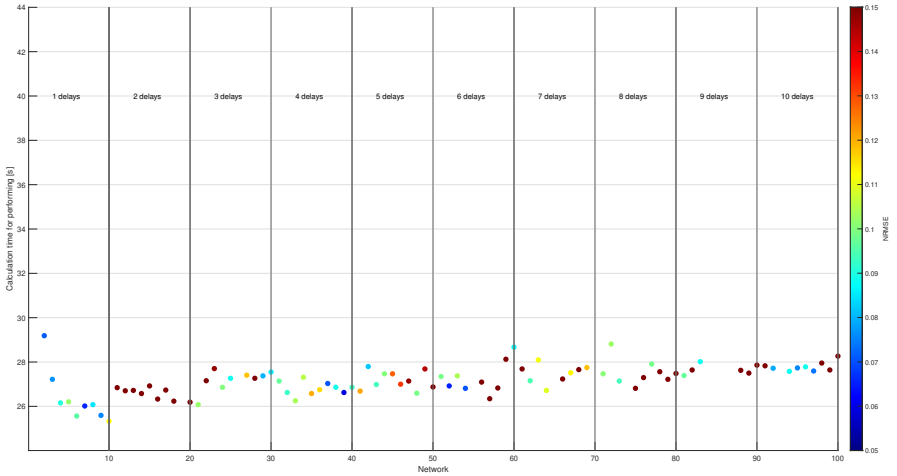


Figure 6.42: Kiruna 2 validation data on delay test

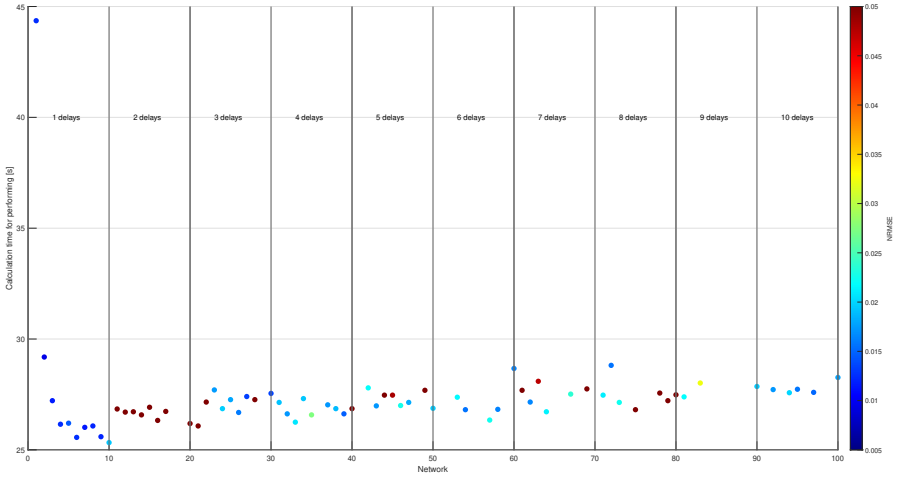


Figure 6.43: Gibraltar 6 validation data on delay test

As can be seen on the colour map the NRMSE whereas a dark blue colour represents the lowest value on NRMSE, one delay performs the best and so only one delay will be used.

Input delays

The cross-covariance for the inputs varies between data sets. Looking at the cross-covariance for two data sets gives the following figures

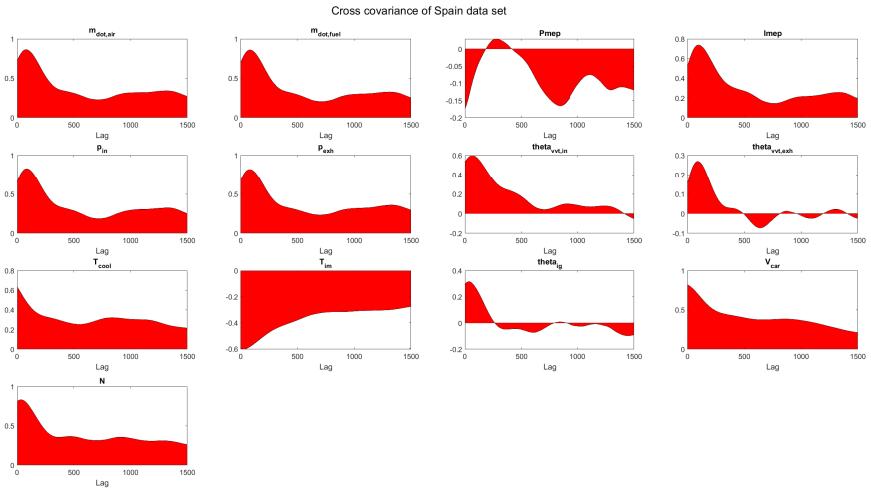


Figure 6.44: cross-covariance between the inputs and the exhaust temperature up to 2 min sample delay

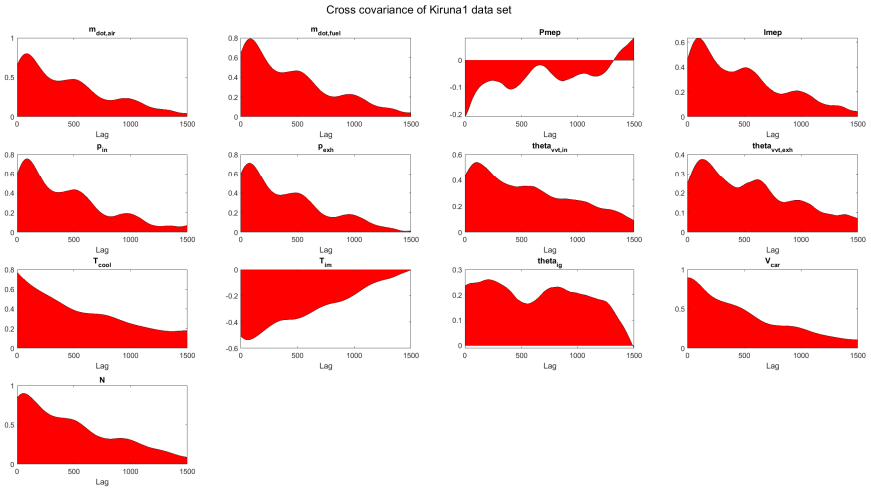


Figure 6.45: cross-covariance between the inputs and the exhaust temperature up to 2 min sample delay

The delay where the correlation is the highest varies for the different inputs, but in most cases, it is the highest at lower sample delays.

To see how the number of delays affects the performance, a test has been done where we incrementally ad a delay with an increased value of one, we start by sending [0 1] and in the last 10 iterations, we reach [0 1 2 3 ... 10] input delays. The colour will represent the NRMSE. This resulted in the following graph.

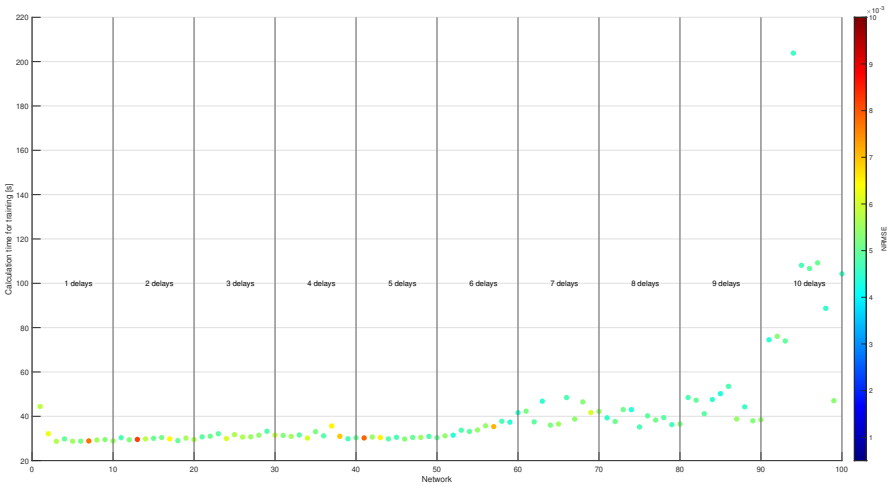


Figure 6.46: Training data input delay test

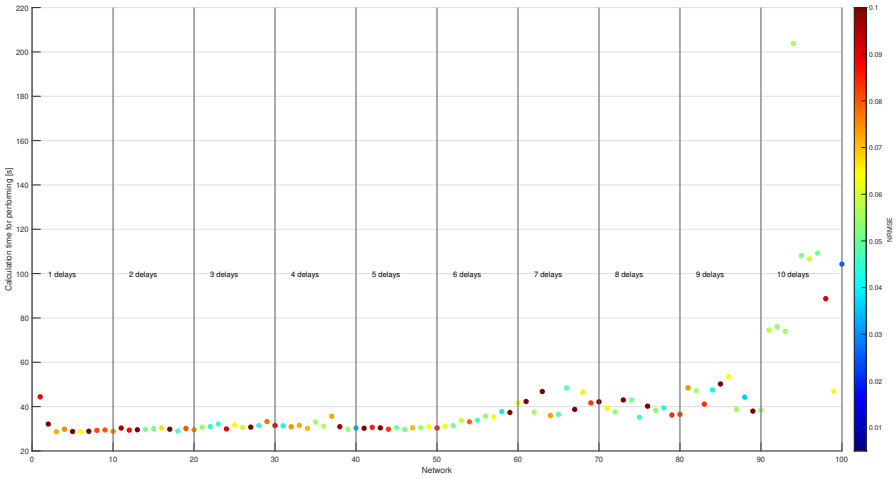


Figure 6.47: Validation data Kiruna 2

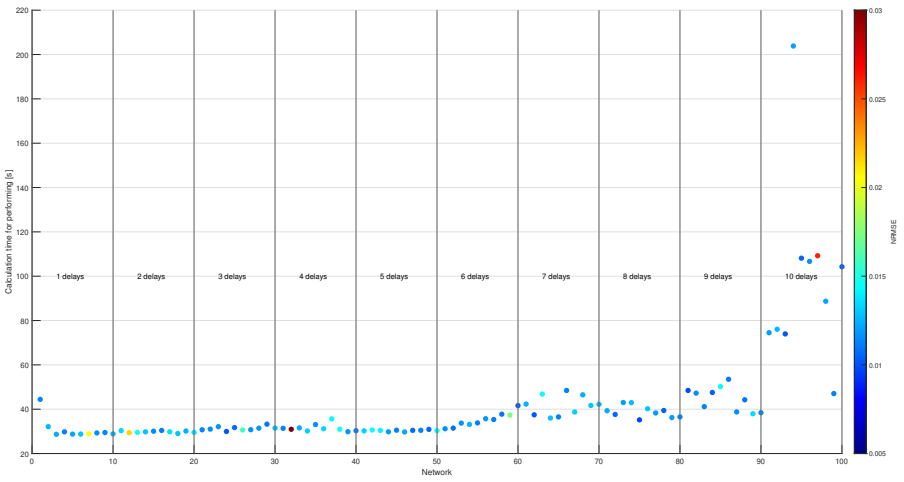


Figure 6.48: Validation data Gibraltar 6

The figures above seem to not affect the validation data much, but for the training data, the result seems to get a bit better the more input delays you add, but this also adds a lot of complexity to the training as you would have to train with 11 times more data with the added delays.

6.5.2 Validation of NARX Model

Trained ANN without cold start

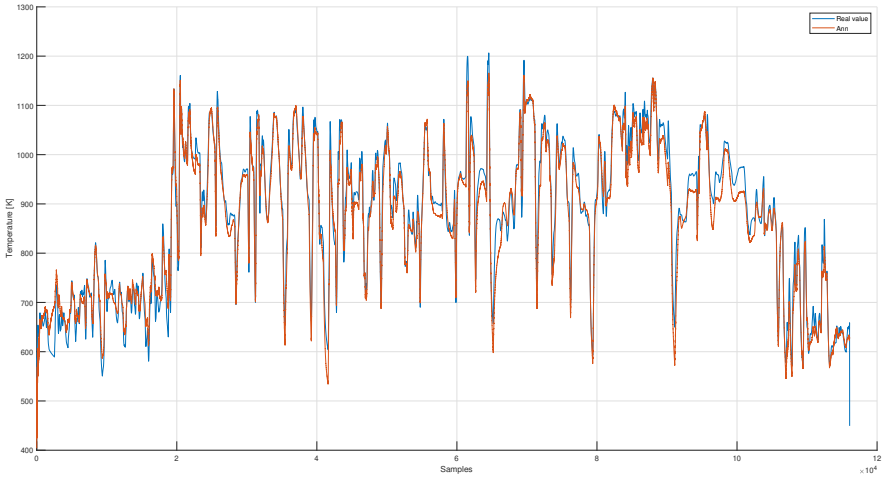


Figure 6.49: Validation on Gibraltar 6 data set, orange: ANN, blue:original

Validation data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Gibraltar6	0.9666	0.0118	0.986	0.0076	warm

Table 6.3: Narx performance on validation data against Aurobays physical model with no cold start.

Training data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Vnt	0.9445	0.0399	0.8890	0.0564	warm
Spain	0.9179	0.0429	0.9611	0.0295	warm
Kiruna1	0.9609	0.0387	0.9688	0.0345	warm
Gibraltar2	0.9697	0.0088	0.9666	0.0092	warm
Gibraltar3	0.9459	0.0160	0.9860	0.0092	warm
Gibraltar5	0.9721	0.0094	0.9761	0.0087	warm

Table 6.4: Narx performance on training data against Aurobays physical model with no cold start.

This test is to check if only training with warm start data improves the network performance for driving with a warm engine. If that is the case then it is possible to have two separate networks, one for cold start and one for when the engine is warm, but comparing the two tables 6.3 and 6.5we can see that training with no cold start does not bring any improvements to the network.

Best fully trained ANN

In the figures and tables below is the best performing ANN including cold start behaviour, which uses 6 neurons and sample five as a feedback delays

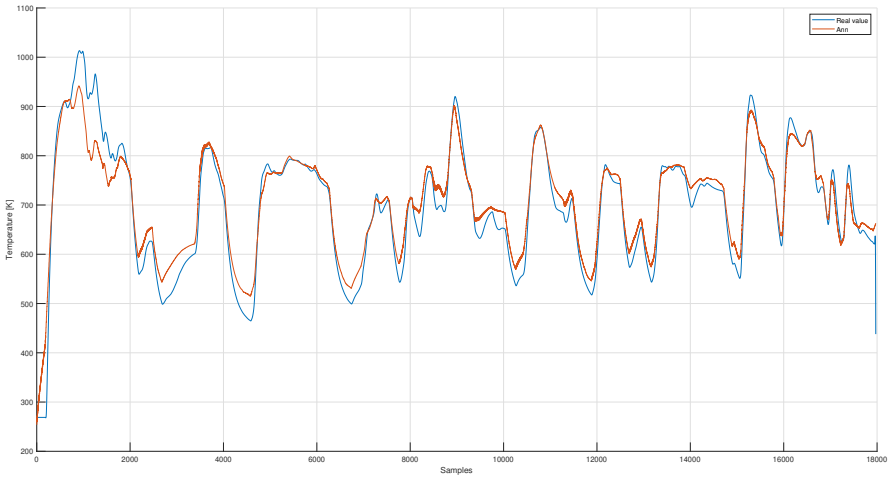


Figure 6.50: Validation on Kiruna 2 data set, orange: ANN, blue:original

Validation data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Kiruna 2	0.9236	0.05234	0.9737	0.03077	cold
Gibraltar 6	0.9700	0.0112	0.9860	0.00764	warm

Table 6.5: Narx performance on validation data against Aurobays physical model.

Training data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Vnt	0.9527	0.0368	0.8890	0.0564	warm
Spain	0.9388	0.0370	0.9611	0.0295	warm
Kiruna1	0.9407	0.0476	0.9688	0.0345	warm
Kiruna3	0.9454	0.0422	0.9710	0.0308	cold
Gibraltar1	0.92243	0.0491	0.6943	0.0976	cold
Gibraltar2	0.9705	0.0086	0.9666	0.0092	warm
Gibraltar3	0.9459	0.0160	0.9860	0.0092	warm
Gibraltar4	0.8887	0.0206	0.7871	0.0285	cold
Gibraltar5	0.9720	0.0094	0.9761	0.0087	warm

Table 6.6: Narx performance on training data against Aurobays physical model.

6.5.3 Validation on Steady-State Data

The transient NARX model is also validated against the steady-state data. This is done by artificially creating a time series for the input of each data point and simulating the temperature model until it has reached steady state. The results for all steady-state data points can be seen in figure 6.51. This corresponds to an average error of around 60 degrees.

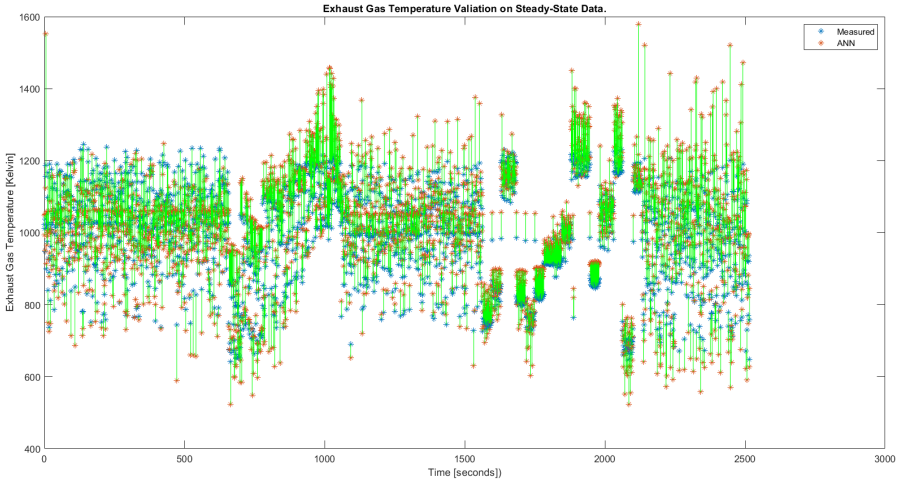


Figure 6.51: NARX model validated against steady-state data. The green line indicates the size of the error for each measurement.

6.5.4 Full ANN Simulation

When fully simulating the exhaust gas temperature without measuring all signals the static models need to be used as well. The feed-forward neural networks for the cylinder flow, $IMEP_g$ and PMEP are used to simulate inputs for the exhaust gas temperature network. It should be noted that the $IMEP_g$ and PMEP networks require the cylinder flow output as inputs and that the cylinder flow network requires the exhaust gas temperature output as an input. The simulation, therefore, includes feedback which will make the errors of all models accumulate. The simulation is implemented in Simulink and does not include the deep learning toolbox. This is done to increase performance. The results can be seen in table 6.7. The figreffig:FullANN shows a graph of the estimated temperature for the Kiruna 2 data set.

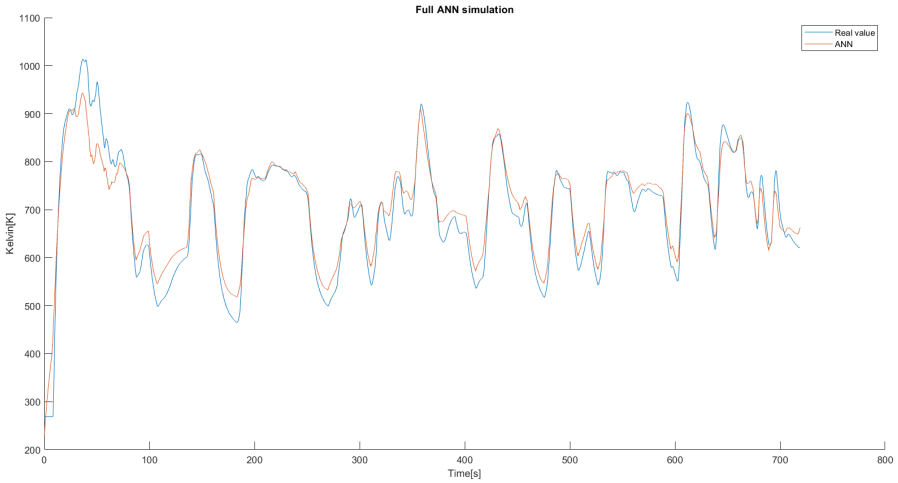


Figure 6.52: Validation on Kiruna 2 data set using ANN-models for cylinder flow, $IMEP_g$ and PMEP inputs.

Data set	R^2 Full ANN	R^2 Measured Inputs	NRMSE Full ANN	NRMSE Measured Inputs	starting condition
Kiruna 2	0.9262	0.9236	0.0515	0.05234	cold
Gibraltar 6	0.9675	0.9700	0.0117	0.0112	warm

Table 6.7: Performance of a full ANN simulation of the exhaust gas temperature. Data from table **??** is repeated here for readability.

6.6 Exhaust Gas Temperature Model: LSTM

Running and training these types of networks are very time-consuming so the structure here was chosen by doing many different training runs with different setups and choosing the best ANN.

This ANN uses gradient descent optimization method As this method does not have the Levenberg-Marquardt method and the Adams method brought no improvements when testing. The setup also uses 15 neurons.

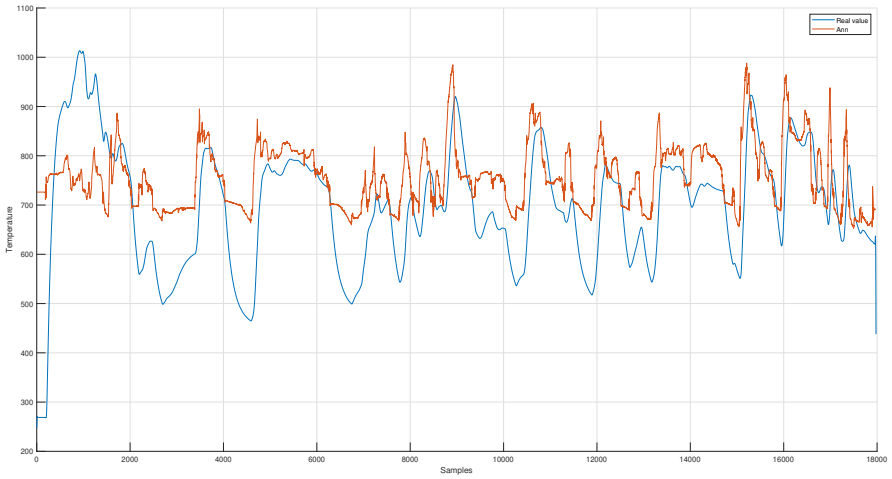


Figure 6.53: Validation on Cold 2 data set, orange: ANN, blue:original

Validation data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Kiruna 2	0.0575	0.1842	0.9737	0.03077	cold
Gibraltar 6	0.8109	0.0281	0.9860	0.00764	warm

Table 6.8: LSTM performance on validation data against Aurobays physical model.

Training data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Vnt	0.7339	0.0873	0.8890	0.0564	warm
Spain	0.6467	0.0890	0.9611	0.0295	warm
Kiruna1	0.6569	0.1146	0.9688	0.0345	warm
Kiruna3	0.2377	0.1581	0.9710	0.0308	cold
Gibraltar1	0.0605	0.1711	0.6943	0.0976	cold
Gibraltar2	0.7108	0.0270	0.9666	0.0092	warm
Gibraltar3	0.7419	0.0349	0.9860	0.0092	warm
Gibraltar4	0.0222	0.0611	0.7871	0.0285	cold
Gibraltar5	0.7541	0.0279	0.9761	0.0087	warm

Table 6.9: LSTM performance on training data against Aurobays physical model.

6.7 Exhaust Gas Temperature Model: GRU

The Gated recurrent network was trained similarly to the LSTM

This ANN uses gradient descent optimization method As this method does not have the Levenberg marquardt method and the Adams method brought no improvements when testing. The setup also uses 15 neurons.

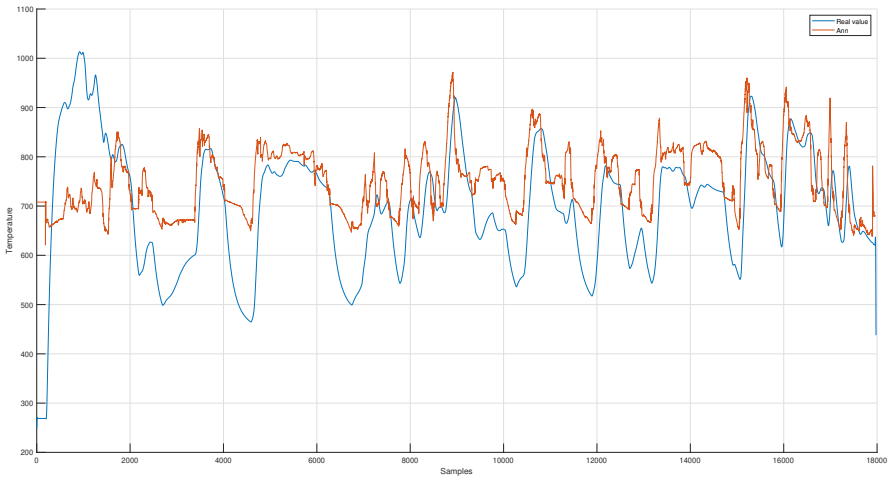


Figure 6.54: Validation on Cold two data set, orange: ANN, blue:original

Validation data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Kiruna 2	0.0627	0.1837	0.9737	0.03077	cold
Gibraltar 6	0.8196	0.0274	0.9860	0.00764	warm

Table 6.10: GRU performance on validation data against Aurobays physical model.

Training data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Vnt	0.7265	0.0886	0.8890	0.0564	warm
Spain	0.6959	0.0825	0.9611	0.0295	warm
Kiruna1	0.6508	0.1156	0.9688	0.0345	warm
Kiruna3	0.1662	0.1653	0.9710	0.0308	cold
Gibraltar1	-0.0795	0.1834	0.6943	0.0976	cold
Gibraltar2	0.6751	0.0287	0.9666	0.0092	warm
Gibraltar3	0.7425	0.0348	0.9860	0.0092	warm
Gibraltar4	-0.0969	0.0648	0.7871	0.0285	cold
Gibraltar5	0.7478	0.0283	0.9761	0.0087	warm

Table 6.11: GRU performance on training data against Aurobays physical model.

6.8 Exhaust Gas Temperature Model: LRN

Activation function

This test is made similarly to the NARX test in the NARX result section. The neuron setup is [6 x] where x is varied from 1 to 8 and each activation function is tested for 50 iterations before adding a neuron. The plot will then be presented as the median for the specific activation function at the neuron.

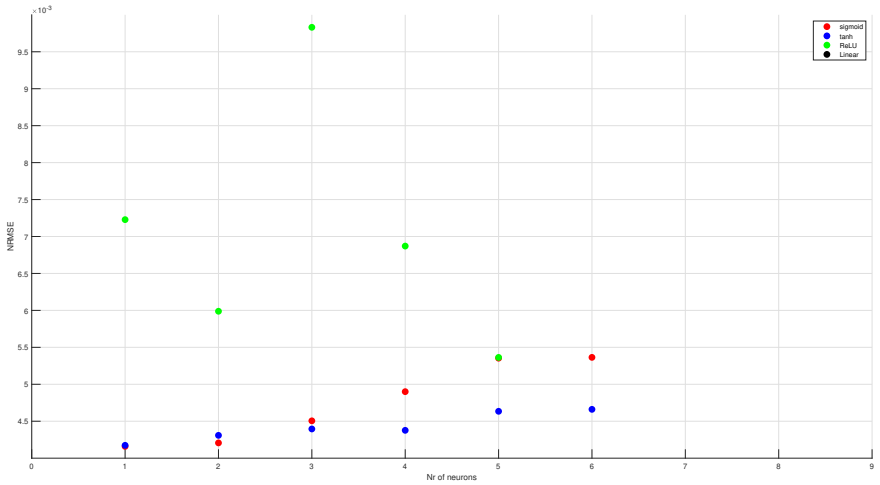


Figure 6.55: Median from training data

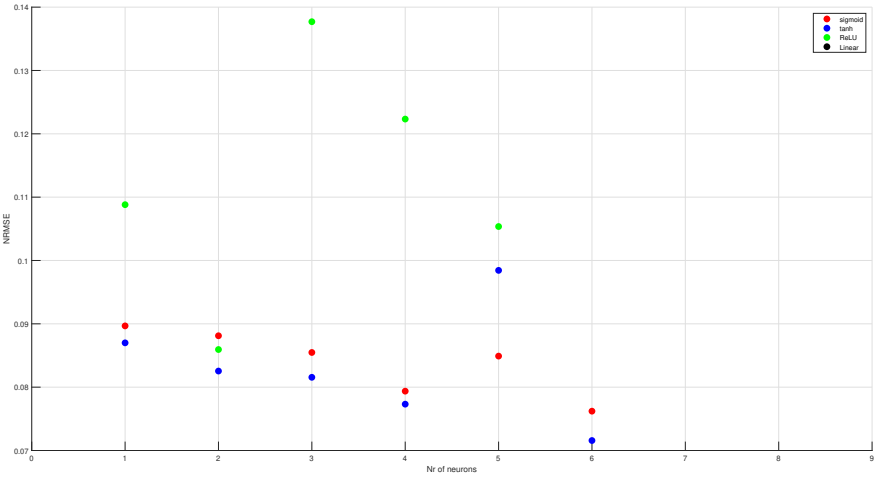


Figure 6.56: Median Result with the validation data set Kiruna 2

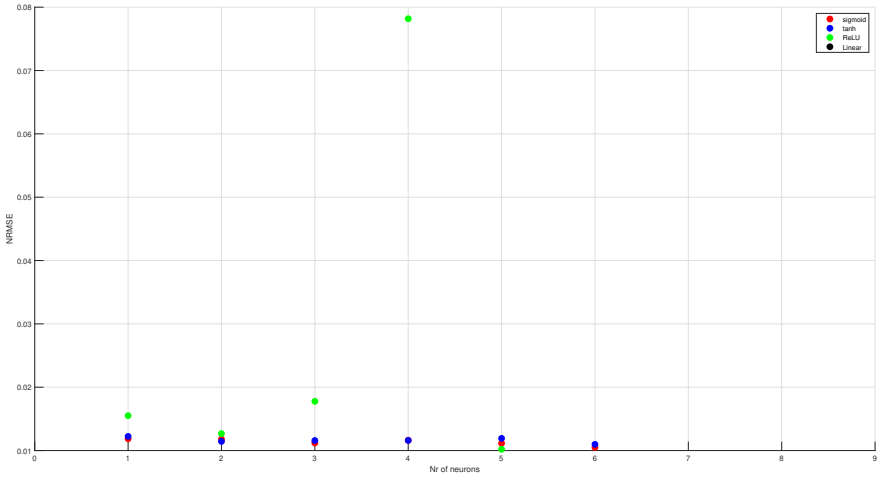


Figure 6.57: MedianResult with the validation data set Gibraltar 6

Looking at figure 6.55 it seems that the ReLU does not perform the best but the y-axis is also very small and is 10^{-3}, so using ReLu is still a viable option to get a good performing ANN while also minimizing the calculation time as it takes longer to train an ANN with sigmoid and tanh. Something noticeable here in the figures is also that some of the values are missing in the higher neuron count which is because of the ANN not being able to find a good solution which returns "not a number" as a result.

Number of Neurons and hidden layers

Similarly to NARX, there needs to be a test on the number of neurons. The difference now is that the test is conducted with two hidden layers as a single hidden layer recurrent network is similar to a NARX network.

The test locks the first hidden layer to 5 and a second hidden layer varying the neuron count, with 100 training iterations before adding a neuron until it reaches 15 neurons in the second hidden layer. This resulted in the following graphs

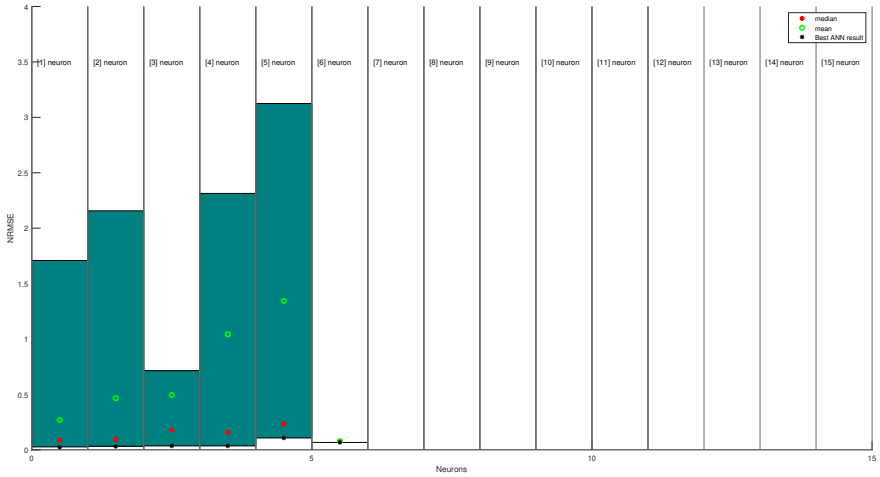


Figure 6.58: Result with training data

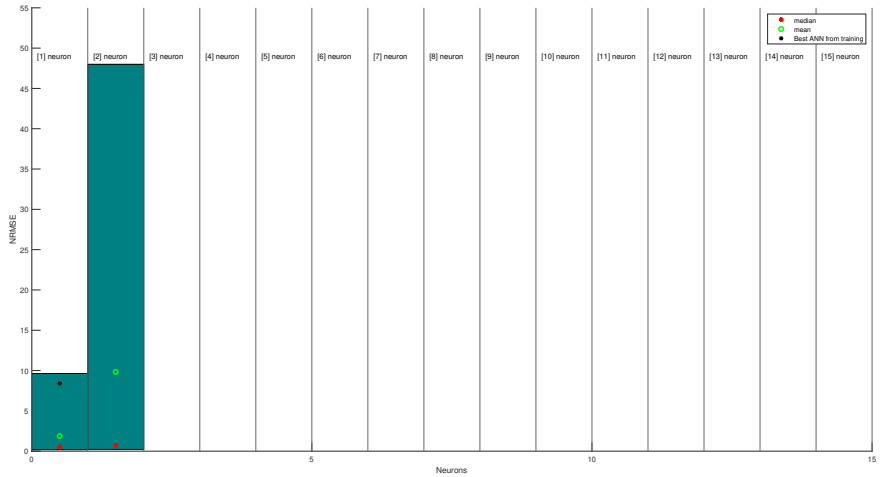


Figure 6.59: Result with the validation data Kiruna 2

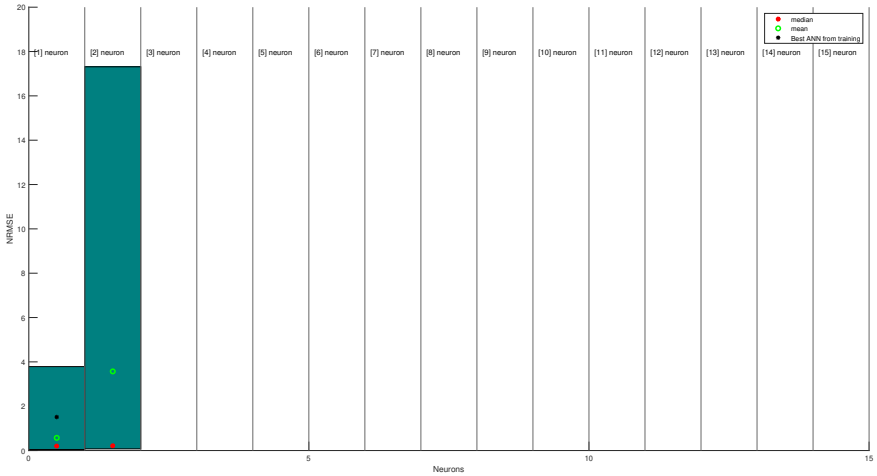


Figure 6.60: Result with the validation data Gibraltar 6

There are a lot of data points that are missing. This is because of the ANN not finding a good result at the higher neuron count for the second hidden layer and so return a Nan or infinite as a result. Finally, the choice of neurons here should be similar to NARX with the first hidden layer whereas the performance is sufficient with a neuron count above 5 and for the second hidden layer 1 neuron is sufficient.

Feedback delay test

For the Layer recurrent network, a test will be done with the feedback delay. This test is done with the initial feedback setup of [1] and in delay 10 it is set up as [1 2 3 ... 10] to see how the number of delays affects the performance of the ANN. The number of neurons used for this test is [6 1] and the activation function used is ReLU. This resulted in the following graphs

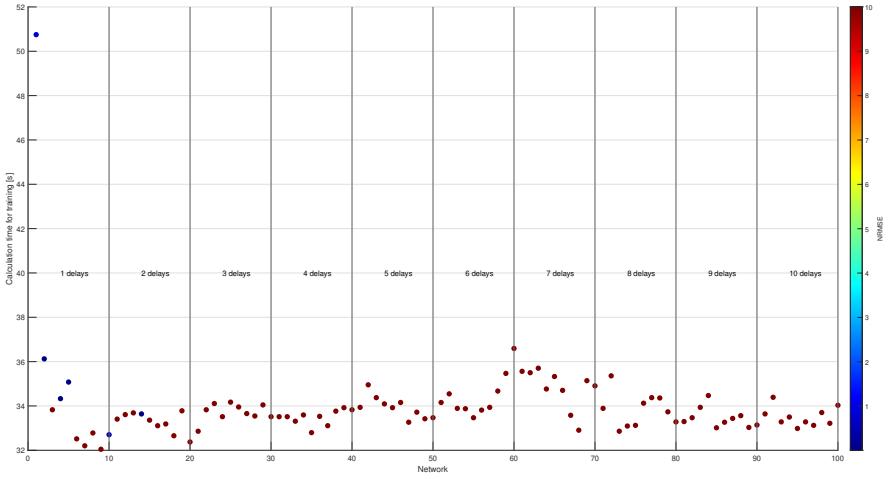


Figure 6.61: Feedback test with training data

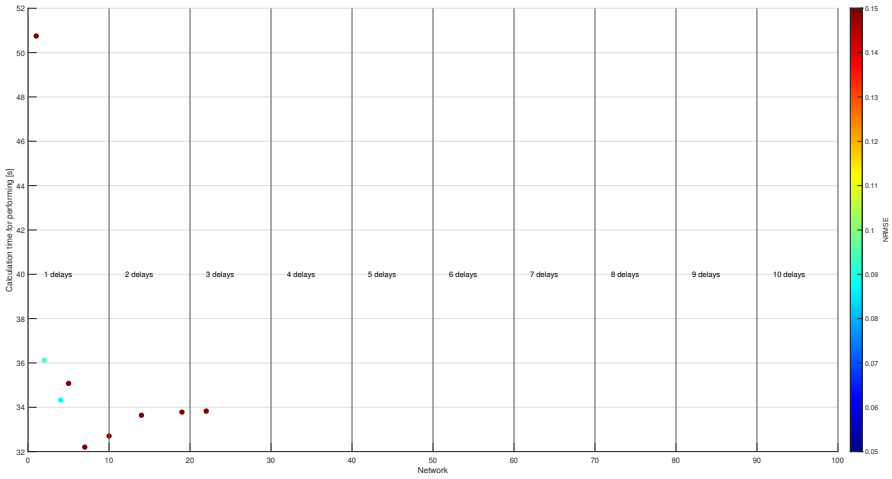


Figure 6.62: Feedback test with validation data Kiruna 2

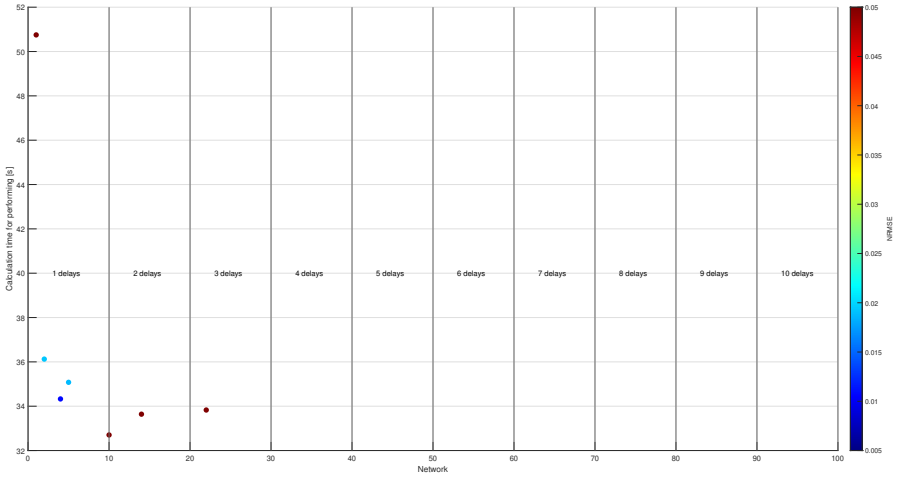


Figure 6.63: Feedback test with validation data Gibraltar 6

As can be seen on the graphs above, the performance degrades with more delays with some of the neural network dots having an infinite NRMSE, with the validation data, these dots do not appear because the value returned is Nan as the ANN does not know what to do with the validation data, which is why there are no dots above 3 delays. So only one delay is used and that delay will be set to 5.

Layer Recurrent Network

The best LRN network is presented below. The network has five in feedback delay and the number of neurons is set to [6 1].

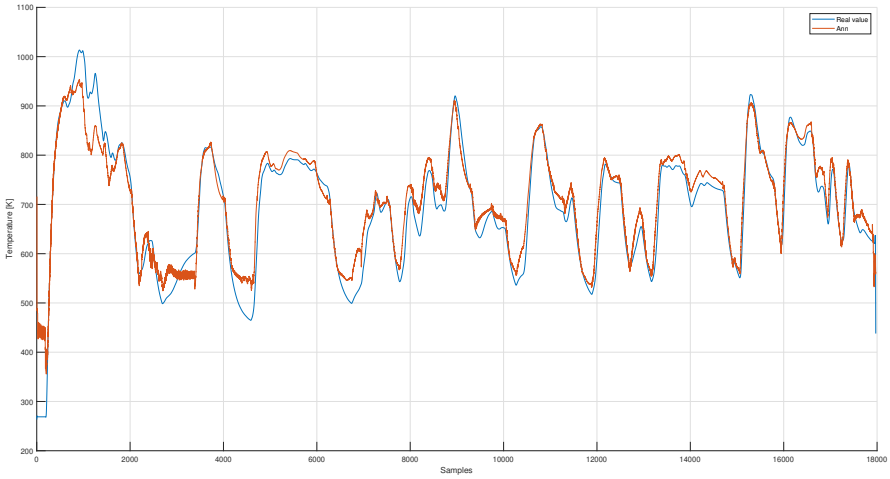


Figure 6.64: Validation on Kiruna 2 data set, orange: ANN, blue:original

Validation data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Kiruna 2	0.9039	0.0587	0.9737	0.03077	cold
Gibraltar 6	0.9739	0.0104	0.9860	0.00764	warm

Table 6.12: Layer Recurrent performance on validation data against Aurobays physical model.

Training data	R^2	NRMSE	R^2 Aurobay	NRMSE Aurobay	Starting condition
Vnt	0.9749	0.0268	0.8890	0.0564	warm
Spain	0.9643	0.0282	0.9611	0.0295	warm
Kiruna1	0.9735	0.0318	0.9688	0.0345	warm
Kiruna3	0.9444	0.0426	0.9710	0.0308	cold
Gibraltar1	0.9201	0.0498	0.6943	0.0976	cold
Gibraltar2	0.9842	0.0063	0.9666	0.0092	warm
Gibraltar3	0.9596	0.0138	0.9860	0.0092	warm
Gibraltar4	0.8827	0.0212	0.7871	0.0285	cold
Gibraltar5	0.9724	0.0093	0.9761	0.0087	warm

Table 6.13: Layer Recurrent performance on training data against Aurobays physical model.

6.9 Summation of Results

A summation of the performance for the different network structures can be seen in table 6.14. The performance of Aurobay's current empirical model is also included for comparison.

Dataset	Model	R^2	NRMSE	Starting condition
Kiruna 2	Aurobay	0.9737	0.03077	cold
Gibraltar 6	Aurobay	0.9860	0.00764	warm
Kiruna 2	NARX	0.9236	0.0523	cold
Gibraltar 6	NARX	0.9700	0.0112	warm
Kiruna 2	LRN	0.9039	0.0587	cold
Gibraltar 6	LRN	0.9739	0.0104	warm
Kiruna 2	LSTM	0.0575	0.1842	cold
Gibraltar 6	LSTM	0.8109	0.0281	warm
Kiruna 2	GRU	0.0627	0.1837	cold
Gibraltar 6	GRU	0.8196	0.0274	warm

Table 6.14: Summation of model performance on validation data.

6.9.1 Implementation in the electronic control unit

The models mentioned in the section "Best fully trained ANN" with NARX, alongside the static models in the sections PMEP IMEP$_g$ and cylidner flow were made in Simulink, flashed into the electronic control unit of the engine and logged. The reason for this is to see how well the ANN works with new data in real-time. The data from the cylinder flow and the exhaust gas temperature are presented below in the following figure.

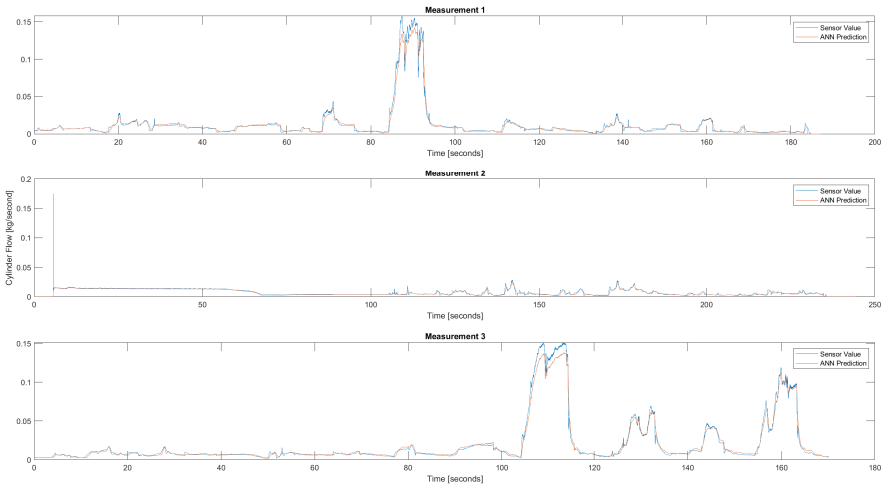


Figure 6.65: Results of the mass flow into the engine, in real-time

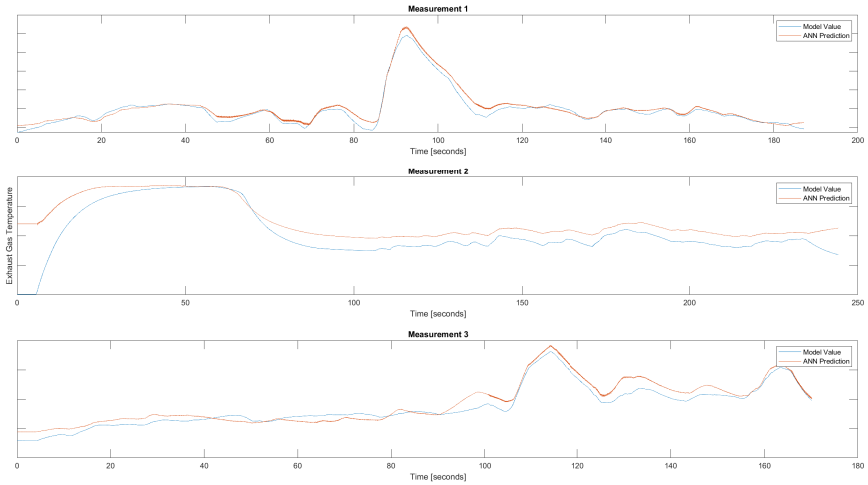


Figure 6.66: Results of the exhaust gas temperature, in real-time

The exhaust gas temperature sensor was not logged and so it is only comparing the ANN to Aurobays empirical model.

7 Discussion

7.1 Results

7.1.1 Cylinder Flow

It is easy to estimate very accurate single hidden layer feed-forward neural networks to model the cylinder flow. This can be done using only a few neurons using the most common activation functions. Due to the cylinder flow being measured during the transient data sets as well as the steady-state data sets there is an abundance of data that can be used to train with. This also increases the performance of the model. The cylinder flow model performs well on both cold and warm starts. The good performance of the model even on cold starts is somewhat surprising, however, the transient data used for the training does include cold starts as well. By only training the network on steady-state data the cold start performance would decrease.

7.1.2 PMEP and IMEP$_g$

Since the IMEP$_g$ is not measured during the transient data sets the model is trained only on steady-state data. This results in predictions with extreme transient spikes. The IMEP$_g$ is validated against Aurobays model and, except for the spikes, follows the model well. It is possible that using a lowpass filter in conjunction with the steady-state trained ANN could improve the performance.

The PMEP model has a much more reasonable behaviour than the IMEP$_g$ model and follows Aurobays prediction well. There are however slight deviations between the two when looking at cold start data sets.

7.1.3 NARX Temperature Model

The network structure test and evaluations have given a good indication as to how the structure should be set up. Due to the variations in the performance for each training process, it is still necessary to train multiple networks to find one that performs well. It can also be seen in figure 6.32 that the variance in the performance is larger for the validation set than for the

training set. The ANN presented in the result section is very good and close to the behaviour of the sensor.

Validating the NARX model on the steady-state data does not give very accurate predictions. This could be either because of how the measurements are taken, since the steady-state data consists of engine mappings and does not include an actual vehicle in the test, which could affect the engine in different ways, such as with convection. An other explanation could be that the network is simply not trained on data that is eventually stabilising at steady-state, thus the model only captures faster dynamics. The average error of 60 degrees could be considered significant if the model is intended to be used for controlling the engine at steady-state. This is an unlikely scenario, however. In reality, the engine operating point changes a lot throughout the drive cycle and the temperature rarely, if ever, settles at steady-state.

7.1.4 Layer-Recurrent Network Temperature Model

For the setting of having a single hidden layer, an LRN is very similar to a NARX network lacking an input delay.

The best LRN has very similar results to the best NARX network when looking at the evaluation values. The variation of the results from training implies that both NARX and LRN can outperform each other depending on luck in training but not significantly. The difference here is that an LRN is much more unstable with many networks trained converging to infinity or returns NAN as a result which can be seen by looking at for example 6.58 where values are missing at higher neuron count. It is concluded then that it would be fine to use either NARX or Layer-Recurrent network as an ANN for modelling purposes.

7.1.5 LSTM and gated recurrent network

Both the LSTM and the gated recurrent networks give similar results. When it comes to the structure of these networks. What has been varied in these tests were the layer structure, activation functions, having multiple LSTM/Gated recurrent blocks, and also the learning rate and drop rate. All of these factors had minimal effect on the result, with some of the changes worsening the performance.

7.1.6 NARX training with warm start data

The ANN trained with warm start data did not bring any improvements to the warm start data, initially, the idéa here was to see if it is a viable option to train a network for a cold start scenario and a warm start scenario as an alternative instead of a single ANN handling both, but by comparing the results from the tables 6.3 6.4 to 6.5 6.6 we can see that a neural network only trained for warm start scenarios does not bring any performance benefits and so the idéa will not be implemented.

7.1.7 Full ANN

The initial expectation was that when using the feed-forward networks as input to the dynamic temperature network, as well as using the dynamic temperature network as input to the feed-forward networks the small errors of all networks would start to accumulate over time. This does not seem to be the case in the decrease in performance is marginal.

7.1.8 Implementation in the electronic control uni

As can be seen in figure 6.65 and 6.66 the results are excellent and as such, truly validates the performance of the ANN as it has been tested in real-time in a vehicle and proves the performance of the ANN could be used for engine modelling purposes.

7.2 Method

7.2.1 Network Structure Tests

The tests are limited in the sense that some parameters had to be fixed to conduct the test of a chosen parameter. This is to decrease computation time as these tests are limited to the computers that are being used. A more thorough way of conducting these tests is to vary all of the existing parameters in different combinations, but this requires a very strong computer to be feasible. The method used might miss certain optimal combinations of parameters, and interdependence of parameters.

7.2.2 Cylinder flow, IMEP$_g$ and PMEP Models

Training the cylinder flow model on transient data from an actual vehicle improves the performance compared to simply training the data from an engine test cell. Training the IMEP$_g$ and PMEP models in a similar way might increase the performance of these models as well.

7.2.3 Network structure of Exhaust Gas Temperature Model

The performance of the network does not seem to depend on the number of neurons used in the network, as long as at least five neurons are used. The most efficient way to produce an accurate network seems to be to use around six neurons and then choose the best performing network out of several attempts. This is a computationally heavy approach.

The most common reason for deeming the training finished is the early stopping regularisation. It is therefore surprising that the ratio between the data used for training the network and the early stopping regularisation does not seem to have a huge impact on the performance of the network.

Model Inputs

The ambient temperature could have a large impact on the exhaust gas temperature. The reason for not including the ambient temperature as an input is that the temperature networks rely on time-series data. Throughout each data set, the ambient temperature is relatively constant, however, it does change significantly between data sets. For the feed-forward network, the data is collected in a test cell. This means that the ambient temperature is constant for the steady-state data as well.

The approach was to use as many reasonable input signals as possible and let the network training decide if the inputs were to be used or not. Instead, the input-output cross-correlation could be investigated to only choose signals with a high correlation to the output.

7.2.4 NRMSE and R^2

The idea of using a normalised RMSE was to have the values be independent of which feature was evaluated. Instead, what could be considered good changes for each data set. Solely relying on the R^2 metric can also be slightly misleading. It is almost a necessity to compare plots of the model response with measured data when evaluating the performance.

7.3 Future Work

It is possible that better hyperparameter combinations can be found by testing multiple structure parameters such as the network size, delays et cetera at the same time. It could also be of interest to instead train the network on data from a sensor with faster dynamics to give

a more accurate prediction of the temperature transients. In order to achieve better predictions during cold start, a separate network could be used in conjunction with an indicator of whether the engine is warm or cold. Such an indicator could for example be the engine coolant temperature.

In order to guarantee robustness more validation should be done, including an analysis of how changes in specific inputs affect the network predictions. How the network interacts with a control system that is dependent on the network predictions should also be analysed.

In this thesis, regularisation of the networks is done mainly through early stopping. Different regularisation methods could be attempted and compared as well.

7.4 The work in a wider context

A neural network in the context of engine development has the potential to speed up development time to release a product more environmentally friendly and cheaper, as the reduced development cost could be reflected in the consumer price. With the potential of neural networks, it might be able to get a better model than a physical model which might normally be hard to design. The negative aspect of neural networks is that it is hard to prove the robustness of the model without a lot of testing. This can be dangerous for a driver if the engine starts acting strange during driving, because of a certain driving scenario that the model hasn't trained for.

8 Conclusion

The research questions presented in the thesis can be concluded in the following way:

Is artificial neural network modelling a viable method for estimating the exhaust gas temperature and How does the performance of the artificial neural network model compare to existing empirical models?

In consideration to the results, it can be presumed that the resulting ANN is satisfactory enough as it is very good for NARX networks and LRN. This can be seen by looking at the tables6.13,6.12, 6.6 and 6.5 which performs very close to the physical models with some of the training data performing better than the physical model. Additionally, with the result from the electronic control unit implementation, seen in figure 6.66, the ANN behaves similarly to the physical model, you could then conclude that the performance can be deemed satisfactory. This is not the case for LSTM or GRU networks as their performance is subpar in comparison to NARX, LRN and the physical model which can be seen in the two tables 6.10 and 6.8.

How can the structure of a neural network model of exhaust gas temperature be optimised with respect to the results and computational effort?

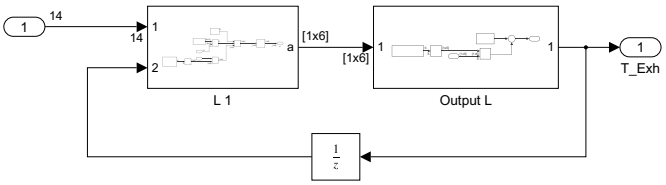
A small network is enough to give a model with decent accuracy, both using a layer recurrent network and a NARX network for the temperature and when using the feedforward networks for the cylinder flow, $IMEP_g$ and PMEP. A compromise between performance and computational effort does not need to be made.

Which network structure is best suited for modelling the exhaust gas temperature?

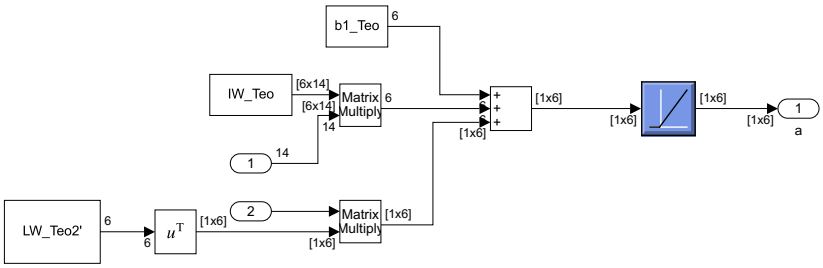
Based on the result, the layer-recurrent network and NARX handles modelling the exhaust temperature very well but LSTM and GRU do not give any good results in comparison to the other two network types.

Appendix

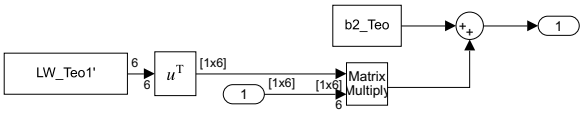
Simulink model of NARX network

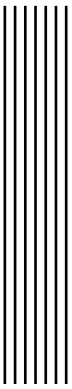


Simulink model of NARX network layer subsystem



Simulink model of NARX network output layer subsystem





Whole bibliography

[1] Nitin Shrivastava Aditya Narayan Bhatt. "Application of Artificial Neural Network for Internal Combustion Engines: A State of the Art Review". In: *Archives of Computational Methods in Engineering* (2021).

[2] Jinlong Liu & Qiao Huang & Christopher Ulishney & Cosmin E. Dumitrescu. "Machine learning assisted prediction of exhaust gas temperature of a heavy-duty natural gas spark ignition engine". In: *Sciencedirect* (2021).

[3] Qingyuan Tan & Xiaoye Han & Ming Zheng & Jimi Tjong. "Neural Network Soft Sensors for Gasoline Engine Exhaust Emission Estimation". In: *Journal of Energy Resources Technology* (2022).

[4] Sediako A.D & Andric J & Sjoblom J & Faghani E. "Heavy Duty Diesel Engine Modeling with Layered Artificial Neural Network Structures". In: *SAE Technical Paper* (2018).

[5] Lars Eriksson Lars Nielsen. *Modeling and Control of Engines and Drivelines*. Upplaga 1. Unknown, 2014. ISBN: 9781118479995.

[6] Robin Holmbom. "Modeling and Model-based Control of Automotive Air Paths". In: *Linköping University Electronic Press* (2022).

[7] Michael Nielsen. *Neural Networks and Deep Learning*. URL: `http://neuralnetworksanddeeplearning.com/index.html`. (accessed: 01.24.2022).

[8] Stephen A. Billings and Stephen Billings. *Nonlinear System Identification : NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. Upplaga 1. John Wiley & Sons, Incorporated, 2013. ISBN: 978-1-119-94359-4.

[9] MathWorks. *Layer-Recurrent Neural Networks*. URL: `https://se.mathworks.com/help/deeplearning/ug/design-layer-recurrent-neural-networks.html`. (accessed: 03.11.2022).

[10] MathWorks. *Long Short-Term Memory (LSTM)*. URL: `https://se.mathworks.com/discovery/lstm.html`. (accessed: 03.08.2022).

[11] Junyoung Chung & Caglar Gulcehre & KyungHyun Cho & Yoshua Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *arXiv:1412.3555* (2014).

[12] Ananth Ranganathan. "The Levenberg-Marquardt Algorithm". In: - (2004).

[13] Ziyu Shi. "Study on the influence of different number of neuron layers and activation functions on neural networks". In: *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)* (2021).

[14] Simone Sharma Siddharth Sharma and Anidhya Athaiya. "ACTIVATION FUNCTIONS IN NEURAL NETWORKS". In: *International Journal of Engineering Applied Sciences and Technology* (2020).

[15] Lennart Ljung Torkel Glad. *Modellbygge och simulering*. Upplaga 2. Studentlitteratur AB, 2003. ISBN: 9789144024431.

[16] Rajesh Kumar. *Fundamental of Artifical Neural Network and Fuzzy Logic*. Upplaga 1. Laxmi Publications, 2010. ISBN: 9788131807101.

[17] PennState Eberly college of Science. *Coefficient of Determination*. URL: https://online.stat.psu.edu/stat500/lesson/9/9.3. (accessed: 03.11.2022).

[18] PennState Eberly college of Science. *Coefficient of Determination and Correlation Examples*. URL: https://online.stat.psu.edu/stat462/node/97/. (accessed: 03.11.2022).

[19] Ali Kattan & Rosni Abdullah & Zong Woo Geem. *Artificial Neural Network Training and Software Implementation Techniques*. Upplaga 1. Nova Science Publishers, 2011. ISBN: 9781622571031.