



**Politecnico
di Torino**

POLITECNICO DI TORINO

College of Computer Engineering, Cinema and
Mechatronics

Master's Degree Thesis

Motor Temperature Virtual Sensing for a Small Scale Propulsion System

Supervisors

Prof. Bartolomeo MONTRUCCHIO
Prof. Matteo D.L. DALLA VEDOVA

Dr. Antonio Costantino MARCEDDU
Ing. Alessandro AIMASSO
Ing. Matteo BERTONE

Candidate

Brett SCHAUS

OCTOBER 2025

Abstract

Electric propulsion systems are increasingly adopted in aerospace applications such as drones, Unmanned Aerial Vehicles (UAVs), and emerging Urban Air Mobility (UAM). They offer advantages in efficiency, reduced emissions, and mechanical simplicity compared to combustion-based propulsion, but also introduce challenges. Among the most critical is thermal management: motors can overheat under high loads or prolonged operation, leading to reduced performance, reliability issues, and safety risks. Accurate monitoring and prediction of motor temperatures is therefore essential.

In testing, fiber optic temperature sensors are often used because they provide high accuracy and immunity to electromagnetic interference. However, they are costly and fragile: bending their cables can distort wavelengths, making them unsuitable for widespread or long-term use. If motor temperatures could instead be predicted from accessible electrical measurements such as currents and voltages, reliance on these delicate sensors could be reduced. This would enable safer and more economical testing while still detecting overheating risks.

This thesis addresses this challenge by developing a machine learning system for virtual engine temperature sensing via standard electrical signals, which could also be used for conditional monitoring. A dedicated test bench was designed to collect a dataset including motor currents, resistance, and speeds, alongside reference data from fiber optic sensors. Several Neural Network (NN) architectures were investigated, based on Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and more. The sequential nature of temperature evolution motivated RNNs, while CNNs were considered for feature extraction. Within the recurrent family, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers were compared with lightweight counterparts minimal LSTM (minLSTM) and minimal GRU (minGRU).

The results showed that the considered architectures are suitable for virtual measurement of engine-related temperatures. For predictive purposes, however, the amount and variability of the acquired data proved insufficient to describe the full thermal dynamics. Consequently, the networks often converged on predicting average values rather than tracking actual fluctuations, limiting their usefulness for motor temperature condition monitoring. This outcome emphasizes that richer datasets with greater operating diversity and longer recording times are essential for developing condition monitoring systems.

In summary, this thesis demonstrates that neural network architectures can support condition monitoring of electric propulsion systems using readily available electrical measurements. By reducing reliance on fiber optic sensors, the approach contributes to safer and more economical testing while laying the groundwork for predictive monitoring in UAVs, robotics, and electric vehicles.

Acknowledgements

I would first like to express my sincere gratitude to my supervisors, Alessandro Aimasso and Matteo Bertone, for their invaluable guidance, support, and encouragement throughout the course of this thesis. Their skill, constructive feedback, and patience were instrumental in helping me navigate challenges and complete this project.

I would also like to thank professors Antonio Marceddu, Matteo Dalla Vedova, and Bartolomeo Montrucchio, whose teaching and guidance have shaped the research involved and the tools needed to complete this thesis. Their dedication and expertise for the subject have been a precious source of motivation.

Finally, and most importantly, I wish to thank my wife, Monica Marchese, for her understanding, and encouragement. Her support, patience, and belief in me made this work possible, and I am deeply grateful for her presence throughout this journey.

Contents

List of Figures	VI
List of Tables	IX
Acronyms	XIII
1 Introduction	1
2 A Brief Background on Recurrent Neural Networks	3
2.1 Recurrent Neural Network Definition	3
2.2 Loss Functions	6
2.2.1 Mean Absolute Error	6
2.2.2 Mean Squared Error	7
2.2.3 Mean Bias Error	7
2.2.4 Mean Absolute Percentage Error	8
2.2.5 Root Mean Squared Error	9
2.3 Hyperparameter	9
2.4 Overfitting and Underfitting	11
2.5 Training Model	12
2.5.1 Backpropagation Through Time	12
2.6 Optimization Algorithms	12
2.6.1 Stochastic Gradient Descent	12
2.6.2 Momentum	13
2.6.3 AdaGrad	13
2.6.4 RMSProp	13
2.6.5 AdaDelta	13
2.6.6 Adam	14
2.7 Vanishing Gradient Problem	14
2.8 Neural Network Models	15
2.8.1 Long Short-Term Memory	15
2.8.2 Gated Recurrent Unit	17
2.8.3 minLSTM	17

2.8.4	minGRU	18
2.8.5	Extended Long Short-Term Memory	18
2.8.6	Bidirectional	19
2.8.7	Convolutional Neural Network	19
2.9	Inference time	20
2.10	Applications of Recurrent Neural Networks	21
3	Experiments	22
3.1	Equipment	22
3.1.1	Sensors	22
3.1.2	Test Bench with Arduino Due	24
3.1.3	Interrogator	26
3.2	Procedure	26
3.3	Condition Monitoring and Filtering	28
3.3.1	(Convolutional Layer + ReLU + MaxPool) x 1 + Recurrent Layers	28
3.3.2	(Convolutional Layer + ReLU + MaxPool) x 2 + Recurrent Layers	32
3.3.3	(Convolutional Layer + ReLU + MaxPool) x 3 + Recurrent Layers	36
3.3.4	Recurrent Layers	39
3.3.5	Recurrent Layers with Additional FC Layers (128, 64)	43
3.3.6	Recurrent Layers with Additional FC Layers (256, 128, 64)	47
3.4	Sequence Length of 500 and Output of 50	50
3.4.1	(Convolutional Layer + ReLU + MaxPool) x 3 + Recurrent Layers	51
3.4.2	(Convolutional Layer + ReLU + MaxPool) x 2 + Recurrent Layers	55
3.4.3	(Convolutional Layer + ReLU + MaxPool) x 1 + Recurrent Layers	58
3.5	Sequence Length of 250 and Output of 50	62
3.5.1	Recurrent Layers with Additional FC Layers of 256, 128, and 64 Neurons .	62
3.5.2	Recurrent Layers with Additional FC Layers of 128 and 64 Neurons	66
3.5.3	Recurrent Layers	70
3.6	Virtual Sensing	73
3.6.1	(Convolutional Layer + ReLU + MaxPool) x 1 + Recurrent Layers	73
3.6.2	Recurrent Layers with Additional FC Layers of 256, 128, and 64 Neurons .	74
3.6.3	Recurrent Layers with Additional FC Layers of 128 and 64 Neurons	75
3.6.4	Recurrent Layer with Additional FC Layer of 64 Neurons	76
4	Results	77
4.1	Condition Monitoring and Filtering Results	77
4.1.1	Sequence Length of 50 and Output of 1	77
4.1.2	Sequence Length of 500 and Output of 50	81
4.1.3	Sequence Length of 250 and Output of 50	83
4.2	Consideration of the Results Obtained	85
4.3	Virtual Sensing Results	86
5	Conclusions	91
	Bibliography	93

List of Figures

2.1	RNNs shown via cycle edges. Each layer is unfolded as shown[1].	3
2.2	An RNN with a hidden state [1].	4
2.3	Sigmoid or Logistic activation function graph [5].	4
2.4	Tanh activation function graph [5].	5
2.5	ReLU activation function graph [5].	5
2.6	Softmax activation function graph [5].	5
2.7	SoftPlus activation function graph [5].	6
2.8	Mean Absolute Error [6].	6
2.9	Mean Squared Error [6].	7
2.10	Mean Bias Error [6].	8
2.11	Mean Absolute Percentage Error [6].	8
2.12	Root Mean Squared Error [6].	9
2.13	Hyperparameter Optimization [1].	9
2.14	Example of Layer stacking [7].	10
2.15	Example of hidden layers [8].	10
2.16	Learning Rate Visualization [9].	11
2.17	Underfitting and Overfitting [10].	11
2.18	Backpropagation in RNN model [1].	12
2.19	Architecture of the memory cell [11].	15
2.20	Forget Gate Layer [12].	15
2.21	Input Gate Layer [12].	16
2.22	Updating the cell state [12].	16
2.23	LSTM Output [12].	16
2.24	Gated Recurrent Unit [13].	17
2.25	sLSTM forward pass [15].	18
2.26	Stabilizing state m_t [15].	18
2.27	mLSTM Forward Pass [15].	19
2.28	xLSTM Architecture [15].	19
2.29	Max Pooling Operation [16].	20
2.30	Convolutional Network Stack.	20

3.1	Fiber Bragg Grating Schematic.	22
3.2	Structure of an optical fiber cable [17].	23
3.3	INA219 Digital Current Sensor [18].	23
3.4	ACS712 Analog Current Sensor [19].	24
3.5	Test Bench used to take measurements with cover.	25
3.6	Test Bench used to take measurements without cover	25
3.7	Interrogator [20].	26
3.8	Recurrent and convolutional flowchart.	28
3.9	Recurrent and convolutional flowchart.	32
3.10	Recurrent and convolutional flowchart.	36
3.11	Recurrent flowchart.	39
3.12	Recurrent network with additional fully connected layers flowchart.	43
3.13	Recurrent Layer with additional fully connected layers	47
3.14	Recurrent and convolutional flowchart.	51
3.15	Recurrent and convolutional flowchart.	55
3.16	Recurrent and convolutional flowchart.	58
3.17	Recurrent layer with additional fully connected layers flowchart.	62
3.18	Recurrent layer with additional fully connected layers flowchart.	66
3.19	Recurrent layer flowchart.	70
3.20	Flowchart of CNN x 1.	73
3.21	Flowchart of FC layers of 256, 128 and 64.	74
3.22	Flowchart of FC layers of 128 and 64.	75
3.23	Flowchart of FC layer of 64.	76
4.1	Complexity of minGRUseq with one convolutional layer.	77
4.2	Complexity of minLSTMseq with one convolutional layer.	78
4.3	Complexity of GRU with two convolutional layers.	78
4.4	Complexity of LSTM with two convolutional layers.	79
4.5	Complexity of GRU with two convolutional layers.	79
4.6	Complexity of xLSTM.	80
4.7	Complexity of xLSTM with only recurrent layer.	80
4.8	Complexity of RNN with additional fully connected layers.	81
4.9	Complexity of minLSTMseq with three convolutional layers.	81
4.10	Predicted motor temperatures with three convolutional layers.	82
4.11	Complexity of minLSTMseq with two convolutional layers.	82
4.12	Complexity of minLSTMseq with one convolutional layer.	83
4.13	Predicted motor temperatures with sequence length of 250 and output of 50	83
4.14	Complexity of minGRUseq with additional fully connected layers.	84
4.15	Predicted motor temperatures with sequence length of 250 and output of 50	84

4.16	Combined temperatures of all datasets. The red line indicates the rise in temperatures as the speed increases. Zone 1 indicates the dataset in which ventilation effects are not negligible. Zone 2 indicates the datasets where the dissipative effects of Joule loss begin to increase.	85
4.17	Flowchart of CNN x 1.	86
4.18	Virtual sensing using one convolutional layer.	86
4.19	Virtual sensing using an FC layer with 64 neurons.	87
4.20	Virtual sensing using an FC layer with 64 neurons.	88
4.21	Virtual sensing using an FC layer with 64 neurons.	88
4.22	Pearson's Correlation Matrix	89
4.23	Spearman's Correlation Matrix	89

List of Tables

3.1	RMSE values and training times for different LR.	28
3.2	MAPE values and training times for different LR.	29
3.3	RMSE values and training times for different batch sizes.	29
3.4	MAPE values and training times for different batch sizes.	29
3.5	RMSE values and training times for different hidden sizes.	30
3.6	MAPE values and training times for different hidden sizes.	30
3.7	RMSE values and training times for different numbers of layers.	30
3.8	MAPE values and training times for different numbers of layers.	31
3.9	Comparison of models in terms of test RMSE, test MAPE, and runtime.	31
3.10	RMSE values and training times for different LR.	32
3.11	MAPE values and training times for different LR.	33
3.12	RMSE values and training times for different batch sizes.	33
3.13	MAPE values and training times for different batch sizes.	33
3.14	RMSE values and training times for different hidden sizes.	34
3.15	MAPE values and training times for different hidden sizes.	34
3.16	RMSE values and training times for different numbers of layers.	34
3.17	MAPE values and training times for different numbers of layers.	35
3.18	Comparison of models in terms of test RMSE, test MAPE, and runtime.	35
3.19	RMSE values and training times for different LR.	36
3.20	MAPE values and training times for different LR.	37
3.21	RMSE values and training times for different batch sizes.	37
3.22	MAPE values and training times for different batch sizes.	37
3.23	RMSE values and training times for different hidden sizes.	38
3.24	[MAPE values and training times for different hidden sizes.	38
3.25	RMSE values and training times for different numbers of layers.	38
3.26	MAPE values and training times for different numbers of layers.	39
3.27	Comparison of models in terms of test RMSE, test MAPE, and runtime.	39
3.28	RMSE values and training times for different LR.	40
3.29	MAPE values and training times for different LR.	40
3.30	RMSE values and training times for different batch sizes.	40

3.31	MAPE values and training times for different batch sizes.	41
3.32	RMSE values and training times for different hidden sizes.	41
3.33	MAPE values and training times for different hidden sizes.	41
3.34	RMSE values and training times for different numbers of layers.	42
3.35	MAPE values and training times for different numbers of layers.	42
3.36	Comparison of models in terms of test RMSE, test MAPE, and runtime.	42
3.37	RMSE values and training times for different LR's.	43
3.38	MAPE values and training times for different LR's.	44
3.39	RMSE values and training times for different batch sizes.	44
3.40	MAPE values and training times for different batch sizes.	44
3.41	RMSE values and training times for different hidden sizes.	45
3.42	MAPE values and training times for different hidden sizes.	45
3.43	RMSE values and training times for different numbers of layers.	45
3.44	MAPE values and training times for different numbers of layers.	46
3.45	Comparison of models in terms of test RMSE, test MAPE, and runtime.	46
3.46	RMSE values and training times for different LR's.	47
3.47	MAPE values and training times for different LR's.	48
3.48	RMSE values and training times for different batch sizes.	48
3.49	MAPE values and training times for different batch sizes.	48
3.50	RMSE values and training times for different hidden sizes.	49
3.51	MAPE values and training times for different hidden sizes.	49
3.52	RMSE values and training times for different numbers of layers.	49
3.53	MAPE values and training times for different numbers of layers.	50
3.54	Comparison of models in terms of test RMSE, test MAPE, and runtime.	50
3.55	LR results at different RMSE values.	51
3.56	LR results at different RMSE values.	52
3.57	Batch size results at different RMSE values.	52
3.58	Batch size results at different RMSE values.	52
3.59	Hidden size results at different RMSE values.	53
3.60	Hidden size results at different RMSE values.	53
3.61	RMSE values obtained in relation to the number of layers.	53
3.62	MAPE values obtained in relation to the number of layers.	54
3.63	Comparison of models based on RMSE.	54
3.64	Comparison of models based on MAPE.	54
3.65	LR results at different RMSE values.	55
3.66	LR results at different RMSE values.	56
3.67	Batch size results at different RMSE values.	56
3.68	Batch size results at different RMSE values.	56
3.69	Hidden size results at different RMSE values.	57

3.70	Hidden size results at different RMSE values.	57
3.71	RMSE values obtained in relation to the number of layers.	57
3.72	MAPE values obtained in relation to the number of layers.	57
3.73	Comparison of models based on RMSE.	58
3.74	Comparison of models based on MAPE.	58
3.75	LR results at different RMSE values.	59
3.76	LR results at different RMSE values.	59
3.77	Batch size results at different RMSE values.	59
3.78	Batch size results at different RMSE values.	60
3.79	Hidden size results at different RMSE values.	60
3.80	Hidden size results at different RMSE values.	60
3.81	RMSE values obtained in relation to the number of layers.	61
3.82	MAPE values obtained in relation to the number of layers.	61
3.83	Comparison of models based on RMSE.	61
3.84	Comparison of models based on MAPE.	62
3.85	LR results at different RMSE values.	63
3.86	LR results at different RMSE values.	63
3.87	Batch size results at different RMSE values.	63
3.88	Batch size results at different RMSE values.	64
3.89	Hidden size results at different RMSE values.	64
3.90	Hidden size results at different RMSE values.	64
3.91	RMSE values obtained in relation to the number of layers.	65
3.92	MAPE values obtained in relation to the number of layers.	65
3.93	Comparison of models based on RMSE.	65
3.94	Comparison of models based on MAPE.	66
3.95	LR results at different RMSE values.	66
3.96	LR results at different RMSE values.	67
3.97	Batch size results at different RMSE values.	67
3.98	Batch size results at different RMSE values.	67
3.99	Hidden size results at different RMSE values.	68
3.100	Hidden size results at different RMSE values.	68
3.101	RMSE values obtained in relation to the number of layers.	68
3.102	MAPE values obtained in relation to the number of layers.	69
3.103	Comparison of models based on RMSE.	69
3.104	Comparison of models based on MAPE.	69
3.105	LR results at different RMSE values.	70
3.106	LR results at different RMSE values.	70
3.107	Batch size results at different RMSE values.	71
3.108	Batch size results at different RMSE values.	71

3.109Hidden size results at different RMSE values.	71
3.110Hidden size results at different RMSE values.	72
3.111RMSE values obtained in relation to the number of layers.	72
3.112MAPE values obtained in relation to the number of layers.	72
3.113Comparison of models based on RMSE.	73
3.114Comparison of models based on MAPE.	73
3.115Learning Rate Testing	74
3.116Batch Size Testing	74
3.117Learning Rate Testing	74
3.118Batch Size Testing	74
3.119Learning Rate Testing	75
3.120Batch Size Testing	75
3.121Learning Rate Testing	76
3.122Batch Size Testing	76
3.123Comparison of models based on RMSE.	76

Acronyms

B

BPTT Backpropagation Through Time 12

C

CNN Convolutional Neural Network 3, 19, 73, 86, 90, 92

F

FBG Fiber Bragg Grating 22

FC Fully Connected 19, 43, 47, 62, 66, 73–76, 80, 81, 83, 84, 87, 88, 90–92

G

GPU Graphics Processing Unit 27

GRU Gated Recurrent Unit 2, 14, 17–19, 26, 27, 31, 77–79, 84, 91

I

IDE Integrated Development Environment 26

L

LR Learning Rate 11, 12, 26–29, 32, 33, 36, 37, 40, 41, 43, 44, 47, 48, 51, 52, 55–57, 59, 60, 63, 66, 67, 70, 71, 74–76

LSTM Long Short-Term Memory 2, 14–19, 26–28, 31, 77–79, 81–83, 91

M

MAE Mean Absolute Error 6, 7

MAPD Mean Absolute Percent Deviation 8

MAPE Mean Absolute Percentage Error 8, 29–31, 33–35, 37–42, 44–46, 48–50, 52–54, 56–73

MBE Mean Bias Error 7

MSE Mean Squared Error 7, 9

N

NLP Natural Language Processing 21

NN Neural Network 1–4, 10–12, 14, 19, 26, 27, 85, 91, 92

R

ReLU Rectified Linear Unit 5, 6, 14, 19, 20, 28, 32, 36, 51, 55, 58, 73

RMSE Root Mean Squared Error 9, 27–61, 63–73, 76, 77, 80–84, 87, 88, 91

RNN Recurrent Neural Network 3, 4, 12, 15, 17, 19, 21, 22, 26, 81, 92

RPM Revolutions Per Minute 26

S

SGD Stochastic Gradient Descent 12, 13

U

UAM Urban Air Mobility 92

UAV Unmanned Aerial Vehicle 1, 91, 92

X

xLSTM Extended Long Short-Term Memory 2, 18, 19, 26, 80

Chapter 1

Introduction

The use of electric propulsion in Unmanned Aerial Vehicles (UAVs), drones, and other small-scale aircraft has grown rapidly in recent years due to its advantages in efficiency, mechanical simplicity, and environmental impact compared to traditional combustion engines. However, with these benefits come new engineering challenges, particularly in the area of thermal management. Motors in electric propulsion systems are subject to high stresses during demanding flight conditions. Excessive heating can lead to efficiency losses, premature component wear, and in the worst case, motor failure during flight. For this reason, understanding and predicting motor temperatures is a critical aspect of ensuring both safety and reliability.

In current laboratory and prototype environments, motor temperatures are commonly measured with fiber optic sensors. These sensors offer high accuracy and immunity to electromagnetic interference, making them suitable for capturing detailed thermal behaviour under dynamic operating conditions. Yet, despite these advantages, fiber optic sensors are expensive and extremely delicate. Their fragility makes them unsuitable for long-term use in repetitive experiments, while their cost limits their application in larger testing campaigns. As a result, there is a clear gap between the precision required for safe system evaluation and the practicality of existing sensing solutions.

This challenge opens the door for alternative approaches to thermal monitoring, particularly those that leverage the large amount of data already available from electrical measurements. Current and speed signals can be measured cheaply and robustly, and they inherently reflect the operating state of the propulsion system. The central problem addressed in this thesis is therefore whether Neural Networks (NNs) can be used to predict motor temperatures from such accessible signals, thereby reducing or even eliminating the need for costly fiber optic sensors.

The scope of this thesis is to extend and elaborate on a previously developed test bench model to provide detailed analysis before implementing small-scale electric propulsion systems in flight. Such an approach ensures that these systems can be thoroughly and safely tested prior to deployment, thereby reducing technical risk and improving confidence in their operational performance.

To address this problem, additional current sensors were integrated into the test bench, allowing for the acquisition of a comprehensive dataset suitable for machine learning applications. By pairing electrical measurements with reference temperature values, it became possible to train models capable of learning the hidden relationships between operating conditions and thermal behaviour.

However, during the course of this work it became clear that while temperature prediction through neural networks holds promise, the results obtained from condition monitoring alone were not always sufficiently meaningful for reliable deployment because of the provided dataset. This observation motivated an expansion of the thesis scope toward virtual sensing more generally. Virtual sensing refers to the use of data-driven methods to estimate quantities that are difficult, expensive, or impractical to measure directly. In the context of electric propulsion systems, such techniques can enable the estimation of a variety of internal states—such as torque, efficiency, or losses—based only on readily available measurements via standard electrical signals.

By incorporating virtual sensing alongside motor temperature prediction, this thesis aims to provide a broader assessment of how neural networks can support the monitoring and evaluation of electric propulsion systems. This dual focus ensures that even if direct thermal estimation proves challenging, the methodology developed can still yield practical insights into system behaviour and performance, thereby retaining its value for test bench development and eventual in-flight applications.

The primary research questions guiding this work can be formulated as follows:

- Can NNs predict motor temperatures with sufficient accuracy to serve as a reliable alternative to direct fiber optic measurements?
- Which NN architectures—recurrent and/or convolutional—offer the best balance between predictive performance and training efficiency?
- How do standard Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers compare to their minimal versions, minGRU and minLSTM, in terms of accuracy, complexity, and suitability for potential real-time deployment?
- How do new models such as Extended Long Short-Term Memory (xLSTM) compare to older models?
- Beyond temperature estimation, can virtual sensing provide practical alternatives for monitoring other relevant systems?

To answer these questions, the thesis explores several architectures, focusing on recurrent and convolutional NNs and evaluating different recurrent layer types. Their performance is compared using prediction accuracy, training time, and convergence behaviour as evaluation criteria. The goal is not only to identify the best-performing model, but also to understand the trade-offs between accuracy and efficiency, thereby offering practical guidance for future applications.

The remainder of this thesis is structured as follows. Chapter 2 introduces the theoretical background of recurrent networks, activation functions, loss functions, hyperparameters, optimization algorithms, and the various recurrent layers considered in this work. Chapter 3 presents the experimental setup, including the training configurations and the network architectures that were evaluated. Chapter 4 discusses the results obtained from these experiments, and Chapter 5 concludes the thesis with a summary of findings and suggestions for future research.

Chapter 2

A Brief Background on Recurrent Neural Network

Synopsis - This chapter explains the basic concepts of Neural Networks (NNs), focusing on Recurrent Neural Networks (RNNs) with a brief explanation of Convolutional Neural Networks (CNNs). RNNs are a class of deep learning models that obtain the dynamics of sequences with *recurrent* connections. These connections are dynamic as they pass information across adjacent time steps. The following figure represents a feedforward NN where each layer's parameters are shared across time steps [1].

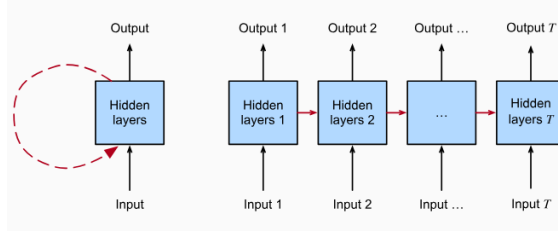


Figure 2.1. RNNs shown via cycle edges. Each layer is unfolded as shown [1].

This type of NN works particularly well at processing sequential data, in which the order of the elements matters. Unlike other networks, RNNs have a retaining ability that allows them to make calculations using previous inputs while processing the current input. The RNN receives an input at each time step. This input is processed by a hidden layer, and the output is fed back into the network for the next time step. This output can be used for predictions or future processing. The retaining ability of RNNs is essentially a "memory" that influences future inputs. These models originated as models of the brain for cognitive scientists and later became practical modeling tools for machine learning. RNNs rose in popularity due to breakthroughs in handwriting recognition [2], machine translation [3], and making diagnoses [4].

2.1 Recurrent Neural Network Definition

To define NNs, batch processing must first be discussed. It is the process of taking the training data and dividing it into smaller "batches". Doing this allows the entire dataset to be fed to the NN in smaller and more manageable portions, allowing more efficiency thus reducing training time.

Assuming we have a mini-batch of inputs $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ at time step t , where n corresponds to the number of sequence examples for each time step t and d corresponds to the number of inputs. The hidden layer's activation function is ϕ . Next, $\mathbf{H}_t \in \mathbb{R}^{n \times h}$ is the hidden layer output

of time step t , and $\mathbf{H}_{t-1}$ is the previously saved time step. Then a weight parameter is introduced $\mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$ to describe the use of the hidden layer output of the previous time step in the current time step. $\mathbf{b}_h \in \mathbb{R}^{1 \times h}$ represents the bias parameter and h the number of hidden units. The entire calculation is shown below:

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h). \quad [1]$$

The layers in RNNs that perform this computation are called *recurrent layers*.

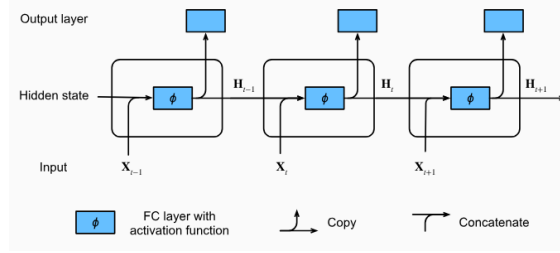


Figure 2.2. An RNN with a hidden state [1].

When designing a NN, a deciding factor is the selection of an activation function for both the hidden layer 2.2 and the output layer. This allows non-linearity in a model, allowing a more complex representation of patterns in data. Without this feature, the NN would always behave in a linear regressive way regardless of the number of layers.

Possible non-linear function choices can be the following:

1. $\phi(x) = \frac{1}{1+e^{-x}}$, the sigmoid function, shown in Figure 2.3, which binds the values of x between 0 and 1.

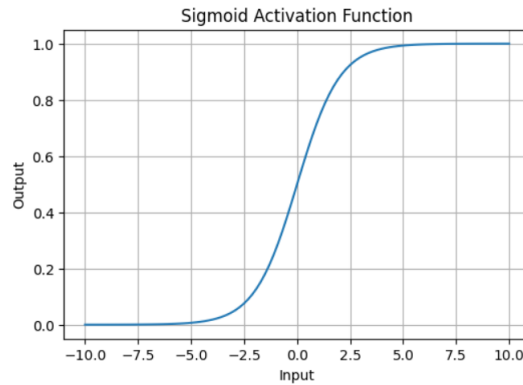


Figure 2.3. Sigmoid or Logistic activation function graph [5].

2. $\phi(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$, the hyperbolic tangent in Figure 2.4, which shrinks the values of x between -1 and 1.

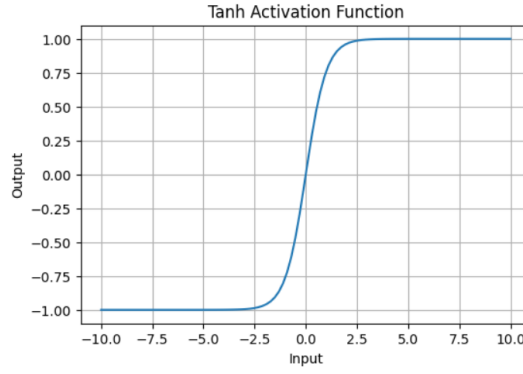


Figure 2.4. Tanh activation function graph [5].

3. $\phi(x) = \max(0, x)$, the Rectified Linear Unit (ReLU) activation function in Figure 2.5, which binds the values of x between 0 and ∞ .

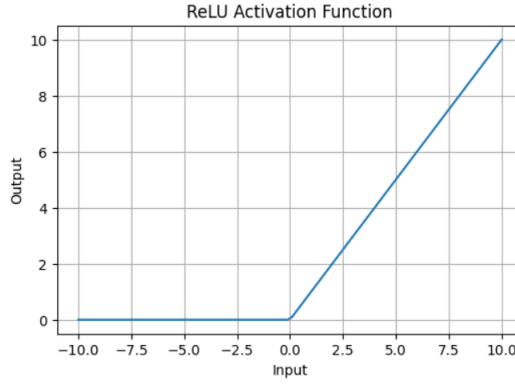


Figure 2.5. ReLU activation function graph [5].

4. $\phi(x) = \text{SoftMax}(x)$, the softmax function in Figure 2.6, which computes as output a probability vector (components in (0,1) summing to 1).

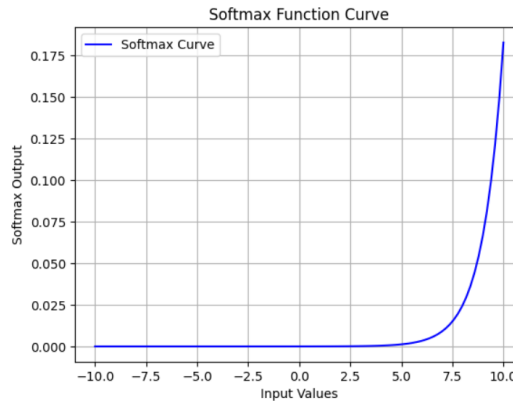


Figure 2.6. Softmax activation function graph [5].

5. $\phi(x) = \log(1 + e^x)$, the SoftPlus function in Figure 2.7, which ranges from 0 to ∞ without the hard zero of ReLU and is smooth and continuous.

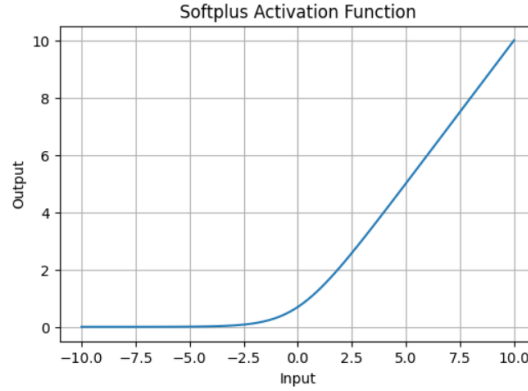


Figure 2.7. SoftPlus activation function graph [5].

2.2 Loss Functions

With time series predictions, choosing a loss function is very important as it drives the learning process of the algorithm. There are many different types of loss functions available. Proposed below are the most commonly used ones for similar problems.

2.2.1 Mean Absolute Error

Mean Absolute Error (MAE), also known as L1 loss in Figure 2.8, is the absolute error between predicted and actual values:

$$L1 = |y_{\text{actual}} - y_{\text{predicted}}| \quad [6]$$

The mean of the absolute errors of all sample values is called MAE:

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad [6]$$

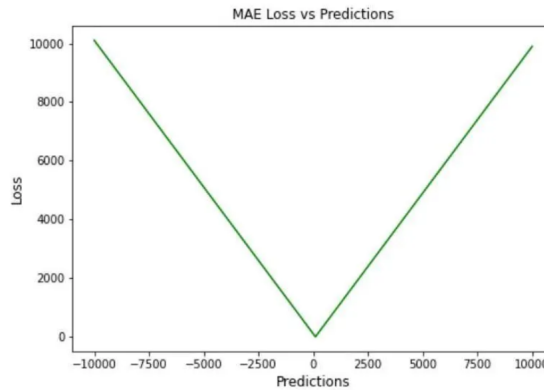


Figure 2.8. Mean Absolute Error [6].

Advantage: MAE is simple to compute and displays uniform model performance. It is also less sensitive to outliers.

Disadvantage: Since the scoring method is linear, all errors are weighted equally which affects the mean. Minima may be lost during backpropagation due to the steepness, and since the function is not continuous, computing gradients is difficult at zero.

2.2.2 Mean Squared Error

Mean Squared Error (MSE), also known as L2 loss in Figure 2.9, is the squared error between predicted and actual values:

$$L_2 = (y_{\text{actual}} - y_{\text{predicted}})^2 \quad [6]$$

The mean of the squared errors of all sample values is called MSE, also known as the mean squared error:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad [6]$$

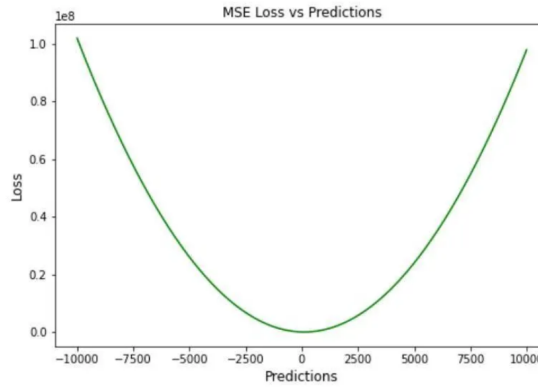


Figure 2.9. Mean Squared Error [6].

Advantage: MSE is effective when gradients are reduced gradually and errors are small, which limits the convergence to a minimum.

Disadvantage: Since the values are squared, the training is faster but with higher losses, and backpropagation can cause large jumps. Unlike MAE, this loss function is sensitive to outliers.

2.2.3 Mean Bias Error

Bias in Figure 2.10 is known as the tendency to overestimate or underestimate values. Deviation can be either positive or negative. Positive indicates overestimation and negative indicates underestimation. It is very similar to MAE except it does not take into account absolute values. Since positive and negative values can cancel each other out, this loss function should be treated cautiously.

$$\text{MBE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i) \quad [6]$$

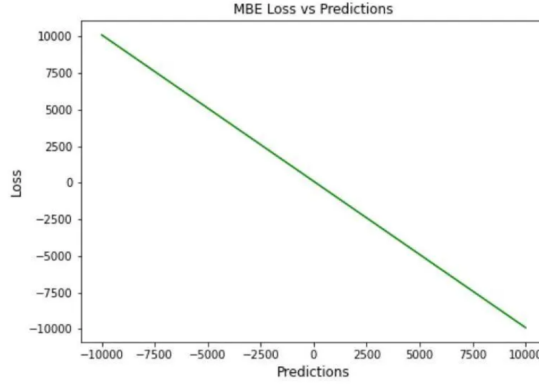


Figure 2.10. Mean Bias Error [6].

Advantage: Used to identify whether the model is biased in either the positive or negative direction.

Disadvantage: Since errors can cancel each other out, it is not appropriate for models that range from $(-\infty, \infty)$.

2.2.4 Mean Absolute Percentage Error

Mean Absolute Percentage Error (MAPE) in Figure 2.11, also known as Mean Absolute Percent Deviation (MAPD), is a loss function that evaluates the accuracy of a system. Often used with regression analysis and model evaluation when outliers are not present.

$$\text{MAPE} = \frac{1}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \quad [6]$$

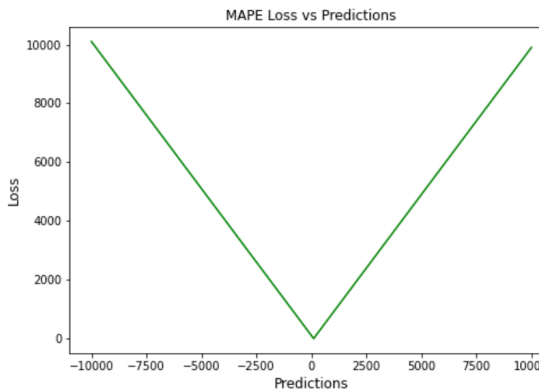


Figure 2.11. Mean Absolute Percentage Error [6].

Advantage: Since MAPE is expressed in percentages, the size of the variable does not matter. This also avoids the problem of positive number offsetting due to negative numbers.

Disadvantage: In the MAPE equation, the predicted output is the denominator and can be zero, which results in an undefined value. Positive errors are penalized more than negative errors.

2.2.5 Root Mean Squared Error

This loss function in Figure 2.12 uses MSE and takes the square, which accounts for the variation in actual values with measured ones by taking the average magnitude of the error. This function is most useful when huge errors are highly undesirable.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad [6]$$

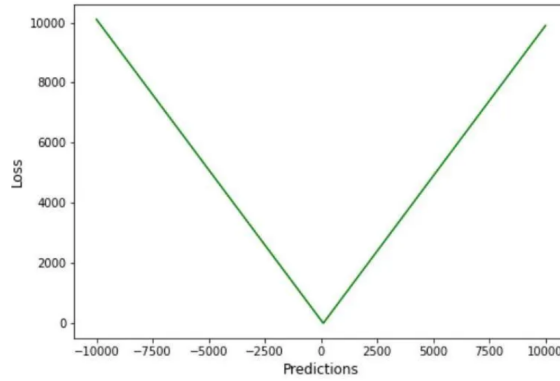


Figure 2.12. Root Mean Squared Error [6].

Advantage: This loss function is used often because of its ease to differentiate and compute. Since the square root penalizes larger values less, extreme losses are far fewer.

Disadvantage: Because of the linearity of Root Mean Squared Error (RMSE), gradients cannot be computed easily near the minimum. Since RMSE is sensitive to outliers, additional pre-processing must be done beforehand, resulting in more overhead.

2.3 Hyperparameter

Hyperparameters Figure 2.13 in machine learning consist of variables that can be tuned in order to help the model learn more effectively and produce better results.

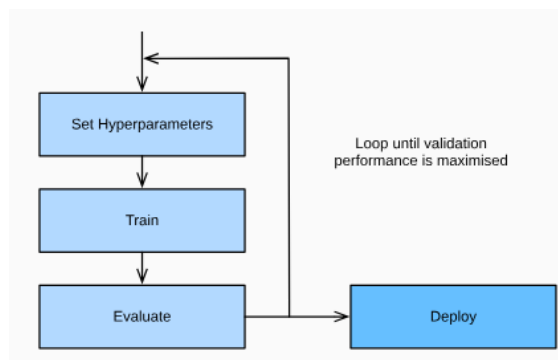


Figure 2.13. Hyperparameter Optimization [1].

Layers

One of the many hyperparameters available with NNs is layers represented in Figure 2.14. The ability to stack layers upon each other in order to have the data processed more times.

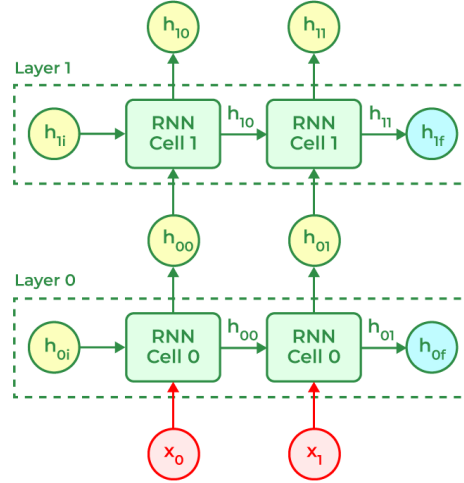


Figure 2.14. Example of Layer stacking [7].

Batch Size

The batch size is a hyperparameter that specifies how many samples are used before updating the weights.

Hidden Size

The hidden size hyperparameter in Figure 2.15 dictates the depth of the NN which is directly related to its complexity and learning ability. With less layers, the model is faster but learns less. With a deeper network, data can be classified more thoroughly.

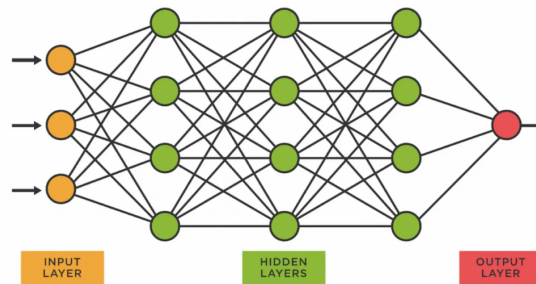


Figure 2.15. Example of hidden layers [8].

Learning Rate

The Learning Rate (LR) is a hyperparameter that dictates how much a NN adjusts its parameters at each step of the optimizing algorithm.

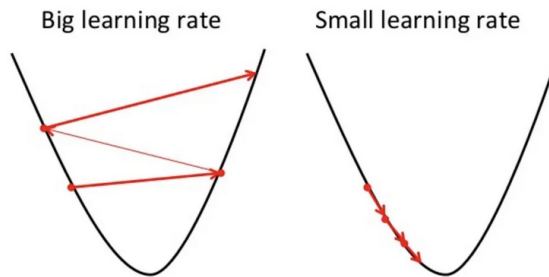


Figure 2.16. Learning Rate Visualization [9].

As can be seen from Figure 2.16, having too large of a learning rate can completely miss minima and having too small of a learning can make training far too slow.

Epoch

An epoch is a hyperparameter that specifies the number of times a NN will use the entire dataset.

2.4 Overfitting and Underfitting

Models can be either overfit or underfit, as can be seen in Figure 2.17. When a model is overfitting data it memorized the training data too much so when new data is introduced, the accuracy is very poor because it is not generalizing.

On the other hand, a model can also be underfit, where it does not produce good results while training or testing because it does not properly learn the patterns in the provided data.

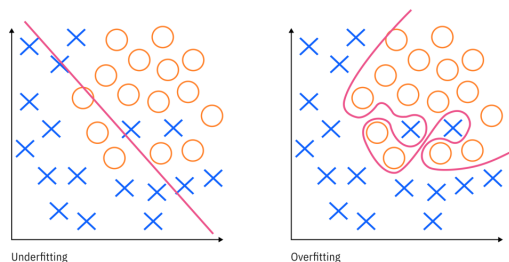


Figure 2.17. Underfitting and Overfitting [10].

2.5 Training Model

2.5.1 Backpropagation Through Time

Backpropagation Through Time (BPTT) is the extension of backpropagation by calculating derivatives of a target with respect to a large set of parameters, applying it to dynamic systems such as NNs.

Figure 2.18 shows an unfolded RNN which illustrates the different layers which for RNNs share the weights. Each of these are considered time steps t . Gradients are computed for each time step and finally summed up:

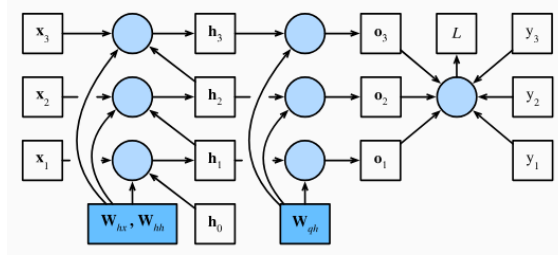


Figure 2.18. Backpropagation in RNN model [1].

$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^T \text{prod} \left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}} \right) = \sum_{t=1}^T \frac{\partial L}{\partial \mathbf{o}_t} \mathbf{h}_t^\top \quad [1]$$

BPTT computes the error at the last time step t and then, as the name suggests, back propagates through the network k times.

2.6 Optimization Algorithms

After computing the gradient with BPTT, the network parameters can be updated using an optimization algorithm to minimize the cost function.

2.6.1 Stochastic Gradient Descent

A solid first option for an optimizer is the Stochastic Gradient Descent (SGD) which updates the parameters going in the opposite direction of the gradient $\eta \nabla f_i$ by a step-size (the learning rate), for each input $x(t)$:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f_i(\mathbf{x}) \quad [1]$$

A problem with SGD is that the objective function fluctuates heavily because of the frequent updates for every time step t . This fluctuation can be used to move from one local minimum to a potentially better one. NNs, generally, perform better when updating mini-batches of n training samples reaching a more stable convergence. This avoids drastic changes by performing optimized matrix operations more efficiently for each mini-batch. Unfortunately, SGD cannot guarantee convergence to a global minima. These challenges arise from the difficulty in choosing a sufficient learning rate. Smaller LRs can converge too slowly and larger LRs can cause disturbances or possibly divergence. One possible solution is to schedule the learning rate during training, allowing it to change if the model sees fit.

Lastly, saddle points and local minima push the gradient too close to zero which stop it from finding optimal minima.

The following optimization algorithms try to address these challenges.

2.6.2 Momentum

Momentum is an algorithm which accelerates SGD by replacing gradients with a leaky average over past gradients which improves convergence. The formula for leaky average is below:

$$\mathbf{v}_t = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau, t-\tau-1} \quad [1]$$

where $\beta \in (0, 1)$ and $\mathbf{v}$ is velocity.

It works well for both noise-free gradient descent and noisy stochastic gradient descent. Momentum prevents the stalling that was discussed with SGD during the optimization process. Since an additional state vector (velocity) is involved, implementation is much more straightforward.

2.6.3 AdaGrad

Instead of manually adjusting the learning rate, Adagrad can be used to dynamically decrease it on a per-coordinate basis. It achieves this by judging the progress $\rightarrow$ large gradients are compensated with smaller learning rates. AdaGrad has the ability to help mitigate the distortion of uneven structures. When there are not many features, AdaGrad can be particularly useful for adjusting the learning rates more slowly for the more infrequent terms.

$$\begin{aligned} \mathbf{g}_t &= \partial_{\mathbf{w}} \ell(y_t, f(\mathbf{x}_t, \mathbf{w})), \\ \mathbf{s}_t &= \mathbf{s}_{t-1} + \mathbf{g}_t^2, \\ \mathbf{w}_t &= \mathbf{w}_{t-1} - \frac{\eta}{\sqrt{\mathbf{s}_t} + \epsilon} \cdot \mathbf{g}_t. \end{aligned} \quad [1]$$

where s_t is used to accumulate past gradient variance.

2.6.4 RMSProp

RMSProp was invented to alleviate the problem of pre-defined learning rate scheduling. It uses a simple fix by decoupling the coordinate-adaptive learning rates of AdaGrad. Since the gradient $\mathbf{g}_t$ is squared in AdaGrad, $\mathbf{s}_t$ continues to grow without bound. One way to fix this problem is to use leaky average in the same way as the Momentum optimization:

$$\begin{aligned} \mathbf{s}_t &= \gamma \mathbf{s}_{t-1} + (1 - \gamma) \mathbf{g}_t^2, \\ \mathbf{w}_t &= \mathbf{w}_{t-1} - \frac{\eta}{\sqrt{\mathbf{s}_t} + \epsilon} \cdot \mathbf{g}_t, \end{aligned} \quad [1]$$

RMSProp is similar to AdaGrad in that they both use the square of the gradient to scale coefficients. However, RMSProp also shares the leaky average with Momentum but in a slightly different way by adjusting the coefficient-wise pre-conditioner. The coefficient γ determines the history of the per-coordinate scale.

2.6.5 AdaDelta

AdaDelta is an extension of AdaGrad except that it does not have a learning rate parameter. Instead, the learning rate is adapted from the rate of change of the parameters. AdaDelta also uses leaky averages to estimate the appropriate statistics.

$$\begin{aligned} \mathbf{s}_t &= \rho \mathbf{s}_{t-1} + (1 - \rho) \mathbf{g}_t^2, \\ \mathbf{x}_t &= \mathbf{x}_{t-1} - \mathbf{g}_t', \\ \mathbf{g}_t' &= \frac{\sqrt{\Delta \mathbf{x}_{t-1} + \epsilon}}{\sqrt{\mathbf{s}_t} + \epsilon} \cdot \mathbf{g}_t, \\ \Delta \mathbf{x}_t &= \rho \Delta \mathbf{x}_{t-1} + (1 - \rho) (\mathbf{g}_t')^2 \end{aligned} \quad [1]$$

where $\Delta \mathbf{x}_{t-1}$ is the leaky average of the squared rescaled gradients $\mathbf{g}'_t$ and $\Delta \mathbf{x}_0$ is initialized to 0. Lastly, ϵ is added to maintain numerical stability.

2.6.6 Adam

Adam combines all of the previous techniques in an efficient learning algorithm, even if there are situations where it can diverge from poor variance control.

A key component of the Adam algorithm is how it uses exponential weighted moving averages for estimating the momentum and the second moment of the gradient. Here are the following state variables:

$$\begin{aligned}\mathbf{v}_t &= \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1) \mathbf{g}_t, \\ \mathbf{s}_t &= \beta_2 \mathbf{s}_{t-1} + (1 - \beta_2) \mathbf{g}_t,\end{aligned}\tag{1}$$

With β_1 and β_2 being non-negative weighting parameters. In this way, the variance estimate moves more slowly than the momentum term. The normalized state variables are as follows:

$$\begin{aligned}\hat{\mathbf{v}}_t &= \frac{\mathbf{v}_t}{1 - \beta_1^t}, \\ \hat{\mathbf{s}}_t &= \frac{\mathbf{s}_t}{1 - \beta_2^t},\end{aligned}\tag{1}$$

With the above formulas, it is now possible to create the update equations. By rescaling the gradient in a manner very similar to RMSProp, the following formula is obtained:

$$\mathbf{g}'_t = \frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon},\tag{1}$$

And lastly, the update formula:

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \mathbf{g}'_t\tag{1}$$

With the various steps, it is clear to see how Adam was derived. The Momentum and scale are shown in the state variables. This definition forces the terms to de-bias but this could be fixed by initializing it differently and updating the condition. RMSProp allows an easy combination of both terms. Lastly, the learning rate η controls the step length which helps address convergence issues.

2.7 Vanishing Gradient Problem

In the optimization process for deep learning, using gradients is a fundamental part of training NNs. But sometimes gradients can encounter two problems, either they vanish or they explode. Vanishing gradients occur when the network is backpropagating and the derivatives or slopes of the activation function become increasingly smaller to the point of "vanishing". This causes the updating weights to change so slightly that it essentially doesn't change so the training can stall. This problem can be solved using better activation functions such as ReLU or by using LSTMs or GRUs, which will be discussed in the next section.

Exploding gradients work in a very similar fashion, but instead of vanishing, they become extremely large and "explode", causing weights to increase too much and changing the learning rate too quickly.

2.8 Neural Network Models

2.8.1 Long Short-Term Memory

LSTMs are designed to avoid the problem of long-term dependency since they can remember information for long periods of time.

The key difference from vanilla RNNs and LSTMs is the cell state that is introduced.

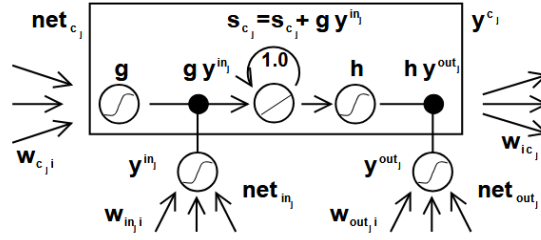


Figure 2.19. Architecture of the memory cell [11].

The first step taken in an LSTM is to decide which information is to be discarded from the cell state in Figure 2.19. This is done through the sigmoid layer, also known as the forget gate layer, by looking at h_{t-1} and x_t in the following formula:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad [12]$$

Shown in Figure 2.20 graphically:

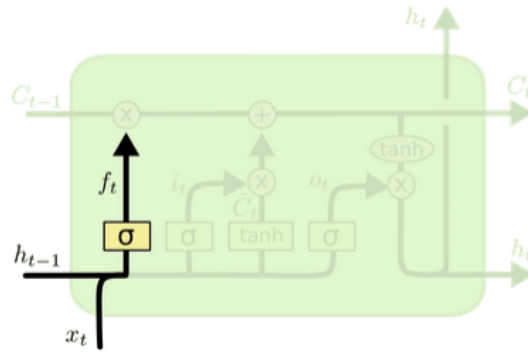


Figure 2.20. Forget Gate Layer [12].

Next, the new information is analyzed to see if it will be stored or not with the following formulas:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \quad [12]$$

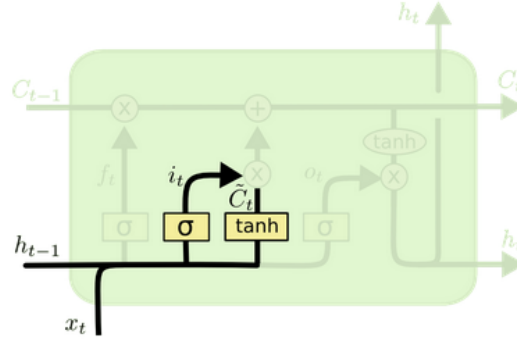


Figure 2.21. Input Gate Layer [12].

Figure 2.21 illustrates the use of the sigmoid layer, also known as the input gate layer, that decides which values to update. Using the tanh layer to create a vector that will be added to the input gate layer for the next step of updating the values.

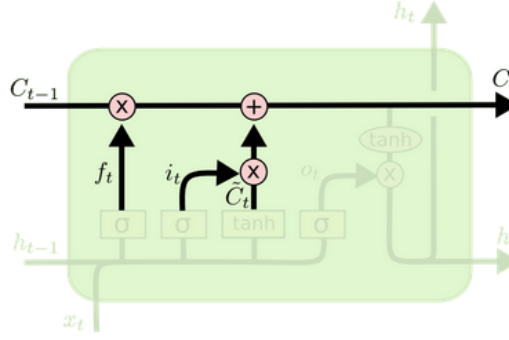


Figure 2.22. Updating the cell state [12].

Updating the cell state with the following formula:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad [12]$$

Finally, the output must be calculated using the following formulas:

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \cdot \tanh(C_t) \end{aligned} \quad [12]$$

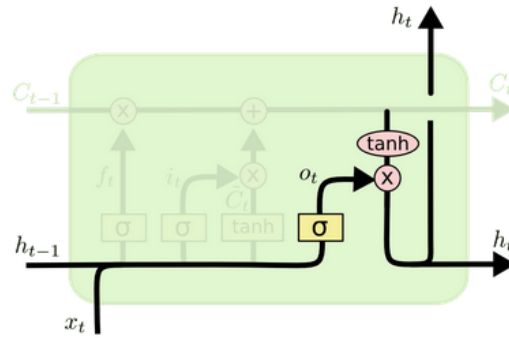


Figure 2.23. LSTM Output [12].

As can be seen from Figure 2.23, the sigmoid layer decides which parts will be the output, then the values are pushed to -1 and 1 using the tanh function, and finally it is multiplied by the output of the sigmoid gate.

2.8.2 Gated Recurrent Unit

GRUs in Figure 2.24 are an RNN model similar to LSTMs in that they use gating units. These gates modify the information flowing in the unit, but unlike LSTMs, GRUs do not have memory cells as shown in the figure below:

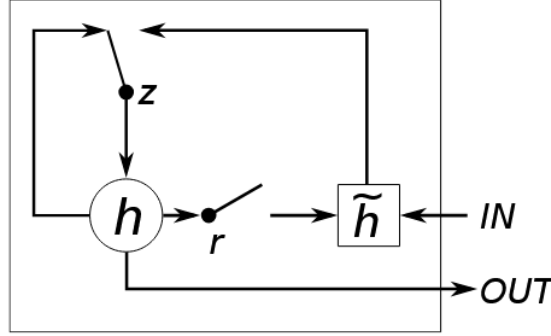


Figure 2.24. Gated Recurrent Unit [13].

While LSTMs contain three gates and two states (the hidden state and cell state), GRUs only use two gates and one state (hidden state), reducing the complexity and training faster while still producing comparative results.

The following formulas calculate the hidden state recurrence $\mathbf{h}_t$, the update gate $\mathbf{z}_t$, the reset gate $\mathbf{r}_t$, and the candidate gate $\tilde{\mathbf{h}}_t$:

$$\begin{aligned}\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \\ \mathbf{z}_t &= \sigma(\text{Linear}_{d_h}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \\ \mathbf{r}_t &= \sigma(\text{Linear}_{d_h}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \\ \tilde{\mathbf{h}}_t &= \tanh(\text{Linear}_{d_h}([\mathbf{x}_t, \mathbf{r}_t \odot \mathbf{h}_{t-1}]))\end{aligned}\tag{13}$$

GRU combines the forget gate and input gate that LSTM uses into a single update gate $\mathbf{z}_t$ that carries the past information forward to the next calculation. GRU also removes LSTMs output gate and replaces it with a reset gate $\mathbf{r}_t$. This gate controls the amount of the hidden state $\mathbf{h}_{t-1}$ is used when computing the candidate hidden state $\tilde{\mathbf{h}}_t$.

2.8.3 minLSTM

Following the same formula for LSTMs:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

but modifying the other formulas as follows:

$$\begin{aligned}\mathbf{f}_t &= \sigma(\text{Linear}_{d_h}(\mathbf{x}_t)) \\ \mathbf{i}_t &= \sigma(\text{Linear}_{d_h}(\mathbf{x}_t)) \\ \tilde{\mathbf{h}}_t &= \tanh(\text{Linear}_{d_h}(\mathbf{x}_t))\end{aligned}\tag{14}$$

The hidden state dependencies $\mathbf{h}_{t-1}$ have been removed. This allows for parallel and sequential training.

2.8.4 minGRU

Following the same formula for hidden state as GRU:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

but modifying the other formulas as follows:

$$\begin{aligned} \mathbf{z}_t &= \sigma(\text{Linear}_{d_h}(\mathbf{x}_t)) \\ \tilde{\mathbf{h}}_t &= \tanh(\text{Linear}_{d_h}(\mathbf{x}_t, \mathbf{r}_t)) \end{aligned} \quad [14]$$

Since $\mathbf{h}_{t-1}$ has been removed, the reset gate $\mathbf{r}_t$ is no longer needed. These calculations can also be done in parallel, as well as sequentially now.

2.8.5 Extended Long Short-Term Memory

xLSTMs try to surpass the problems associated to LSTMs by introducing two main modifications to the following formula:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

Exponential gating and novel memory structures by adding both sLSTM's "with scalar memory, a scalar update" [15] and mLSTM's "with a matrix memory and a covariance (outer product) update rule, which is fully parallelizable" [15].

Figure 2.25 illustrates the forward pass of the sLSTM:

$c_t = f_t \cdot c_{t-1} + i_t \cdot z_t$		cell state
$n_t = f_t \cdot n_{t-1} + i_t$		normalizer state
$h_t = o_t \cdot \tilde{h}_t$	$\tilde{h}_t = c_t / n_t$	hidden state
$z_t = \varphi(\tilde{z}_t)$	$\tilde{z}_t = \mathbf{w}_z^\top \mathbf{x}_t + r_z h_{t-1} + b_z$	cell input
$i_t = \exp(\tilde{i}_t)$	$\tilde{i}_t = \mathbf{w}_i^\top \mathbf{x}_t + r_i h_{t-1} + b_i$	input gate
$f_t = \sigma(\tilde{f}_t) \text{ OR } \exp(\tilde{f}_t)$	$\tilde{f}_t = \mathbf{w}_f^\top \mathbf{x}_t + r_f h_{t-1} + b_f$	forget gate
$o_t = \sigma(\tilde{o}_t)$	$\tilde{o}_t = \mathbf{w}_o^\top \mathbf{x}_t + r_o h_{t-1} + b_o$	output gate

Figure 2.25. sLSTM forward pass [15].

Since exponential activation functions can cause overflows, $\mathbf{m}_t$ in Figure 2.26 was introduced to stabilize the system:

$m_t = \max(\log(f_t) + m_{t-1}, \log(i_t))$		stabilizer state
$i'_t = \exp(\log(i_t) - m_t) = \exp(\tilde{i}_t - m_t)$		stabil. input gate
$f'_t = \exp(\log(f_t) + m_{t-1} - m_t)$		stabil. forget gate

Figure 2.26. Stabilizing state m_t [15].

Figure 2.27 demonstrates the forward pass of the mLSTM:

$$\begin{aligned}
C_t &= \hat{f}_t \odot C_{t-1} + \hat{i}_t v_t k_t^T && \text{cell state} \\
n_t &= \hat{f}_t n_{t-1} + \hat{i}_t k_t && \text{normalizer state} \\
h_t &= o_t \odot \tilde{h}_t, && \tilde{h}_t = C_t q_t / \max\{|n_t^T q_t|, 1\} && \text{hidden state} \\
q_t &= W_q x_t + b_q && \text{query input} \\
k_t &= \frac{1}{\sqrt{d}} W_k x_t + b_k && \text{key input} \\
v_t &= W_v x_t + b_v && \text{value input} \\
\hat{i}_t &= \exp(\tilde{i}_t), && \tilde{i}_t = w_i^T x_t + b_i && \text{input gate} \\
\hat{f}_t &= \sigma(\tilde{f}_t) \text{ OR } \exp(\tilde{f}_t), && \tilde{f}_t = w_f^T x_t + b_f && \text{forget gate} \\
o_t &= \sigma(\tilde{o}_t), && \tilde{o}_t = W_o x_t + b_o && \text{output gate}
\end{aligned}$$

Figure 2.27. mLSTM Forward Pass [15].

Figure 2.28 displays how the original LSTM and its memory cells become sLSTM and mLSTM. These all combine into the xLSTM blocks, and finally, they produce stacks of xLSTM blocks:

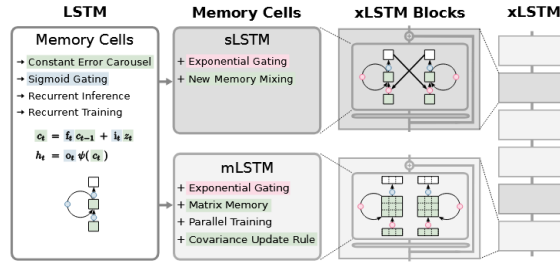


Figure 2.28. xLSTM Architecture [15].

2.8.6 Bidirectional

Two NNs were also tested using bidirectional, which means that the data is processed once in the forward direction and again in the backward direction. They take longer to process but usually produce a better result. Bidirectional was only used with GRU and LSTM networks.

2.8.7 Convolutional Neural Network

A CNN, also known as a ConvNet, is a NN mainly used for object recognition, such as image classification, detection, and segmentation. It can also be stacked upon RNN layers to produce similar time-series computations. CNNs consist of four main components:

1. Convolutional layers.
2. ReLU.
3. Pooling layers.
4. Fully Connected (FC) layers.

Convolutional Layers

In this process, the main action is convolution where a sliding window function is applied to a matrix of pixels which represent the image. This sliding window is called the kernel or filter. Filters are used to recognize specific patterns such as curving, edges, whole shapes, and more.

Rectified Linear Unit

The ReLU activation function in Figure 2.5 is used after each convolutional layer. By applying this function, the network can more effectively learn non-linear relationships, and it also helps eliminate the vanishing gradient problem.

Max Pooling

Max pooling in Figure 2.29 is an operation used to downsample by reducing the spatial dimensions by retaining only the important features, as can be seen from the image below:

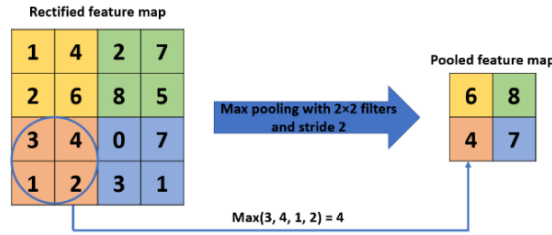


Figure 2.29. Max Pooling Operation [16].

In Figure 2.30, the convolutional network is shown with the main features of the convolutional layer with a 3x3 kernel, ReLU, and MaxPool1D with a 2x2 kernel.

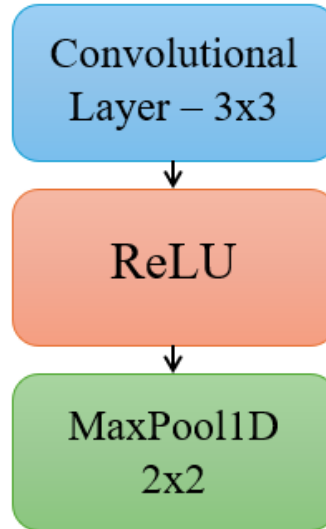


Figure 2.30. Convolutional Network Stack.

2.9 Inference time

Inference time, also known as inference latency, is the time a machine learning system requires to produce a value after receiving an input.

2.10 Applications of Recurrent Neural Networks

Possible uses for RNNs include, but are not limited to:

- Natural Language Processing (NLP): it can predict the next word in a sentence (used in programs like Microsoft Word). RNNs can also translate words by encoding the input sentence and decoding it in the target language, and sentiment analysis by capturing context information from words in a sequence.
- Speech Recognition and Synthesis: RNNs can be used to convert spoken language into written text, as well as, text to speech.
- Time-Series Analysis Forecasting: Financial forecasting can be done with stock prices, currency exchange rates, and even other financial variables. RNNs can also be used to predict the weather using historical weather data.
- Music Generation: RNNs have the ability to produce music by learning patterns from existing music and producing music sequences.
- Robotics and Autonomous Systems: RNNs can be used for path prediction and also gesture control to enable a more natural interaction with robots.

Chapter 3

Experiments

This thesis was built on the foundations created by Fabio Amoretti [17], who, with the help of his supervisors, built the test bench for the study of small-scale electric propulsion systems. From this foundation, three additional sensors were added to the current cables. These and the pre-existing sensors (fiber optic sensors on the temperature, top and bottom torque) allowed the capture of data in order to train an RNN model to virtually sense optimal temperatures during its function. The software used was Python to create the program for RNN models and Arduino IDE in order to capture the data from the test bench. The interrogator used its own software to create a log file for the temperature and both torques.

3.1 Equipment

3.1.1 Sensors

Fiber Bragg Grating

The fiber optic sensors used on the test bench were Fiber Bragg Grating (FBG). They work by modifying the original optical fiber by having a periodic variation in the refractive index along the core, as shown in the image below:

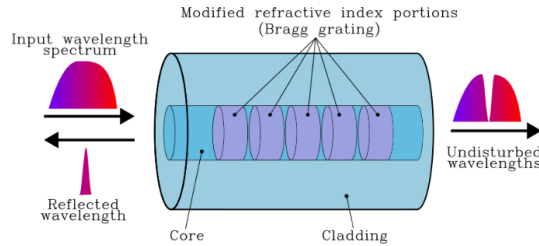


Figure 3.1. Fiber Bragg Grating Schematic [17].

This periodic spacing is known as the Bragg wavelength, and its operating principle can be shown in the formula below:

$$\Delta\lambda_B = \mathbf{K}_\epsilon \Delta\epsilon + \mathbf{K}_T \Delta T$$

Where $\Delta\lambda_B$ is the shift in the Bragg wavelength, $\mathbf{K}_\epsilon$ and $\mathbf{K}_T$ are the strain and temperature sensitivity coefficients. The applied strain is represented by $\Delta\epsilon$, and ΔT is the variation in temperature. In order to find a uniform distance for each Bragg wavelength, a laser-modified segment is used and denoted by Λ .

$$\lambda_B = 2 \cdot \eta_{EFF} \cdot \Lambda$$

Along with the optical fiber construction:

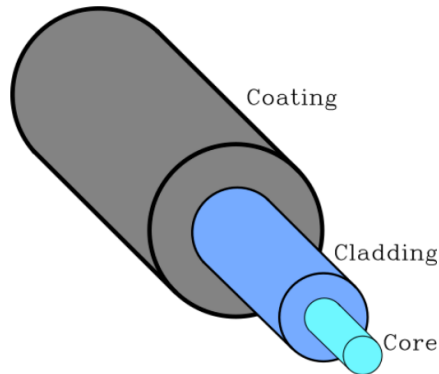


Figure 3.2. Structure of an optical fiber cable [17].

Digital Sensors

This sensor measures the shunt voltage, bus voltage, current and power using its integrated circuit. The first sensors installed on the current lines were INA219. They are shown in 3.3:

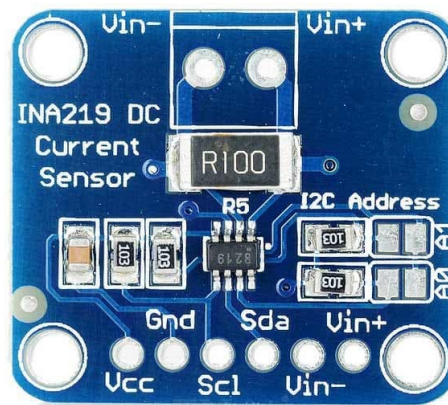


Figure 3.3. INA219 Digital Current Sensor [18].

After thorough testing, it was found that using the 3-phase motor was not compatible with these sensors, as the initial spike of current would send the sensors into error mode, and no measurements could be taken.

Analog Sensors

This sensor utilizes the Hall effect to measure the magnetic field that the flowing current generates through the internal copper conductor. These sensors allowed the capture of all the necessary data used in this thesis.

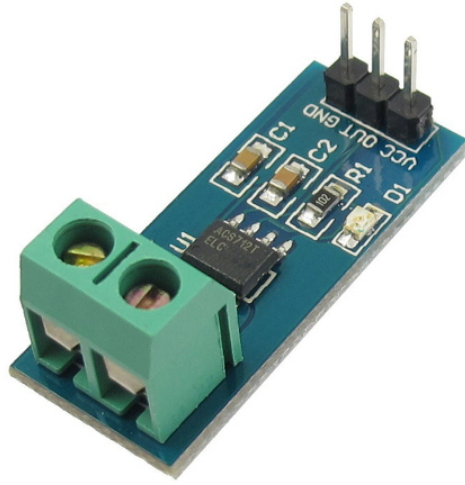


Figure 3.4. ACS712 Analog Current Sensor [19].

3.1.2 Test Bench with Arduino Due

In Figures 3.5 and 3.6, the test bench used to simulate a small-scale electric propulsion system is shown. Figure 3.5 highlights the protective black cover shielding users from the rotating shaft. The Arduino Due, located in the upper-left corner, provides control of the machine, while the resistor assembly on the far right simulates variable loads on the system over a range of 0Ω to 47.5Ω .

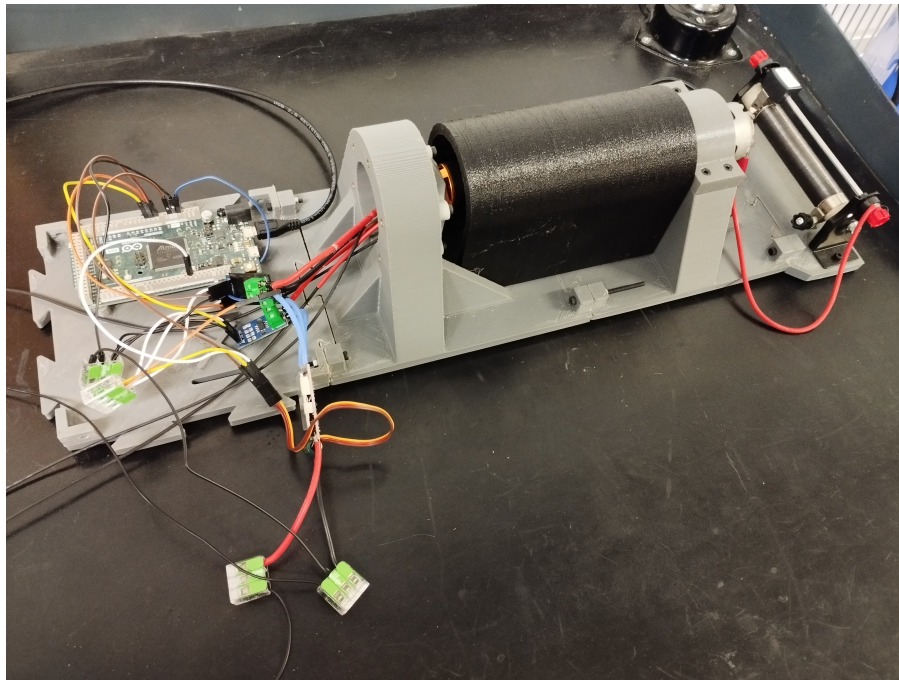


Figure 3.5. Test Bench used to take measurements with cover.

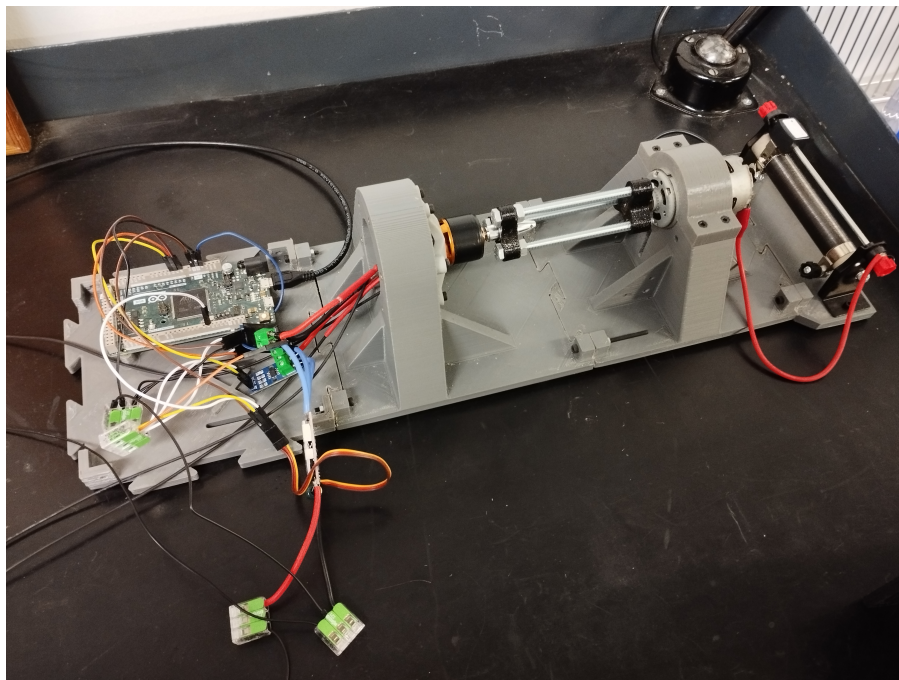


Figure 3.6. Test Bench used to take measurements without cover

3.1.3 Interrogator

The SmartScan interrogator is a compact and robust instrument used for dynamic FBG sensor measurement.



Figure 3.7. Interrogator [20].

3.2 Procedure

This project started with the installation of the INA219 digital sensors, which were found to be unsuitable for the 3-phase motor. Afterwards, ACS712 analog sensors were used to properly capture the necessary data for the testing of NN models.

Once the sensors were installed, the previous Arduino Integrated Development Environment (IDE) script for the test bench was modified to work correctly with the new analog sensors. Then a Python script was developed to allow the throttle to be modified while running (ranging Revolutions Per Minutes (RPMs) of 1450 - 1610), while also creating an Excel of all the data processed through the sensors. The previous Arduino IDE script did not provide the Microsoft Excel creation feature.

The data from the Arduino was captured alongside the interrogator, which captured the top and bottom torques and temperature separately. This data was put into a .log file. These files were put into a new Excel file in order to be pre-processed.

A new Python script was created to combine the Arduino Excel file and the interrogator Excel file. Once complete, it included ElapsedTime(s), Interpolated Current1, Interpolated Current2, Interpolated Current3, Motor Temperature, Bottom Torque, Top Torque, RPM, and Resistance (Ohm) for each test. The tests were between 5 and 10 minutes long.

Once all the tests were completed, the data captured, and the files combined, a new Python script was created to combine all 25 tests into one large file containing over 700,000 lines of data. Since this is a large amount of data, another shorter file containing the first 2 minutes of each test was created for training purposes.

Another Python script was created to test the NNs. There were 10 different types of recurrent layers, which include - Vanilla RNN, LSTM, GRU, BiGRU, BiLSTM, minGRUseq, minGRUpair, minLSTMseq, minLSTMpair, and finally xLSTM, which were all explained in the theory chapter. This final script includes features for early stopping, LR scheduling and check-pointing.

Early stopping is a PyTorch function that stops the program if no improvements have been made within a certain number of epochs, also known as patience, which was set to 3 in this thesis. This signifies that after 3 values higher than the lowest, the program stops.

LR scheduling is another PyTorch function that enables the reduction of the LR after a certain number of epochs that do not see improvement during training. This function goes hand in hand with early stopping.

Check-pointing is a feature in PyTorch that saves the point during training at which the value is the best (closest to zero in the case of this thesis, as RMSE values were used). Once this value is saved, it will be compared to the final epoch of testing, once training and validation are done.

The structure of this final Python script is as follows:

- Import all necessary libraries.
- Initialize starting time to find testing times.
- Make sure the device uses any Graphics Processing Unit (GPU).
- Load necessary Microsoft Excel files.
- Create proper input and output columns.
- Define whether or not convolutional layers are to be added.
- Define which recurrent layers to test.
- Set hyperparameters.
- Initialize early stopping.
- Initialize Check-point Manager.
- Process the data into proper sequence lengths.
- Split the data into train, validation, and test datasets.
- Introduce all necessary NN models.
- Introduce appropriate loss functions.
- Initialize train and evaluate functions.
- Initialize main training loop.
- Initialize final testing loop.
- Save all loss function values.
- Create all necessary graphs and plots.

Before testing each recurrent layer, the proper hyperparameters were found through experimentation. Since the dataset was large, the smaller two-minute Excel file was used with the two fastest recurrent layers (minGRUpar and minLSTMpar).

The first set of tests were done by stacking convolutional layers on top of the existing recurrent layers. The process for finding the best hyperparameters went in the following order:

1. LR.
2. Batch Size.
3. Hidden Size.
4. Number of layers.

3.3 Condition Monitoring and Filtering

3.3.1 (Convolutional Layer + ReLU + MaxPool) x 1 + Recurrent Layers

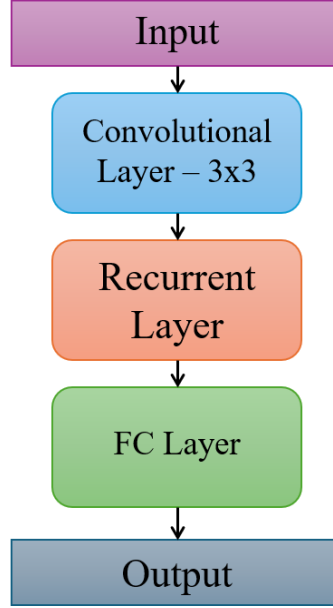


Figure 3.8. Recurrent and convolutional flowchart.

Starting with a LR of 1×10^{-1} and doing 8 tests of decreasing order:

RMSE Values				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.165434268	0.180056447	68.57	101.39
1×10^{-2}	0.163654772	0.178883801	96.44	126.29
1×10^{-3}	0.168685255	0.181138145	292.72	164.91
1×10^{-4}	0.164142318	0.172230679	865.23	345.09
1×10^{-5}	0.164497643	0.178625064	451.15	784.44
1×10^{-6}	0.164150967	0.173059131	554.44	1715.31
1×10^{-7}	0.162394468	0.170179217	823.97	5918.26
1×10^{-8}	0.163277058	0.17515371	3030.1	2851.42

Table 3.1. RMSE values and training times for different LRs.

From Table 3.1, the LR of 1×10^{-2} was chosen as it produced the best compromise of time and RMSE value. In 1×10^{-7} , minLSTM went for the entire 1000 epochs.

MAPE Values (%)				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.005441864	0.005725546	68.57	101.39
1×10^{-2}	0.00537667	0.005664748	96.44	126.29
1×10^{-3}	0.005572841	0.005811223	292.72	164.91
1×10^{-4}	0.005403522	0.005614073	865.23	345.09
1×10^{-5}	0.005416824	0.005688307	451.15	784.44
1×10^{-6}	0.005389463	0.005598958	554.44	1715.31
1×10^{-7}	0.00534052	0.00554428	823.97	5918.26
1×10^{-8}	0.005352355	0.005589679	3030.1	2851.42

Table 3.2. MAPE values and training times for different LRs.

Next, the batch size was iterated through, starting from 16.

RMSE Values				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.162146452	0.173740679	684.48	636.41
32	0.161246504	0.169001611	249.74	282.4
64	0.160901925	0.174171117	152.33	156.6
128	0.163514036	0.174735543	104.49	131.36
256	0.163392902	0.171043675	96.93	120.14
512	0.163817767	0.174257504	113.56	142.36
768	0.162112539	0.170185625	82.49	89.76
1024	0.163654772	0.178883801	96.44	126.29

Table 3.3. RMSE values and training times for different batch sizes.

MAPE Values (%)				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.005355551	0.005590358	684.48	636.41
32	0.005311112	0.005519811	249.74	282.4
64	0.005294301	0.005558273	152.33	156.6
128	0.005394238	0.005632979	104.49	131.36
256	0.005389063	0.005583668	96.93	120.14
512	0.005396796	0.005632129	113.56	142.36
768	0.005331315	0.005542971	82.49	89.76
1024	0.00537667	0.005664748	96.44	126.29

Table 3.4. MAPE values and training times for different batch sizes.

After performing the above tests, the batch size was chosen to be 768 since it produced good RMSE and MAPE values with the best training times.

Following the batch size tests, hidden sizes were computed:

RMSE Values				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.16078593	0.160821007	113.75	145.72
32	0.159980112	0.159973814	92.8	89.73
64	0.163061723	0.16309157	81.54	78.67
128	0.160145769	0.034671879	85.41	412.4
256	0.162112539	0.170185625	82.49	89.76
512	0.162378604	0.075090399	300.66	632.09
768	0.16450962	0.038129288	415.89	1056.6
1024	0.035593651	0.026112959	1136.78	1844.4

Table 3.5. RMSE values and training times for different hidden sizes.

MAPE Values (%)				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.006231335	0.006260205	113.75	145.72
32	0.006278547	0.006255587	92.8	89.73
64	0.006399248	0.006338409	81.54	78.67
128	0.006277618	0.001442805	85.41	412.4
256	0.005331315	0.005542971	82.49	89.76
512	0.006326903	0.00318732	300.66	632.09
768	0.006390671	0.001715639	415.89	1056.6
1024	0.001419622	0.001239776	1136.78	1844.4

Table 3.6. MAPE values and training times for different hidden sizes.

From Table 3.5 and Table 3.6, the hidden size value of 128 was chosen. Hidden size of 1024 produced better values but with considerably longer training times.

Finally, the number of layers was chosen by iterating from 1 through 7:

RMSE Values				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.160145769	0.034671879	85.41	412.4
2	0.160346626	0.160507711	97.11	119.08
3	0.162620351	0.162620023	90.78	134.02
4	0.161561793	0.161637209	77.39	239.72
5	0.026741481	0.161807778	380.16	206.4
6	0.031022825	0.160877902	430.97	239.55
7	0.160582498	0.160769955	97.73	275.68

Table 3.7. RMSE values and training times for different numbers of layers.

MAPE Values (%)				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.006277618	0.001442805	85.41	412.4
2	0.006142531	0.006245725	97.11	119.08
3	0.006274206	0.006302726	90.78	134.02
4	0.006271278	0.006264809	77.39	239.72
5	0.000951997	0.006305146	380.16	206.4
6	0.001078731	0.006269124	430.97	239.55
7	0.006308342	0.006259085	97.73	275.68

Table 3.8. MAPE values and training times for different numbers of layers.

Consulting Table 3.7 and Table 3.8, 5 layers produced the best RMSE and MAPE values for minGRU but 1 layer produced the best results for minLSTM. In the end, 5 layers was chosen.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Test RMSE	Test MAPE (%)	Time (s)
RNN	0.166559976	0.0060842	328.15
GRU	0.166587006	0.00606097	707.42
LSTM	0.166585872	0.00607848	922.25
BiGRU	0.166588601	0.006143295	1305.63
BiLSTM	0.166586161	0.006072643	2255.29
minGRUseq	0.026159278	0.001081351	683.17
min	0.042024089	0.001677121	414.46
minLSTMseq	0.025813607	0.001058345	787.26
minLSTMpar	0.166598886	0.006196748	908.38
xLSTM	0.166585612	0.006101989	951.19

Table 3.9. Comparison of models in terms of test RMSE, test MAPE, and runtime.

3.3.2 (Convolutional Layer + ReLU + MaxPool) x 2 + Recurrent Layers

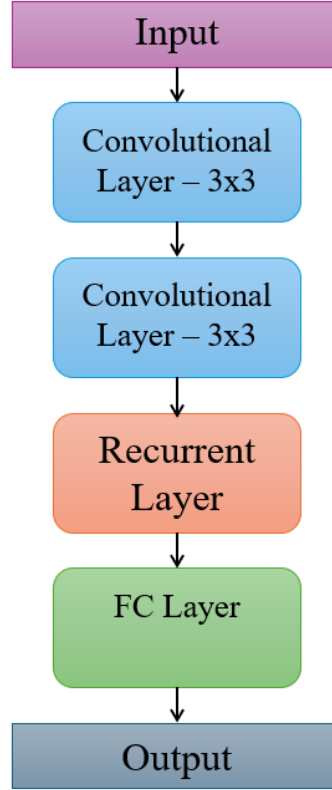


Figure 3.9. Recurrent and convolutional flowchart.

Starting with a LR of 1×10^{-1} and doing 7 tests of decreasing order:

RMSE Values				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.167478853	0.167460648	76.03	67.86
1×10^{-2}	0.161813741	0.161797727	94.34	63.06
1×10^{-3}	0.162356912	0.162339888	106.73	87.69
1×10^{-4}	0.162743285	0.162726261	168.63	184.82
1×10^{-5}	0.165738923	0.165721043	401.49	437.02
1×10^{-6}	0.16113642	0.161120417	504.15	564.88
1×10^{-7}	0.161233707	0.161217487	707.13	801.2

Table 3.10. RMSE values and training times for different LRs.

MAPE Values (%)				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.005549783	0.005553616	76.03	67.86
1×10^{-2}	0.005315389	0.005319216	94.34	63.06
1×10^{-3}	0.005335288	0.005339051	106.73	87.69
1×10^{-4}	0.005353946	0.005357735	168.63	184.82
1×10^{-5}	0.005481114	0.005484929	401.49	437.02
1×10^{-6}	0.005266093	0.005269869	504.15	564.88
1×10^{-7}	0.005278205	0.005282039	707.13	801.2

Table 3.11. MAPE values and training times for different LR.

From Table 3.10 and Table 3.11, the LR of 1×10^{-2} was chosen as it produced the best compromise of time and RMSE value.

Next, the batch size was iterated through, starting from 16.

RMSE Values				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.164355396	0.164337622	659.1	551.56
32	0.162018033	0.162001132	267.5	297.74
64	0.163465702	0.163448535	165.81	146.04
128	0.161619017	0.161602531	107.86	95.19
256	0.161716077	0.161700051	81.13	75.44
512	0.16364291	0.163625673	73.03	73.91
768	0.162622616	0.16260596	75.02	96.26
1024	0.161813741	0.161797727	94.34	63.06

Table 3.12. RMSE values and training times for different batch sizes.

MAPE Values (%)				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.005399187	0.005402954	659.1	551.56
32	0.005347342	0.005351077	267.5	297.74
64	0.005353705	0.005357442	165.81	146.04
128	0.005320108	0.005323937	107.86	95.19
256	0.005321114	0.005324989	81.13	75.44
512	0.005389616	0.005393372	73.03	73.91
768	0.005345737	0.00534955	75.02	96.26
1024	0.005335288	0.005339051	94.34	63.06

Table 3.13. MAPE values and training times for different batch sizes.

After considering Table 3.12 and Table 3.13, the chosen batch size with comparable speeds and relatively low RMSE values was 128.

Following the batch size tests, hidden sizes were computed:

RMSE Values				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.159551644	0.012494116	209.34	530.72
32	0.013890669	0.019180654	271.18	308.52
64	0.015523442	0.019811193	299.9	247.96
128	0.029403493	0.031166317	220.11	206.89
256	0.161619017	0.161602531	107.86	95.19
512	0.025655187	0.035342486	212.54	254.93
768	0.03408068	0.034346784	301.63	398.57
1024	0.029340316	0.032167001	404.01	534.22

Table 3.14. RMSE values and training times for different hidden sizes.

MAPE Values (%)				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.006223002	0.000457469	209.34	530.72
32	0.000534296	0.000585377	271.18	308.52
64	0.000475298	0.000695654	299.9	247.96
128	0.001111296	0.001185353	220.11	206.89
256	0.005320108	0.005323937	107.86	95.19
512	0.00096662	0.001261151	212.54	254.93
768	0.001203848	0.001289796	301.63	398.57
1024	0.00099748	0.001153773	404.01	534.22

Table 3.15. MAPE values and training times for different hidden sizes.

From Table 3.14 and Table 3.15, the hidden size value of 32 was chosen.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

RMSE Values				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.013890669	0.019180654	271.18	308.52
2	0.00639064	0.162012254	205.39	108.4
3	0.161777519	0.161776251	119.46	127.86
4	0.162352215	0.162351061	155.66	190.08
5	0.16219022	0.162201644	208.77	233.48
6	0.160098828	0.1601408	266.44	267.76
7	0.161972855	0.161974952	236.79	342.7

Table 3.16. RMSE values and training times for different numbers of layers.

MAPE Values (%)				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.000534296	0.000585377	271.18	308.52
2	0.000264693	0.00623299	205.39	108.4
3	0.006199155	0.006193665	119.46	127.86
4	0.006330182	0.006358333	155.66	190.08
5	0.006173359	0.00621732	208.77	233.48
6	0.005974621	0.006055092	266.44	267.76
7	0.006255999	0.006295007	236.79	342.7

Table 3.17. MAPE values and training times for different numbers of layers.

After testing all of these layer stacks, 2 layers produced the best RMSE and MAPE values.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Test RMSE	Test MAPE (%)	Time (s)
RNN	0.168764	0.005207	937.68
GRU	0.006455	0.000253	871.89
LSTM	0.008140	0.000281	876.74
BiGRU	0.006043	0.000251	752.73
BiLSTM	0.006196	0.000271	904.14
minGRUseq	0.012894	0.000426	1648.43
minGRUpar	0.041215	0.001739	634.27
minLSTMseq	0.007215	0.000264	1696.33
minLSTMpar	0.167392	0.006112	476.98
xLSTM	0.007648	0.000250	3816.41

Table 3.18. Comparison of models in terms of test RMSE, test MAPE, and runtime.

3.3.3 (Convolutional Layer + ReLU + MaxPool) x 3 + Recurrent Layers

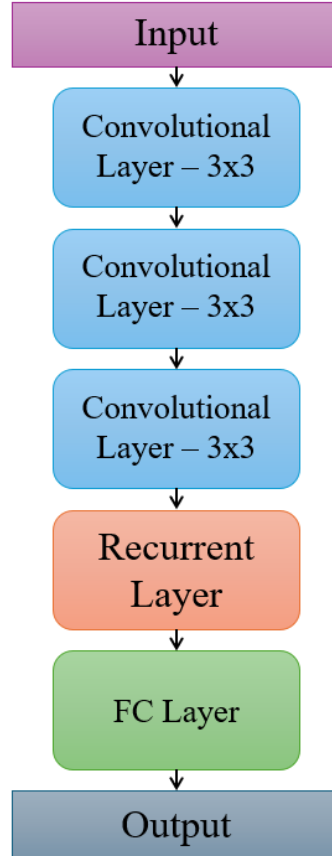


Figure 3.10. Recurrent and convolutional flowchart.

Starting with a LR of 1×10^{-1} and doing 8 tests of decreasing order:

RMSE Values				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.160886424	0.16080549	50.26	60.52
1×10^{-2}	0.162666517	0.162581804	80.26	75.56
1×10^{-3}	0.162602781	0.162521043	220.59	252.67
1×10^{-4}	0.15978434	0.159707536	369.43	404.08
1×10^{-5}	0.162955608	0.162869972	442.77	551.5
1×10^{-6}	0.164105338	0.164016661	704.43	731.96
1×10^{-7}	0.162345546	0.162262577	46.8	46.17
1×10^{-8}	0.164238134	0.164149844	59.43	58.26

Table 3.19. RMSE values and training times for different LR.

MAPE Values (%)				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.005253585	0.005272699	50.26	60.52
1×10^{-2}	0.005346823	0.005365759	80.26	75.56
1×10^{-3}	0.00532492	0.005344175	220.59	252.67
1×10^{-4}	0.005234999	0.005254507	369.43	404.08
1×10^{-5}	0.005350523	0.005369519	442.77	551.5
1×10^{-6}	0.005385111	0.005403745	704.43	731.96
1×10^{-7}	0.005318394	0.005337529	46.8	46.17
1×10^{-8}	0.005393533	0.005412275	59.43	58.26

Table 3.20. MAPE values and training times for different LRs.

From Table 3.19 and Table 3.20, the LR of 1×10^{-4} was chosen as it produced the best compromise of time and RMSE value. For the LR 1×10^{-7} and 1×10^{-8} , the delta for early stopping was changed to 1×10^{-7} because the training was going for too many epochs without any noticeable change.

Next, the batch size was iterated through, starting from 16.

RMSE Values				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.164185757	0.164100374	704.09	733.81
32	0.162682618	0.162598034	357.73	386.89
64	0.165808	0.165719226	201.38	207.02
128	0.162394322	0.16231054	149.22	133.95
256	0.163658473	0.163572019	136.05	132.8
512	0.163375397	0.163289069	187.91	195
768	0.163195033	0.163112561	252.13	262.66
1024	0.15978434	0.159707536	369.43	404.08

Table 3.21. RMSE values and training times for different batch sizes.

MAPE Values (%)				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.005402739	0.005421616	704.09	733.81
32	0.005338456	0.005357337	357.73	386.89
64	0.005446307	0.00546534	201.38	207.02
128	0.005326036	0.005344994	149.22	133.95
256	0.005382477	0.005401203	136.05	132.8
512	0.005353097	0.005372065	187.91	195
768	0.00533934	0.005358546	252.13	262.66
1024	0.005234999	0.005254507	369.43	404.08

Table 3.22. MAPE values and training times for different batch sizes.

After analyzing Table 3.21 and Table 3.22, the chosen batch size was 1024.

Following the batch size tests, hidden sizes were computed:

RMSE Values				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	N/A	N/A	N/A	N/A
32	N/A	N/A	N/A	N/A
64	0.160779594	0.035695906	1429.41	1892.3
128	0.163195033	0.163112561	252.13	262.66
256	0.024758631	0.029748679	778.26	951.13
512	0.158524866	0.029410662	430.22	1001.79
768	0.025730149	0.026913749	1038.66	1345.57
1024	0.030165646	0.027764314	1437.58	1712.94

Table 3.23. RMSE values and training times for different hidden sizes.

MAPE Values (%)				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	N/A	N/A	N/A	N/A
32	N/A	N/A	N/A	N/A
64	0.006257285	0.001450478	1429.41	1892.3
128	0.00533934	0.005358546	252.13	262.66
256	0.000805898	0.001194783	778.26	951.13
512	0.006152693	0.00103323	430.22	1001.79
768	0.000938561	0.00094881	1038.66	1345.57
1024	0.001158183	0.000994182	1437.58	1712.94

Table 3.24. MAPE values and training times for different hidden sizes.

From Table 3.23 and Table 3.24, the hidden size value of 256 was chosen. For hidden sizes 16 and 32, the test reached 1000 epochs without producing a value lower than 1500.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

RMSE Values				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.024758631	0.029748679	778.26	951.13
2	0.026348148	0.027570868	601.79	949.14
3	0.15610237	0.161447106	490.48	824.37
4	0.161268799	0.161269741	455.97	1016.85
5	0.15842083	0.158420844	489.73	1016.53
6	0.034491171	0.158887897	683.51	1274
7	0.032692719	0.162254643	738.24	1334.6

Table 3.25. RMSE values and training times for different numbers of layers.

MAPE Values (%)				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.000805898	0.001194783	778.26	951.13
2	0.000968875	0.001039136	601.79	949.14
3	0.006419919	0.006323955	490.48	824.37
4	0.0062282	0.006217076	455.97	1016.85
5	0.006092283	0.006092288	489.73	1016.53
6	0.001303369	0.006208	683.51	1274
7	0.001172206	0.006332971	738.24	1334.6

Table 3.26. MAPE values and training times for different numbers of layers.

After testing all these layer numbers, 2 produced the best RMSE and MAPE values.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Test RMSE	Test MAPE (%)	Time (s)
RNN	0.015025572	0.000581157	1081.34
GRU	0.013920791	0.000546532	1068.07
LSTM	0.013811874	0.000576055	1227.66
BiGRU	0.012511244	0.000529129	1412.52
BiLSTM	0.013294844	0.000552232	1623.71
minGRUseq	0.018723753	0.000680519	1502.56
minGRUpar	0.013484591	0.000512774	750.91
minLSTMseq	0.016956062	0.000658493	1443.8
minLSTMpar	0.015708951	0.00060212	600.99
xLSTM	0.013608992	0.000549438	1283.73

Table 3.27. Comparison of models in terms of test RMSE, test MAPE, and runtime.

3.3.4 Recurrent Layers

The following tests took place with just the recurrent layers:

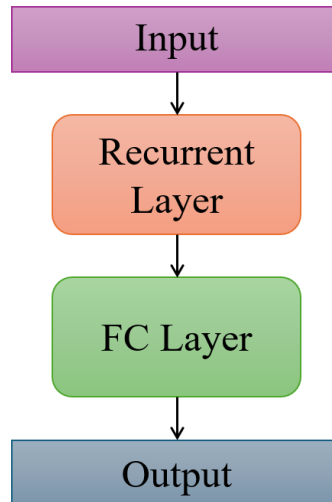


Figure 3.11. Recurrent flowchart.

Starting with a LR of 1×10^{-2} and doing 7 tests of decreasing order:

RMSE Values				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-2}	0.033183597	0.061358671	247.82	293.81
1×10^{-3}	0.028471221	0.053791006	290.66	305.23
1×10^{-4}	0.028515224	0.053198929	178.97	224.15
1×10^{-5}	0.028940727	0.053532953	109.86	152.12
1×10^{-6}	0.030004521	0.054189395	183.21	195.44
1×10^{-7}	0.027621162	0.052318716	275.29	107
1×10^{-8}	0.029483945	0.054992003	271.81	107.87

Table 3.28. RMSE values and training times for different LRs.

MAPE Values (%)				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-2}	0.001288771	0.002794539	247.82	293.81
1×10^{-3}	0.001120727	0.002420872	290.66	305.23
1×10^{-4}	0.001086432	0.00240024	178.97	224.15
1×10^{-5}	0.001096884	0.002402436	109.86	152.12
1×10^{-6}	0.001119898	0.002428136	183.21	195.44
1×10^{-7}	0.001100419	0.002406615	275.29	107
1×10^{-8}	0.001115097	0.002478208	271.81	107.87

Table 3.29. MAPE values and training times for different LRs.

From Table 3.28 and Table 3.29, a LR of 1×10^{-7} was chosen until hidden sizes were tested and even after running the tests with 1000 epochs, values could not be found so a lower LR was chosen at 1×10^{-4} .

Next, the batch size was iterated through, starting from 16.

RMSE Values				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.027842071	0.053404972	2165.93	992.68
32	0.028165635	0.053586421	1274.95	590.08
64	0.030498921	0.05558802	700.98	492.07
128	0.029479436	0.053793628	235.15	505.48
256	0.028587256	0.054492888	223.3	240.62
512	0.028464588	0.054424539	133.49	131.23
768	0.029616229	0.053879427	174.28	140.77
1024	0.027621162	0.052318716	275.29	107

Table 3.30. RMSE values and training times for different batch sizes.

MAPE Values (%)				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.001090723	0.002402486	2165.93	992.68
32	0.001089421	0.002424909	1274.95	590.08
64	0.00111678	0.002456237	700.98	492.07
128	0.001094367	0.002423599	235.15	505.48
256	0.001108776	0.002429103	223.3	240.62
512	0.001088735	0.0024307	133.49	131.23
768	0.001108498	0.002402849	174.28	140.77
1024	0.001100419	0.002406615	275.29	107

Table 3.31. MAPE values and training times for different batch sizes.

After analyzing Table 3.30 and Table 3.31, the chosen batch size with shortest training time and relatively low RMSE and MAPE values was 512.

Following the batch size tests, hidden sizes were computed:

RMSE Values				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.055136511	0.05615964	1059.86	884.74
32	0.056862751	0.062688251	622.01	539.6
64	0.06328322	0.058011891	496.53	672.11
128	0.054288386	0.05715185	778.73	1246.5
256	0.028890862	0.054220973	210.89	213.43
512	0.060480495	0.061237713	1692.66	2213.88
768	0.061314139	0.061546169	2784.74	3893.4
1024	0.077997362	0.068194395	3529.6	5307.38

Table 3.32. RMSE values and training times for different hidden sizes.

MAPE Values (%)				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.002424788	0.002631843	1059.86	884.74
32	0.002517572	0.003060279	622.01	539.6
64	0.00306897	0.002673848	496.53	672.11
128	0.00255562	0.00276183	778.73	1246.5
256	0.001062787	0.002378155	210.89	213.43
512	0.002847573	0.002907763	1692.66	2213.88
768	0.002784252	0.002828986	2784.74	3893.4
1024	0.003720658	0.003305398	3529.6	5307.38

Table 3.33. MAPE values and training times for different hidden sizes.

After performing the tests for hidden sizes 16 and 32, it was clear that the LR was too low as it performed 1000 epochs and was only at an RMSE value 1529. The tests were re-done at a LR of 1×10^{-4} .

The hidden size was chosen to be 256.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

RMSE Values				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.028890862	0.054220973	210.89	213.43
2	0.058637073	0.063789094	823.16	2396.72
3	0.05541354	0.063599684	1051.16	2986.95
4	0.056024437	0.155395487	1045.94	2681.71
5	0.057766606	0.162113785	807.57	1320.31
6	0.058898432	0.159515124	886.25	1712.21
7	0.055393435	0.162478768	888.65	2029.9

Table 3.34. RMSE values and training times for different numbers of layers.

MAPE Values (%)				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.001062787	0.002378155	210.89	213.43
2	0.002635514	0.003051264	823.16	2396.72
3	0.002510565	0.003005428	1051.16	2986.95
4	0.002501396	0.006939376	1045.94	2681.71
5	0.002643205	0.006319978	807.57	1320.31
6	0.00264513	0.006216331	886.25	1712.21
7	0.002479416	0.006335794	888.65	2029.9

Table 3.35. MAPE values and training times for different numbers of layers.

After testing all these layers, 3 layers produced the best RMSE and MAPE values.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Test RMSE	Test MAPE (%)	Time (s)
RNN	0.044054186	0.00188626	2726.27
GRU	0.032325035	0.001220794	4014.42
LSTM	0.045241669	0.001955267	4337.61
BiGRU	0.16636307	0.006092314	3374.27
BiLSTM	0.166324221	0.006194498	4217.84
minGRUseq	0.041539674	0.001756217	2380.03
minGRUpar	0.043676657	0.001869935	444.27
minLSTMseq	0.039295471	0.001658661	2790.79
minLSTMpar	0.044499444	0.001905339	2260.44
xLSTM	0.019910915	0.000821625	4996.79

Table 3.36. Comparison of models in terms of test RMSE, test MAPE, and runtime.

3.3.5 Recurrent Layers with Additional FC Layers (128, 64)

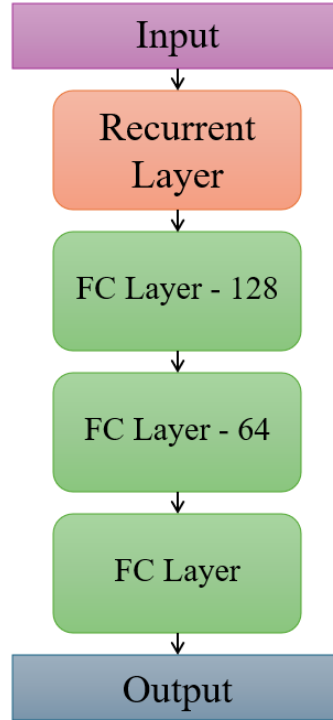


Figure 3.12. Recurrent network with additional FC layers flowchart.

The following tables are with the addition of FC layers with 128 neurons for the first layer and then 64 neurons for the second before finally becoming a single output.

RMSE Values				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.16379873	0.087193165	65.7	237.41
1×10^{-2}	0.159836652	0.061440567	93.13	304.97
1×10^{-3}	0.160635683	0.056112812	159.66	250.35
1×10^{-4}	0.158775265	0.05193862	358.79	354.16
1×10^{-5}	0.160248141	0.051710479	462.41	305.66
1×10^{-6}	0.161366487	0.044055308	58.2	2131.4
1×10^{-7}	0.161364354	0.046816452	56.24	115.81
1×10^{-8}	0.157381212	0.045126597	56.87	82.82

Table 3.37. RMSE values and training times for different LRs.

MAPE Values (%)				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.00626545	0.004306451	65.7	237.41
1×10^{-2}	0.006129041	0.002827758	93.13	304.97
1×10^{-3}	0.006148426	0.002524385	159.66	250.35
1×10^{-4}	0.006063236	0.002280834	358.79	354.16
1×10^{-5}	0.006151549	0.0022359	462.41	305.66
1×10^{-6}	0.006162931	0.001896683	58.2	2131.4
1×10^{-7}	0.006187103	0.001933953	56.24	115.81
1×10^{-8}	0.006048701	0.00189973	56.87	82.82

Table 3.38. MAPE values and training times for different LRs.

From Table 3.37 and Table 3.38, a LR of 1×10^{-4} . Once the LR of 1×10^{-6} , early stop was not stopping the system even though the improvements were not visible. At this point, delta was changed to 1×10^{-5} which allowed the early stop to work.

Next, the batch size was iterated through, starting from 16.

RMSE Values				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.162444627	0.028638505	411.29	693.78
32	0.163038596	0.02324201	289.78	314.79
64	0.161937034	0.021015976	135.91	210.44
128	0.162517337	0.02050551	100.04	167.02
256	0.162228777	0.020455412	79.1	217.45
512	0.16096387	0.020820959	60.14	136.78
768	0.16548851	0.01855163	59.81	147.83
1024	0.158775265	0.05193862	358.79	354.16

Table 3.39. RMSE values and training times for different batch sizes.

MAPE Values (%)				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.006233584	0.001180672	411.29	693.78
32	0.006221828	0.00090901	289.78	314.79
64	0.0062024	0.000776861	135.91	210.44
128	0.00621809	0.000783723	100.04	167.02
256	0.006218971	0.000765753	79.1	217.45
512	0.006167131	0.000743664	60.14	136.78
768	0.006337626	0.000735047	59.81	147.83
1024	0.006063236	0.002280834	358.79	354.16

Table 3.40. MAPE values and training times for different batch sizes.

After analyzing Table 3.39 and Table 3.40, the chosen batch size with shortest training time and relatively low RMSE and MAPE values was 768.

Following the batch size tests, hidden sizes were computed:

RMSE Values				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.12246541	0.301784616	498.41	455.14
32	0.10704473	0.118332959	347.23	399.51
64	0.108285907	0.10980893	349.87	408.28
128	0.16548851	0.01855163	59.81	147.83
256	0.093856689	0.113354884	687.32	731.91
512	0.084882329	0.128811771	1549.94	1804.21
768	0.089755901	0.124659696	2670.35	3517.67
1024	0.092934062	0.145074353	4366.11	5403.93

Table 3.41. RMSE values and training times for different hidden sizes.

MAPE Values (%)				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.005550834	0.009438377	498.41	455.14
32	0.005055559	0.005487247	347.23	399.51
64	0.005394344	0.005178773	349.87	408.28
128	0.006337626	0.000735047	59.81	147.83
256	0.004606265	0.0053362	687.32	731.91
512	0.004024376	0.006470177	1549.94	1804.21
768	0.004046057	0.005501539	2670.35	3517.67
1024	0.004304639	0.007791934	4366.11	5403.93

Table 3.42. MAPE values and training times for different hidden sizes.

The hidden size was chosen to be 128.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

RMSE Values				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.16548851	0.01855163	59.81	147.83
2	0.089340448	0.084558278	489.8	1260.2
3	0.081888984	0.069550318	521.29	1746.03
4	0.096598135	0.111317852	479.63	2211.47
5	0.103099925	0.12716057	473.02	2614.27
6	0.108092236	0.14997174	483.07	2904.59
7	0.091722111	0.15066039	493.86	3404.99

Table 3.43. RMSE values and training times for different numbers of layers.

MAPE Values (%)				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.006337626	0.000735047	59.81	147.83
2	0.004377753	0.00434788	489.8	1260.2
3	0.003808839	0.003600229	521.29	1746.03
4	0.004490946	0.005971211	479.63	2211.47
5	0.004503294	0.006765534	473.02	2614.27
6	0.004802121	0.007342564	483.07	2904.59
7	0.00407081	0.006669093	493.86	3404.99

Table 3.44. MAPE values and training times for different numbers of layers.

After testing all these layers, 3 layers produced the best RMSE and MAPE values.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Test RMSE	Test MAPE (%)	Time (s)
RNN	0.059564413	0.003018663	714.74
GRU	0.050662165	0.002295517	2151.64
LSTM	0.049485922	0.002136271	2372.25
BiGRU	0.164132882	0.006145375	1594.71
BiLSTM	0.096210824	0.005550821	3594.83
minGRUseq	0.067868405	0.003594746	1411.1
minGRUpar	0.047571115	0.002199737	411.35
minLSTMseq	0.051319581	0.002459247	1530.21
minLSTMpar	0.045176448	0.001940005	1169.93
xLSTM	0.033348602	0.001361074	2884.19

Table 3.45. Comparison of models in terms of test RMSE, test MAPE, and runtime.

3.3.6 Recurrent Layers with Additional FC Layers (256, 128, 64)

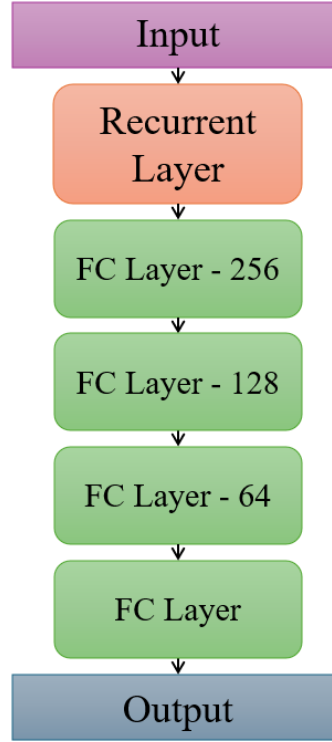


Figure 3.13. Recurrent Layer with additional fully connected layers

The following tables are with the addition of FC layers with 256 neurons for the first layer, and then 128 neurons for the second layer, 64 neurons for the third layer, before finally becoming a single output.

RMSE Values				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.163954522	0.162434247	95.56	203.22
1×10^{-2}	0.163500488	0.161997813	108.81	130.7
1×10^{-3}	0.156857914	0.155726727	353.67	97.96
1×10^{-4}	0.163218436	0.161619013	84.63	97.59
1×10^{-5}	0.164029009	0.162394816	84.4	97.21
1×10^{-6}	0.163494424	0.161874723	83.93	97.59
1×10^{-7}	0.160748972	0.159350079	84.37	97.74
1×10^{-8}	0.162896303	0.161372792	84.24	97.19

Table 3.46. RMSE values and training times for different LRs.

MAPE Values (%)				
Learning Rate	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1×10^{-1}	0.00540088	0.006218233	95.56	203.22
1×10^{-2}	0.00539784	0.006215272	108.81	130.7
1×10^{-3}	0.005134926	0.005960945	353.67	97.96
1×10^{-4}	0.005366045	0.006178204	84.63	97.59
1×10^{-5}	0.005384302	0.00619403	84.4	97.21
1×10^{-6}	0.005418553	0.006223663	83.93	97.59
1×10^{-7}	0.005314914	0.006127107	84.37	97.74
1×10^{-8}	0.005356397	0.006170193	84.24	97.19

Table 3.47. MAPE values and training times for different LRs.

From Table 3.46 and Table 3.47, a LR of 1×10^{-3} was chosen.

Next, the batch size was iterated through, starting from 16.

RMSE Values				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.163730447	0.162199275	631.47	658.48
32	0.159778555	0.158552441	320.18	405.37
64	0.165543376	0.163736969	198.56	213.38
128	0.165074855	0.163392261	145.58	174.15
256	0.164339964	0.162696173	210.52	135.55
512	0.15905899	0.157768317	253.18	160.04
768	0.164761922	0.163129792	302.17	103.49
1024	0.156857914	0.155726727	353.67	97.96

Table 3.48. RMSE values and training times for different batch sizes.

MAPE Values (%)				
Batch Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.005387938	0.006199373	631.47	658.48
32	0.00522834	0.00606802	320.18	405.37
64	0.005481075	0.006271737	198.56	213.38
128	0.005420221	0.006233501	145.58	174.15
256	0.005425641	0.006232603	210.52	135.55
512	0.00520574	0.006027011	253.18	160.04
768	0.005438252	0.006250576	302.17	103.49
1024	0.005134926	0.005960945	353.67	97.96

Table 3.49. MAPE values and training times for different batch sizes.

After analyzing Table 3.48 and Table 3.49, the chosen batch size with lowest RMSE and MAPE values was 1024.

Following the batch size tests, hidden sizes were computed:

RMSE Values				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.844095024	0.836120277	190.8	206.69
32	0.261098127	0.675338609	183.81	166.2
64	0.203302963	0.246019626	205.81	279.56
128	0.122297456	0.123244829	493.76	604.31
256	0.156857914	0.155726727	353.67	97.96
512	0.183629751	0.144526562	1159.94	1565.09
768	0.162742166	0.135094429	2149.15	2377.06
1024	0.077067382	1.132321335	2209.5	765.57

Table 3.50. RMSE values and training times for different hidden sizes.

MAPE Values (%)				
Hidden Size	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
16	0.041538892	0.041194572	190.8	206.69
32	0.012021869	0.033637285	183.81	166.2
64	0.009533154	0.010957777	205.81	279.56
128	0.005820387	0.006513358	493.76	604.31
256	0.005134926	0.005960945	353.67	97.96
512	0.009270914	0.006160185	1159.94	1565.09
768	0.007714842	0.007015909	2149.15	2377.06
1024	0.003421256	0.057695632	2209.5	2453.6

Table 3.51. MAPE values and training times for different hidden sizes.

The hidden size was chosen to be 128.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

RMSE Values				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.122297456	0.123244829	493.76	604.31
2	0.146961371	0.178240986	293.06	688.63
3	0.142418676	0.183880866	295.77	946.39
4	0.150560002	0.146747738	282.35	1351.18
5	0.1340052	0.077369661	320.64	1599.64
6	0.130640406	0.10913062	301.95	1861.44
7	0.136473836	0.091260104	339.37	2151.5

Table 3.52. RMSE values and training times for different numbers of layers.

MAPE Values (%)				
Number of Layers	minGRU	minLSTM	Time (s)(minGRU)	Time (s)(minLSTM)
1	0.005820387	0.006513358	493.76	604.31
2	0.007106652	0.008268836	293.06	688.63
3	0.006434805	0.009882909	295.77	946.39
4	0.006906361	0.007929114	282.35	1351.18
5	0.007145964	0.00397835	320.64	1599.64
6	0.006507447	0.005870211	301.95	1861.44
7	0.006295224	0.004854068	339.37	2151.5

Table 3.53. MAPE values and training times for different numbers of layers.

After testing all these layers, 1 layer produced the best RMSE and MAPE values.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Test RMSE	Test MAPE (%)	Time (s)
RNN	0.0607834	0.003003553	343.11
GRU	0.124305623	0.006178694	474.66
LSTM	0.087069353	0.004102954	727.37
BiGRU	0.09724208	0.005014743	807.18
BiLSTM	0.086394026	0.003672421	1259.5
minGRUseq	0.165899384	0.009133869	835.08
minGRUpair	0.079231819	0.003558075	419.59
minLSTMseq	0.099862399	0.004563866	948.66
minLSTMpair	0.067308431	0.003606763	564.81
xLSTM	0.082364089	0.003551821	840.15

Table 3.54. Comparison of models in terms of test RMSE, test MAPE, and runtime.

3.4 Sequence Length of 500 and Output of 50

After performing the above tests, the sequence length was modified from 50 to 500, and the output was changed from 1 to 50. This way, a longer prediction period could be computed. Since the data was obtained with a frequency of 50 Hertz, this equates to 10 seconds of input to obtain 1 second of output.

Also, with the addition of the lengthened sequence length, more RMSE values could be obtained instead of just the average that was mentioned in the above tests. The chosen RMSE values were at 0, 25, and 50, which means the start, middle, and end of the output, to verify how accurate the RMSE and MAPE values were.

3.4.1 (Convolutional Layer + ReLU + MaxPool) x 3 + Recurrent Layers

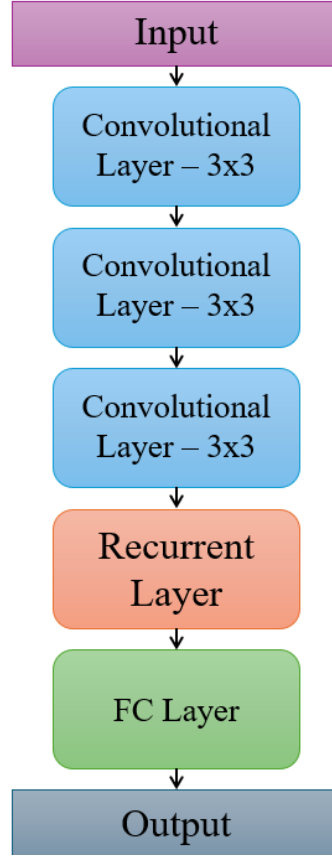


Figure 3.14. Recurrent and convolutional flowchart.

Starting with a LR of 1×10^{-1} and doing 8 tests of decreasing order:

Learning Rate	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1×10^{-1}	0.161824	0.161808	0.161756	0.161907	231.97	minGRU
	0.161786	0.161736	0.161860	0.161761	241.41	minLSTM
1×10^{-2}	0.160451	0.160154	0.160458	0.160739	179.35	minGRU
	0.160399	0.160065	0.160546	0.160587	209.89	minLSTM
1×10^{-3}	0.163208	0.163316	0.163256	0.163050	241.80	minGRU
	0.163207	0.163312	0.163255	0.163054	276.99	minLSTM
1×10^{-4}	0.161081	0.161235	0.160934	0.161073	212.20	minGRU
	0.161081	0.161231	0.160934	0.161076	261.35	minLSTM
1×10^{-5}	0.159956	0.160100	0.160055	0.159715	345.25	minGRU
	0.159956	0.160100	0.160054	0.159713	722.33	minLSTM
1×10^{-6}	0.162232	0.162123	0.162339	0.162233	565.00	minGRU
	0.162232	0.162121	0.162339	0.162235	1129.42	minLSTM
1×10^{-7}	0.161595	0.161354	0.161587	0.161844	1281.02	minGRU
	0.161596	0.161353	0.161588	0.161847	1780.73	minLSTM
1×10^{-8}	0.162040	0.162136	0.162090	0.161892	1733.59	minGRU
	0.162040	0.162131	0.162091	0.161897	2325.65	minLSTM

Table 3.55. LR results at different RMSE values.

Learning Rate	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1×10^{-1}	0.006572	0.006503	0.006578	0.006634	231.97	minGRU
	0.006072	0.006229	0.005874	0.006112	241.41	minLSTM
1×10^{-2}	0.006521	0.006445	0.006530	0.006588	179.35	minGRU
	0.006026	0.006173	0.005835	0.006071	209.89	minLSTM
1×10^{-3}	0.006304	0.006294	0.006309	0.006309	241.80	minGRU
	0.006304	0.006316	0.006309	0.006287	276.99	minLSTM
1×10^{-4}	0.006214	0.006206	0.006215	0.006221	212.20	minGRU
	0.006214	0.006228	0.006215	0.006200	261.35	minLSTM
1×10^{-5}	0.006177	0.006168	0.006184	0.006178	345.25	minGRU
	0.006177	0.006190	0.006185	0.006156	722.33	minLSTM
1×10^{-6}	0.006265	0.006250	0.006272	0.006273	565.00	minGRU
	0.006265	0.006272	0.006272	0.006251	1129.42	minLSTM
1×10^{-7}	0.006216	0.006193	0.006222	0.006234	1281.02	minGRU
	0.006216	0.006215	0.006223	0.006212	1780.73	minLSTM
1×10^{-8}	0.006243	0.006236	0.006249	0.006243	1733.59	minGRU
	0.006243	0.006258	0.006250	0.006221	2325.65	minLSTM

Table 3.56. LR results at different MAPE values.

From Table 3.55 and Table 3.56, the LR of 1×10^{-5} was chosen as it produced the best compromise of time and RMSE value.

Next, the batch size was iterated through, starting from 16.

Batch Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	0.165548135	0.165545201	0.165563849	0.165535356	824.85	minGRU
	0.165546575	0.165537769	0.16556129	0.165540667	778.82	minLSTM
32	0.157927029	0.158382573	0.157973747	0.157424766	415.45	minGRU
	0.157926353	0.158384613	0.157975062	0.157419383	432.36	minLSTM
64	0.162512035	0.162664314	0.162521751	0.162350039	324.67	minGRU
	0.162509628	0.162660111	0.162522021	0.162346752	426	minLSTM
128	0.16379668	0.16367023	0.163930102	0.16378971	234.63	minGRU
	0.163785534	0.163658251	0.163916847	0.163781504	298.9	minLSTM
256	0.163127859	0.163128505	0.163118177	0.163136897	250.6	minGRU
	0.163120297	0.16311921	0.163113563	0.163128116	233.98	minLSTM
512	0.163987263	0.164006042	0.163973517	0.16398223	245.67	minGRU
	0.163975335	0.163983848	0.163964736	0.163977419	282.66	minLSTM
768	0.163227033	0.163193256	0.163081649	0.163406195	325.45	minGRU
	0.163215552	0.16317202	0.163072905	0.163401732	283.95	minLSTM
1024	0.163046935	0.162756782	0.163192366	0.163191658	237.26	minGRU
	0.163038852	0.162743964	0.16318481	0.163187783	262.81	minLSTM

Table 3.57. Batch size results at different RMSE values.

Batch Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.006358491	0.006353572	0.006360353	0.006361548	824.85	minGRU
	0.00636214	0.006370008	0.006365838	0.006350575	778.82	minLSTM
32	0.006060534	0.006077105	0.006068407	0.00603609	415.45	minGRU
	0.006056818	0.006088363	0.006074033	0.00600806	432.36	minLSTM
64	0.006273084	0.006275923	0.006286867	0.006256463	324.67	minGRU
	0.006325021	0.006331482	0.006348194	0.006295386	426	minLSTM
128	0.006287143	0.006279158	0.006361544	0.00630941	234.63	minGRU
	0.006333378	0.00632918	0.006294148	0.006258513	298.9	minLSTM
256	0.00627462	0.006271197	0.006294148	0.006258513	250.6	minGRU
	0.006335491	0.006343417	0.00634953	0.006313527	233.98	minLSTM
512	0.006342221	0.006319899	0.006359172	0.006347593	245.67	minGRU
	0.006388268	0.006396552	0.006404059	0.006364193	282.66	minLSTM
768	0.00627658	0.006249389	0.006287235	0.006293115	325.45	minGRU
	0.006328284	0.006332404	0.006342703	0.006309745	283.95	minLSTM
1024	0.006296109	0.00626599	0.006317323	0.006305015	237.26	minGRU
	0.006342248	0.006343157	0.006362014	0.006321572	262.81	minLSTM

Table 3.58. Batch size results at different MAPE values.

After analyzing Table 3.57 and Table 3.58, the first batch size was chosen to be 32 since it produced such good results, but using such a low batch size proved to take too long, so it was changed to 1024, which still produced good results but with much more reasonable training times.

Following the batch size tests, hidden sizes were computed:

Hidden Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	0.161533253	0.161413861	0.161457665	0.161728233	1279.88	minGRU
	0.161535238	0.161415707	0.161460503	0.161729505	1516.55	minLSTM
32	0.161141984	0.161332267	0.161133789	0.160959897	826.08	minGRU
	0.16113043	0.161317194	0.161124952	0.160949146	839.64	minLSTM
64	0.159691948	0.159819313	0.159652091	0.159604442	603.34	minGRU
	0.159707558	0.15984403	0.159665956	0.159612687	592.96	minLSTM
128	0.044155084	0.04135074	0.043931691	0.047182821	1264.51	minGRU
	0.042960377	0.039982034	0.042491551	0.046407547	1192.95	minLSTM
256	0.157927029	0.158382573	0.157973747	0.157424766	415.45	minGRU
	0.157926353	0.158384613	0.157975062	0.157419383	432.36	minLSTM
512	0.035375264	0.032195767	0.03544202	0.038488005	2978.35	minGRU
	0.046960261	0.04459635	0.04703765	0.049246784	2595.51	minLSTM
768	0.034622874	0.032029018	0.034699035	0.037140568	3487.51	minGRU
	0.039547871	0.037155079	0.039901163	0.04158737	4232.76	minLSTM
1024	0.037208874	0.034409818	0.037192978	0.040023827	3327.54	minGRU
	0.042891224	0.040120671	0.042399554	0.046153446	4479.11	minLSTM

Table 3.59. Hidden Size results at different RMSE values.

Hidden Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.006307118	0.006298113	0.006316423	0.006306817	1279.88	minGRU
	0.006243862	0.00624223	0.006221583	0.006267772	1516.55	minLSTM
32	0.00638767	0.00641064	0.006379635	0.006372736	826.08	minGRU
	0.00633713	0.006348811	0.006340377	0.006322204	839.64	minLSTM
64	0.006159259	0.006162868	0.006145508	0.0061694	603.34	minGRU
	0.006054595	0.006036652	0.006057228	0.006069904	592.96	minLSTM
128	0.001304823	0.001281838	0.001306608	0.001326022	1264.51	minGRU
	0.001245711	0.001192271	0.001191356	0.001353504	1192.95	minLSTM
256	0.006060534	0.006077105	0.006068407	0.00603609	415.45	minGRU
	0.006056818	0.006088363	0.006074033	0.00600806	432.36	minLSTM
512	0.001122022	0.001106978	0.001120867	0.001138219	2978.35	minGRU
	0.001386026	0.001353769	0.001384076	0.001420233	2595.51	minLSTM
768	0.001169474	0.001176304	0.00114877	0.001183347	3487.51	minGRU
	0.001188023	0.001180938	0.001202173	0.00118096	4232.76	minLSTM
1024	0.037208874	0.001336865	0.001335091	0.001362954	3327.54	minGRU
	0.042891224	0.001215219	0.001236982	0.001457343	4479.11	minLSTM

Table 3.60. Hidden Size results at different MAPE values.

From Table 3.59 and Table 3.60, the hidden size value of 128 was chosen.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

Number of Layers	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1	0.044155084	0.04135074	0.043931691	0.047182821	1264.51	minGRU
	0.042960377	0.039982034	0.042491551	0.046407547	1192.95	minLSTM
2	0.051877541	0.050240183	0.05152589	0.053866551	829.27	minGRU
	0.032749768	0.030058887	0.032794299	0.035396117	1771.79	minLSTM
3	0.047613528	0.045166142	0.047732646	0.049941795	685.01	minGRU
	0.071296251	0.069745146	0.071322956	0.072820652	1606.72	minLSTM
4	0.04756146	0.04539531	0.047176418	0.050112651	1077.94	minGRU
	0.066094397	0.064763776	0.065906895	0.067612519	2041.86	minLSTM
5	0.047565017	0.045845971	0.048020713	0.048828366	733.22	minGRU
	0.069825676	0.068709953	0.070024889	0.070742185	1580.33	minLSTM
6	0.047020128	0.044608863	0.04703213	0.049419392	1035.29	minGRU
	0.068922137	0.067459105	0.068784651	0.070522657	1545.44	minLSTM
7	0.051681756	0.049726694	0.051348072	0.053970501	804.28	minGRU
	0.067046637	0.066141745	0.066633631	0.068364535	2334.72	minLSTM

Table 3.61. RMSE values obtained in relation to the number of layers.

Number of Layers	Avg MAPE	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1	0.001304823	0.001281838	0.001306608	0.001326022	1264.51	minGRU
	0.001245711	0.001192271	0.001191356	0.001353504	1192.95	minLSTM
2	0.001770245	0.00176476	0.00175911	0.001786867	829.27	minGRU
	0.000862715	0.000836453	0.000862092	0.0008896	1771.79	minLSTM
3	0.001390872	0.001396846	0.001391038	0.001384732	685.01	minGRU
	0.001960434	0.001968561	0.00194346	0.001969281	1606.72	minLSTM
4	0.001412144	0.001375167	0.001404674	0.001456592	1077.94	minGRU
	0.001750581	0.00172602	0.001748166	0.001777559	2041.86	minLSTM
5	0.001427749	0.001408158	0.001442657	0.001432431	733.22	minGRU
	0.001903121	0.001890619	0.001897862	0.001920881	1580.33	minLSTM
6	0.001423773	0.001376363	0.00144507	0.001449888	1035.29	minGRU
	0.001875118	0.001869926	0.001865351	0.001890076	1545.44	minLSTM
7	0.001617137	0.001619246	0.001610248	0.001621918	804.28	minGRU
	0.00178332	0.001858204	0.00175012	0.001741637	2334.72	minLSTM

Table 3.62. MAPE values obtained in relation to the number of layers.

After testing all these layer numbers, 1 produced the best RMSE and MAPE values.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)
RNN	0.168471	0.168451	0.168449	0.168512	1319.84
GRU	0.168466	0.168448	0.168446	0.168503	1479.52
LSTM	0.168395	0.168388	0.168361	0.168436	1679.09
BiGRU	0.037559	0.037078	0.037596	0.038003	3035.16
BiLSTM	0.028386	0.028073	0.027994	0.029091	5990.02
minGRUseq	0.168442	0.168418	0.168424	0.168483	2977.45
minGRUpar	0.026458	0.025251	0.026582	0.027540	1510.60
minLSTMseq	0.023219	0.021757	0.023474	0.024425	7097.71
minLSTMpar	0.029663	0.028306	0.029724	0.030959	1323.39
xLSTM	0.168401	0.168378	0.168370	0.168456	3447.97

Table 3.63. Comparison of models based on RMSE.

Model	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)
RNN	0.006163	0.006160	0.006163	0.006165	1319.84
GRU	0.006142	0.006148	0.006150	0.006127	1479.52
LSTM	0.006120	0.006121	0.006118	0.006122	1679.09
BiGRU	0.001287	0.001324	0.001277	0.001260	3035.16
BiLSTM	0.000982	0.001086	0.000899	0.000961	5990.02
minGRUseq	0.006163	0.006154	0.006164	0.006172	2977.45
minGRUpar	0.000856	0.000839	0.000856	0.000872	1510.60
minLSTMseq	0.000672	0.000638	0.000681	0.000697	7097.71
minLSTMpar	0.000854	0.000800	0.000848	0.000914	1323.39
xLSTM	0.006092	0.006092	0.006088	0.006095	3447.97

Table 3.64. Comparison of models based on MAPE.

3.4.2 (Convolutional Layer + ReLU + MaxPool) x 2 + Recurrent Layers

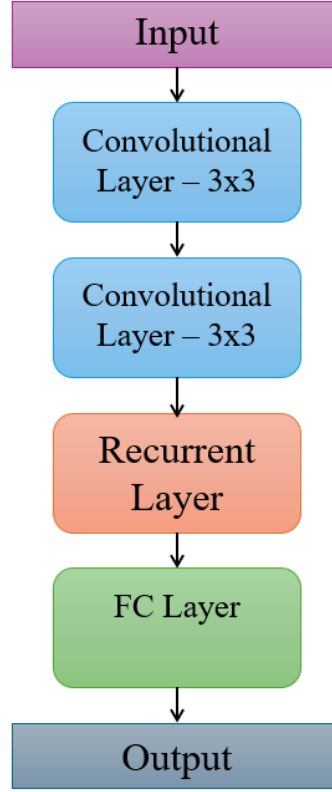


Figure 3.15. Recurrent and convolutional flowchart.

Starting with a LR of 1×10^{-1} and doing 8 tests of decreasing order:

Learning Rate	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1×10^{-1}	0.163039311	0.163547929	0.163017418	0.162552584	335.73	minGRU
	0.1629056	0.163243543	0.162919136	0.162554119	385.88	minLSTM
1×10^{-2}	0.161088028	0.161269679	0.161068846	0.160925558	400.5	minGRU
	0.161044077	0.161272774	0.161021177	0.16083828	507.87	minLSTM
1×10^{-3}	0.158719814	0.158594636	0.158846876	0.158717931	282.58	minGRU
	0.158647116	0.15860894	0.158756566	0.158575843	552.93	minLSTM
1×10^{-4}	0.158188918	0.158207829	0.158192558	0.158166369	406.73	minGRU
	0.158185119	0.158203207	0.158188083	0.158164067	419.76	minLSTM
1×10^{-5}	0.15810147	0.158214168	0.158007442	0.158082799	443.1	minGRU
	0.158097427	0.158209518	0.15800252	0.158080242	342.23	minLSTM
1×10^{-6}	0.158216061	0.158434857	0.158256004	0.157957322	819.93	minGRU
	0.158212101	0.158429896	0.158251696	0.15795471	1234.26	minLSTM
1×10^{-7}	0.163613526	0.163558749	0.163562501	0.163719329	1700.08	minGRU
	0.163617883	0.163562091	0.163569026	0.163722532	2187.42	minLSTM
1×10^{-8}	0.164546308	0.164422645	0.164483858	0.164732422	2375.8	minGRU
	0.164552348	0.164429	0.16449132	0.164736725	3313.93	minLSTM

Table 3.65. LR results at different RMSE values.

Learning Rate	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1×10^{-1}	0.006066347	0.005829276	0.006049552	0.006320213	335.73	minGRU
	0.006173041	0.006036785	0.00617893	0.006303407	385.88	minLSTM
1×10^{-2}	0.00636584	0.006240274	0.006401645	0.006455602	400.5	minGRU
	0.006218219	0.00628353	0.006192719	0.006178407	507.87	minLSTM
1×10^{-3}	0.006279363	0.006140767	0.006322564	0.006374757	282.58	minGRU
	0.006130983	0.006184184	0.006112833	0.0060959323	552.93	minLSTM
1×10^{-4}	0.006152053	0.006161463	0.006153028	0.006141668	406.73	minGRU
	0.0061348	0.006143308	0.006130759	0.006130334	419.76	minLSTM
1×10^{-5}	0.00611889	0.006134087	0.006111132	0.006111453	443.1	minGRU
	0.006101557	0.006115861	0.006088746	0.006100064	342.23	minLSTM
1×10^{-6}	0.006128801	0.006147329	0.006126697	0.006112376	819.93	minGRU
	0.006111563	0.00612915	0.006104469	0.00610107	1234.26	minLSTM
1×10^{-7}	0.006314656	0.006323631	0.006308517	0.006311822	1700.08	minGRU
	0.006297677	0.006305743	0.006286627	0.006300661	2187.42	minLSTM
1×10^{-8}	0.006346874	0.00635318	0.006342233	0.006345208	2375.8	minGRU
	0.006329868	0.006335278	0.006320277	0.006334051	3313.93	minLSTM

Table 3.66. LR results at different MAPE values.

From Table 3.65 and Table 3.66, the LR of 1×10^{-3} was chosen as it produced the best compromise of time and RMSE value.

Next, the batch size was iterated through, starting from 16.

Batch Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	0.115433022	0.130932837	0.16953395	0.045832279	1262.45	minGRU
	0.159223035	0.159361022	0.159230663	0.159077418	1302.08	minLSTM
32	0.041376079	0.036745299	0.047631915	0.039751023	970.73	minGRU
	0.162441153	0.162283155	0.162372766	0.162667537	627.94	minLSTM
64	0.029534126	0.026682688	0.031304716	0.030614974	402.31	minGRU
	0.158778684	0.158767154	0.15874282	0.158826078	483.78	minLSTM
128	0.028476047	0.025485304	0.030436394	0.029506444	601.11	minGRU
	0.160193555	0.160294554	0.160209746	0.160076366	404.34	minLSTM
256	0.028656531	0.026295316	0.030219813	0.029454463	312.13	minGRU
	0.162436666	0.162399624	0.162514221	0.162396154	438.6	minLSTM
512	0.027542357	0.022432039	0.028412865	0.031782168	318.2	minGRU
	0.16490732	0.164793277	0.164885156	0.165043526	368.06	minLSTM
768	0.026147965	0.022419875	0.02577335	0.03025067	395.79	minGRU
	0.160933602	0.160960516	0.160948683	0.160891607	504.89	minLSTM
1024	0.026934166	0.022677539	0.02711843	0.031006528	386.5	minGRU
	0.159804552	0.159769679	0.159862831	0.159781145	412.96	minLSTM

Table 3.67. Batch size results at different RMSE values.

Batch Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.115433022	0.130932837	0.16953395	0.045832279	1262.45	minGRU
	0.159223035	0.159361022	0.159230663	0.159077418	1302.08	minLSTM
32	0.041376079	0.036745299	0.047631915	0.039751023	970.73	minGRU
	0.162441153	0.162283155	0.162372766	0.162667537	627.94	minLSTM
64	0.029534126	0.026682688	0.031304716	0.030614974	402.31	minGRU
	0.158778684	0.158767154	0.15874282	0.158826078	483.78	minLSTM
128	0.028476047	0.025485304	0.030436394	0.029506444	601.11	minGRU
	0.160193555	0.160294554	0.160209746	0.160076366	404.34	minLSTM
256	0.028656531	0.026295316	0.030219813	0.029454463	312.13	minGRU
	0.162436666	0.162399624	0.162514221	0.162396154	438.6	minLSTM
512	0.027542357	0.022432039	0.028412865	0.031782168	318.2	minGRU
	0.16490732	0.164793277	0.164885156	0.165043526	368.06	minLSTM
768	0.026147965	0.022419875	0.02577335	0.03025067	395.79	minGRU
	0.160933602	0.160960516	0.160948683	0.160891607	504.89	minLSTM
1024	0.026934166	0.022677539	0.02711843	0.031006528	386.5	minGRU
	0.159804552	0.159769679	0.159862831	0.159781145	412.96	minLSTM

Table 3.68. Batch size results at different MAPE values.

After analyzing Table 3.67 and Table 3.68, the first batch size was chosen to be 1024 since it produced good results with low times, but it was later decided to choose 256 since the values were comparable and it reduced the load of the test.

Following the batch size tests, hidden sizes were computed:

Hidden Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
32	0.163488122	0.163626995	0.163461265	0.163376105	284.21	minGRU
	0.163459111	0.163596001	0.163434051	0.16334728	277.83	minLSTM
64	0.163292964	0.163327826	0.163308534	0.163242533	258.81	minGRU
	0.163088814	0.163099431	0.163125056	0.163041954	333.56	minLSTM
128	0.161157177	0.161127755	0.161203496	0.16114028	422.75	minGRU
	0.06126768	0.059542908	0.061102001	0.063158132	1575.28	minLSTM
256	0.026934166	0.022677539	0.02711843	0.031006528	386.5	minGRU
	0.159804552	0.159769679	0.159862831	0.159781145	412.96	minLSTM

Table 3.69. Hidden Size results at different RMSE values.

Hidden Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
32	0.006283862	0.006285895	0.006282232	0.006283458	284.21	minGRU
	0.006259832	0.006261126	0.006258815	0.006259555	277.83	minLSTM
64	0.006319372	0.006317046	0.00632194	0.006319132	258.81	minGRU
	0.00631786	0.006312603	0.006324579	0.006316396	333.56	minLSTM
128	0.006341548	0.006337421	0.006344745	0.006342479	422.75	minGRU
	0.002401835	0.002366875	0.00239881	0.002439821	1575.28	minLSTM
256	0.001129662	0.001078348	0.00116133	0.001149309	386.5	minGRU
	0.006211245	0.006183876	0.006193245	0.006256616	412.96	minLSTM

Table 3.70. Hidden Size results at different MAPE values.

From Table 3.69 and Table 3.70, the hidden size value of 256 was chosen. The hidden size values of 16 with the LR of 0.001 did not allow the system to train enough on each epoch, so after more than 100 epochs, the RMSE values were still near 1500, so the next set of hidden sizes was tested. For the values of 512, 768, and 1024, it was decided not to perform them as it was taking 6 minutes per epoch, and after 4 hours of training, it still had not changed the LR with the automatic LR scheduler.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

Number of Layers	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1	0.026934166	0.022677539	0.02711843	0.031006528	386.5	minGRU
	0.159804552	0.159769679	0.159862831	0.159781145	412.96	minLSTM
2	0.045433289	0.044047085	0.044996102	0.047256679	2092.86	minGRU
	0.071948352	0.071065415	0.07163088	0.07314876	2956.22	minLSTM
3	0.065339248	0.063655027	0.06530418	0.067058536	1772.18	minGRU
	0.156633548	0.156824071	0.156567555	0.156509018	2435.23	minLSTM
4	0.073478525	0.071693576	0.073618153	0.075123846	1867.08	minGRU
	0.162488306	0.162593126	0.162652616	0.162219177	4285.97	minLSTM
5	0.028968251	0.024120893	0.029774421	0.033009439	297.91	minGRU
	0.161079772	0.161252665	0.161128293	0.16085836	1761.84	minLSTM
6	0.161931487	0.161887569	0.161958566	0.161948326	1315.42	minGRU
	0.161927328	0.161882591	0.161957549	0.161941844	3275.97	minLSTM
7	0.162386431	0.162312203	0.16233994	0.16250715	1395.36	minGRU
	0.162463028	0.162370143	0.162425821	0.16259312	2982.67	minLSTM

Table 3.71. RMSE values obtained in relation to the number of layers.

Number of Layers	Avg MAPE	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1	0.001129662	0.001078348	0.00116133	0.001149309	386.5	minGRU
	0.006211245	0.006183876	0.006193245	0.006256616	412.96	minLSTM
2	0.001349321	0.001349363	0.001323951	0.001374649	2092.86	minGRU
	0.001996193	0.001977414	0.001986085	0.00202508	2956.22	minLSTM
3	0.002174524	0.002139584	0.002180064	0.002203924	1772.18	minGRU
	0.006295493	0.006301798	0.006313136	0.006271544	2435.23	minLSTM
4	0.002271136	0.002248063	0.002271912	0.002293433	1867.08	minGRU
	0.006289884	0.006286195	0.006291651	0.006291806	4285.97	minLSTM
5	0.000986083	0.000955027	0.000987135	0.001016086	297.91	minGRU
	0.006212091	0.006215298	0.006234193	0.006186783	1761.84	minLSTM
6	0.006373941	0.006378218	0.006372698	0.006370907	1315.42	minGRU
	0.006341858	0.006349674	0.006366999	0.006308901	3275.97	minLSTM
7	0.00621412	0.006204552	0.006224488	0.006213319	1395.36	minGRU
	0.006067514	0.006084492	0.006056165	0.006061884	2982.67	minLSTM

Table 3.72. MAPE values obtained in relation to the number of layers.

After testing all these layer numbers, 1 produced the best RMSE and MAPE values compared with the time spent.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)
RNN	0.1189136	0.12003719	0.117926734	0.118776875	4436.67
GRU	0.058115123	0.055943774	0.060521666	0.05787993	3804.92
LSTM	0.079588964	0.061614335	0.114388566	0.062763991	4851.16
BiGRU	0.044320427	0.044116722	0.045171999	0.043672559	6711.54
BiLSTM	0.036469962	0.034600816	0.03586565	0.038943421	5556.56
minGRUseq	0.062641237	0.091216618	0.049545715	0.047161379	14276.08
minGRUpar	0.046388253	0.045335177	0.045888234	0.04794135	7178.45
minLSTMseq	0.062740637	0.048881586	0.048897491	0.090442833	14345.9
minLSTMpar	0.032917154	0.032027872	0.032688576	0.034035015	7561.61
xLSTM	0.039167888	0.038041133	0.038609682	0.040852849	11801.97

Table 3.73. Comparison of models based on RMSE.

Model	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)
RNN	0.005829309	0.006161047	0.005643584	0.005683295	4436.67
GRU	0.002739498	0.002668743	0.002827436	0.002722315	3804.92
LSTM	0.003821963	0.002462706	0.006507003	0.00249618	4851.16
BiGRU	0.001797931	0.001827061	0.001855992	0.001710742	6711.54
BiLSTM	0.001559537	0.001487297	0.001482691	0.001708623	5556.56
minGRUseq	0.003018346	0.005359518	0.002028316	0.001667206	14276.08
minGRUpar	0.046388253	0.001602951	0.001623747	0.001803697	7178.45
minLSTMseq	0.002982435	0.001886783	0.00185852	0.005202001	14345.9
minLSTMpar	0.000982956	0.00094014	0.000958475	0.001050253	7561.61
xLSTM	0.001333369	0.00128634	0.001275658	0.001438108	11801.97

Table 3.74. Comparison of models based on MAPE.

3.4.3 (Convolutional Layer + ReLU + MaxPool) x 1 + Recurrent Layers

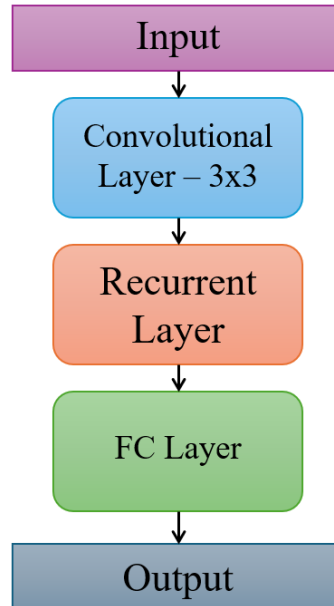


Figure 3.16. Recurrent and convolutional flowchart.

Starting with a LR of 1×10^{-1} and doing 8 tests of decreasing order:

Learning Rate	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1×10^{-1}	0.162579481	0.16275117	0.162858848	0.162128424	1291.04	minGRU
	0.161885148	0.161418759	0.162263869	0.161972815	1245.13	minLSTM
1×10^{-2}	0.159648907	0.159834646	0.159574512	0.159537562	1441.53	minGRU
	0.149296908	0.149156344	0.149270767	0.149463614	1800.91	minLSTM
1×10^{-3}	0.162001422	0.162220399	0.16203694	0.161746927	1736.71	minGRU
	0.149819703	0.14988578	0.149852222	0.149721106	1771.3	minLSTM
1×10^{-4}	0.163795172	0.163853142	0.163881852	0.16365052	1236.13	minGRU
	0.151628264	0.151458283	0.151700838	0.151725671	1286.86	minLSTM
1×10^{-5}	0.1625243	0.162460071	0.16253704	0.162575789	1582.12	minGRU
	0.150279885	0.150005775	0.1503231	0.150510779	1610.16	minLSTM
1×10^{-6}	0.161800052	0.161719286	0.161858736	0.161822133	1397.47	minGRU
	0.150125368	0.149857403	0.150191603	0.150327098	1039.49	minLSTM
1×10^{-7}	0.15928335	0.1591404	0.159177355	0.159532296	1595.75	minGRU
	0.14756494	0.147244566	0.147385984	0.14806427	1559.91	minLSTM
1×10^{-8}	0.16178916	0.16201097	0.161688092	0.161668417	3188.87	minGRU
	0.149861941	0.149925778	0.149749959	0.149910086	1037.25	minLSTM

Table 3.75. LR results at different RMSE values.

Learning Rate	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1×10^{-1}	0.00605591	0.006005435	0.005828648	0.006333646	1291.04	minGRU
	0.006737756	0.006662545	0.006707564	0.006843161	1245.13	minLSTM
1×10^{-2}	0.006309523	0.00626102	0.006252162	0.006415388	1441.53	minGRU
	0.005523069	0.005566938	0.005521893	0.005480378	1800.91	minLSTM
1×10^{-3}	0.006319945	0.006315381	0.006305791	0.006338664	1736.71	minGRU
	0.005916183	0.005892415	0.005882444	0.005973691	1771.3	minLSTM
1×10^{-4}	0.00632061	0.006328692	0.00632013	0.006313008	1236.13	minGRU
	0.006008944	0.00600614	0.006003805	0.006016888	1286.86	minLSTM
1×10^{-5}	0.006283839	0.006287843	0.006278256	0.006285418	1582.12	minGRU
	0.005949982	0.005944713	0.005941807	0.005963426	1610.16	minLSTM
1×10^{-6}	0.006251523	0.006256722	0.006246011	0.006251835	1397.47	minGRU
	0.005979339	0.005975929	0.005970032	0.005992055	1039.49	minLSTM
1×10^{-7}	0.006154408	0.006155314	0.006144108	0.006163802	1595.75	minGRU
	0.005855635	0.005848951	0.005838781	0.005879172	1559.91	minLSTM
1×10^{-8}	0.006249764	0.006265943	0.006240112	0.006243236	3188.87	minGRU
	0.005971954	0.005980659	0.005958632	0.005976573	1037.25	minLSTM

Table 3.76. LR results at different MAPE values.

From Table 3.75 and Table 3.76, the LR of 1×10^{-2} was chosen as it produced the best compromise of time and RMSE value.

Next, the batch size was iterated through, starting from 16.

Batch Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	0.164853322	0.16488132	0.16503316	0.164645485	26939.52	minGRU
	0.152060478	0.15191501	0.152187655	0.152078769	24326.55	minLSTM
32	0.158732172	0.158813612	0.15867555	0.158707355	10663.02	minGRU
	0.147215325	0.147132441	0.147118514	0.147395021	8385.75	minLSTM
64	0.16085535	0.160949382	0.160865968	0.160750699	4156.88	minGRU
	0.149173538	0.149081233	0.149158691	0.149280689	4195.98	minLSTM
128	0.162938324	0.162833993	0.163026208	0.16295477	2737.23	minGRU
	0.091373389	0.083816593	0.088803313	0.101500261	2812.36	minLSTM
256	0.161789093	0.161670721	0.161720447	0.16197611	1605.18	minGRU
	0.086809956	0.078680326	0.083377199	0.098372342	1723.13	minLSTM
512	0.160001419	0.160019964	0.15995896	0.160025333	1113.45	minGRU
	0.081287346	0.07988125	0.080902274	0.083078514	1240.5	minLSTM
768	0.159511739	0.159756864	0.159785132	0.158993221	1187.3	minGRU
	0.081825218	0.080585771	0.081155746	0.083734137	1275.99	minLSTM
1024	0.158939443	0.15884885	0.158940952	0.159028526	1390.63	minGRU
	0.081341487	0.079598539	0.080982189	0.083443731	1426.99	minLSTM

Table 3.77. Batch size results at different RMSE values.

Batch Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.006375475 0.006000462	0.00641802 0.006012536	0.006336842 0.006016706	0.006371564 0.005972145	26939.52 24326.55	minGRU minLSTM
32	0.006188631 0.147215325	0.006238047 0.147132441	0.006135375 0.147118514	0.006192471 0.147395021	10663.02 8385.75	minGRU minLSTM
64	0.006226935 0.005900551	0.006274577 0.005909435	0.006183659 0.005917765	0.006222571 0.005874453	4156.88 4195.98	minGRU minLSTM
128	0.006331826 0.0036814	0.006361885 0.002866325	0.00629617 0.003422904	0.006337422 0.004754971	2737.23 2812.36	minGRU minLSTM
256	0.006303283 0.003597697	0.006355704 0.00278647	0.00624702 0.003332147	0.006307126 0.004674475	1605.18 1723.13	minGRU minLSTM
512	0.006179736 0.00278958	0.006245122 0.00267526	0.006167604 0.002975405	0.006126482 0.002718074	1113.45 1240.5	minGRU minLSTM
768	0.006172931 0.002799886	0.006262149 0.002698027	0.006198339 0.002978172	0.006058303 0.002723459	1187.3 1275.99	minGRU minLSTM
1024	0.006158548 0.002787876	0.006220552 0.002676733	0.00614728 0.002963503	0.006107811 0.002723393	1390.63 1426.99	minGRU minLSTM

Table 3.78. Batch size results at different MAPE values.

After analyzing Table 3.77 and Table 3.78, the first batch size was chosen to be 1024 since it produced good results with low times.

Following the batch size tests, hidden sizes were computed:

Hidden Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	0.161830251 0.161046836	0.161842428 0.161028953	0.161831508 0.161074594	0.161816816 0.16103696	544.74 641.72	minGRU minLSTM
32	0.162793517 0.161298669	0.162740635 0.161242953	0.162916627 0.161357299	0.162723288 0.161295754	1057.27 642.52	minGRU minLSTM
64	0.161278529 0.161444985	0.161212726 0.16139351	0.161192799 0.1613738016	0.161430063 0.161567644	741.93 914.85	minGRU minLSTM
128	0.132075338 0.07175623	0.132012003 0.069986935	0.132177507 0.071175021	0.132036504 0.074106735	1833.78 3930.11	minGRU minLSTM
256	0.158832404 0.08333292	0.158949744 0.082098758	0.158787026 0.082732514	0.158760441 0.085167489	1170.98 1408.82	minGRU minLSTM
512	0.1814774 0.141704515	0.181005894 0.141914778	0.1809718 0.141684492	0.182454506 0.141514274	3645.51 5691.35	minGRU minLSTM

Table 3.79. Hidden Size results at different RMSE values.

Hidden Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.006168623 0.006180399	0.00616425 0.006179726	0.006178118 0.006188273	0.006163502 0.006173197	544.74 641.72	minGRU minLSTM
32	0.006453303 0.006338705	0.006461364 0.006303192	0.006428953 0.006364378	0.006469592 0.006348544	1057.27 642.52	minGRU minLSTM
64	0.006212527 0.00625102	0.006165386 0.006251661	0.006225811 0.006245647	0.006246386 0.006255753	741.93 914.85	minGRU minLSTM
128	0.006050705 0.003017308	0.006033132 0.002957485	0.006086282 0.00293461	0.006032701 0.003159829	1833.78 3930.11	minGRU minLSTM
256	0.006151594 0.002836621	0.006220203 0.002727415	0.006137707 0.003013457	0.006096872 0.002768991	1170.98 1408.82	minGRU minLSTM
512	0.008992816 0.005591098	0.009029146 0.005429485	0.00885571 0.005731516	0.009093593 0.005612293	3645.51 5691.35	minGRU minLSTM

Table 3.80. Hidden Size results at different MAPE values.

From Table 3.79 and Table 3.80, the hidden size value of 128 was chosen. For the values of 768 and 1024, it was decided not to perform them as after 4 hours of training, it still had not changed the LR with the automatic LR scheduler from 1×10^{-2} .

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

Number of Layers	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1	0.132075338	0.132012003	0.132177507	0.132036504	1833.78	minGRU
	0.07175623	0.069986935	0.071175021	0.074106735	3930.11	minLSTM
2	0.078987568	0.081366637	0.076836545	0.078759522	2922.37	minGRU
	0.164546288	0.164402222	0.164632029	0.164604612	1503.54	minLSTM
3	0.085621368	0.084441061	0.085817178	0.086605864	2418.99	minGRU
	0.162662138	0.162566441	0.162637098	0.162782874	1847.65	minLSTM
4	0.131051879	0.13107907	0.130875738	0.131200827	2093.85	minGRU
	0.160704896	0.160657973	0.160663925	0.160792789	2314.41	minLSTM
5	0.0834041	0.082073278	0.08395812	0.084180903	2821.46	minGRU
	0.161250845	0.161167391	0.161388219	0.161196924	4303.46	minLSTM
6	0.098150734	0.095117501	0.102050043	0.097284656	2818.96	minGRU
	0.162240059	0.162418315	0.162179489	0.162122372	3636.5	minLSTM
7	0.08885855	0.087537879	0.08920488	0.08983289	2876.76	minGRU
	0.161386454	0.16114781	0.161510947	0.161500604	7286.91	minLSTM

Table 3.81. RMSE values obtained in relation to the number of layers.

Number of Layers	Avg MAPE	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1	0.006050705	0.006033132	0.006086282	0.006032701	1833.78	minGRU
	0.003017308	0.002957485	0.00293461	0.003159829	3930.11	minLSTM
2	0.002713911	0.003054317	0.002483682	0.002603733	2922.37	minGRU
	0.006415342	0.006442298	0.00633086	0.006472867	1503.54	minLSTM
3	0.003860834	0.00381879	0.003912467	0.003851245	2418.99	minGRU
	0.006304988	0.006326748	0.006295435	0.006292781	1847.65	minLSTM
4	0.005673859	0.005736716	0.005673161	0.005611701	2093.85	minGRU
	0.006247481	0.006236365	0.006230642	0.006275437	2314.41	minLSTM
5	0.003031671	0.002926965	0.00312278	0.003045268	2821.46	minGRU
	0.006262073	0.006201524	0.00628888	0.006295815	4303.46	minLSTM
6	0.004437625	0.004257132	0.004742267	0.004313477	2818.96	minGRU
	0.006264119	0.006246752	0.006245726	0.006299879	3636.5	minLSTM
7	0.003230541	0.003193387	0.003321379	0.003176858	2876.76	minGRU
	0.006433402	0.006453893	0.006434342	0.006411971	7286.91	minLSTM

Table 3.82. MAPE values obtained in relation to the number of layers.

After testing all these layer numbers, 1 produced the best RMSE and MAPE values compared with the time spent.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)
RNN	0.04280508	0.042069009	0.042105792	0.044240439	4776.25
GRU	0.16791007	0.168015147	0.167796969	0.167918093	5738.75
LSTM	0.168146901	0.168163984	0.168141562	0.168135156	6176.06
BiGRU	0.168308282	0.168339925	0.168265891	0.168319029	12353.12
BiLSTM	0.168322166	0.16837908	0.16827499	0.168312427	16556.64
minGRUseq	0.167555704	0.167546795	0.167556633	0.167563686	7930.38
minGRUpar	0.068439665	0.073714477	0.062403187	0.069201331	4717.5
minLSTMseq	0.167562892	0.167550493	0.167567425	0.167570758	7336.27
minLSTMpar	0.168096065	0.168127141	0.168073655	0.168087399	6261.98
xLSTM	0.167557349	0.167549146	0.167556879	0.167566022	10862.78

Table 3.83. Comparison of models based on RMSE.

Model	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)
RNN	0.001201264	0.001173553	0.001067989	0.001362251	4776.25
GRU	0.006027454	0.005783343	0.006413277	0.005885742	5738.75
LSTM	0.006126797	0.006092895	0.006132854	0.006154642	6176.06
BiGRU	0.006310574	0.006241471	0.006331051	0.006359201	12353.12
BiLSTM	0.005989288	0.00592275	0.006024758	0.006020356	16556.64
minGRUseq	0.006189429	0.006196129	0.006168064	0.006204094	7930.38
minGRUpar	0.002641185	0.003124249	0.002266103	0.002533203	4717.5
minLSTMseq	0.006070154	0.006084463	0.006062668	0.00606333	7336.27
minLSTMpar	0.006253877	0.006290901	0.006160359	0.006310371	6261.98
xLSTM	0.006158146	0.00621392	0.006173963	0.006086555	10862.78

Table 3.84. Comparison of models based on MAPE.

3.5 Sequence Length of 250 and Output of 50

3.5.1 Recurrent Layers with Additional FC Layers of 256, 128, and 64 Neurons

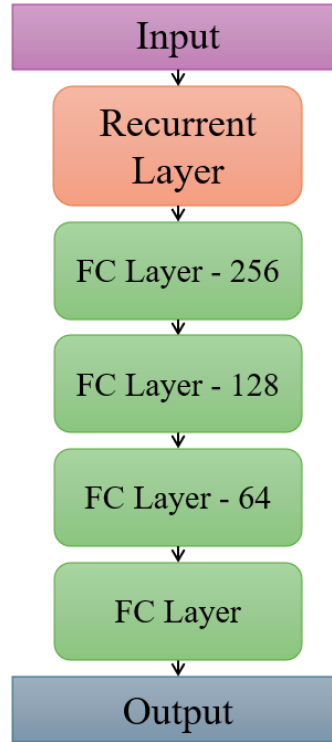


Figure 3.17. Recurrent layer with additional FC layers flowchart.

Starting with a LR of 1×10^{-1} and doing 8 tests of decreasing order:

Learning Rate	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1×10^{-1}	5.968741966	5.964777007	5.97279061	5.968658281	1730.41	minGRU
	1.305377078	1.278417802	1.366577392	1.271136039	887.13	minLSTM
1×10^{-2}	3.492796146	3.491203298	3.493708516	3.493476622	2436.06	minGRU
	0.690115608	0.666400106	0.689405288	0.71454143	2732.3	minLSTM
1×10^{-3}	0.689250523	0.686719117	0.687293505	0.693738948	1165.48	minGRU
	0.472179081	0.469306826	0.474173027	0.47305739	1887.42	minLSTM
1×10^{-4}	0.58983008	0.58601809	0.59836544	0.585106712	4139.09	minGRU
	0.449402644	0.449156536	0.452512301	0.446539096	5150.02	minLSTM
1×10^{-5}	0.212844825	0.211673063	0.214283049	0.212578365	8474.69	minGRU
	0.200151134	0.199965164	0.201028501	0.199459738	11257.99	minLSTM
1×10^{-6}	0.199265413	0.200722912	0.196088356	0.200984971	4132.93	minGRU
	0.198221098	0.200175457	0.194962574	0.199525262	2373.12	minLSTM
1×10^{-7}	0.198831339	0.198307729	0.200862879	0.197323407	2826.16	minGRU
	0.198633186	0.198103359	0.200582982	0.197213218	3562.37	minLSTM
1×10^{-8}	0.19858569	0.199197945	0.1977016	0.198857526	1252.28	minGRU
	0.197747821	0.198749212	0.196666371	0.197827878	2292.48	minLSTM

Table 3.85. LR results at different RMSE values.

Learning Rate	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1×10^{-1}	0.275433865	0.275213201	0.275707222	0.27538117	1730.41	minGRU
	0.052982083	0.050209117	0.058784357	0.049952777	887.13	minLSTM
1×10^{-2}	0.164009852	0.163930613	0.164057482	0.164041462	2436.06	minGRU
	0.032229326	0.031324488	0.032867799	0.032495691	2732.3	minLSTM
1×10^{-3}	0.035690816	0.035655396	0.035515928	0.035901124	1165.48	minGRU
	0.024584497	0.024567445	0.024674308	0.024511738	1887.42	minLSTM
1×10^{-4}	0.030449229	0.030282164	0.030853799	0.030211725	4139.09	minGRU
	0.023204942	0.023186389	0.023296282	0.023132154	5150.02	minLSTM
1×10^{-5}	0.010200764	0.010284138	0.010241311	0.010076843	8474.69	minGRU
	0.009340875	0.009342882	0.009461713	0.009218031	11257.99	minLSTM
1×10^{-6}	0.009184973	0.00919407	0.009041688	0.00931916	4132.93	minGRU
	0.00912179	0.009047086	0.009074384	0.009243902	2373.12	minLSTM
1×10^{-7}	0.009132885	0.009087956	0.009208961	0.009101736	2826.16	minGRU
	0.009200256	0.009188392	0.009271906	0.009140469	3562.37	minLSTM
1×10^{-8}	0.009137954	0.009135196	0.009107573	0.009171093	1252.28	minGRU
	0.009172136	0.009216538	0.009130605	0.009169264	2292.48	minLSTM

Table 3.86. LR results at different MAPE values.

From Table 3.85 and Table 3.86, the LR of 1×10^{-3} was chosen as it produced the best compromise of time and RMSE and MAPE values.

Next, the batch size was iterated through, starting from 16.

Batch Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	0.197688842	0.195911759	0.196866667	0.2002881	16774.47	minGRU
	0.152060478	0.197283972	0.195306974	0.19719091	22496.69	minLSTM
32	0.074203163	0.071671222	0.072319493	0.078618775	9698.26	minGRU
	0.120040517	0.122321354	0.118468668	0.119331531	10135.87	minLSTM
64	0.07325806	0.07111868	0.071463658	0.077191841	6908.46	minGRU
	0.11816186	0.114518768	0.119872146	0.120094667	4223.56	minLSTM
128	0.072805086	0.070104214	0.071052725	0.077258318	2183.34	minGRU
	0.084802248	0.084521481	0.085522491	0.084362773	2445.8	minLSTM
256	0.072445816	0.069906545	0.070691397	0.076739508	2770.96	minGRU
	0.063209835	0.061115302	0.062911119	0.065603083	1331.36	minLSTM
512	0.073570524	0.071263148	0.071709819	0.077738605	1444.41	minGRU
	0.063585316	0.061714509	0.06320074	0.0658407	1043.14	minLSTM
768	0.073155981	0.071237669	0.071289788	0.076940486	1691.08	minGRU
	0.055136092	0.054273545	0.055309993	0.055824737	1167.74	minLSTM
1024	0.074717013	0.072459376	0.073225142	0.07846652	1580.33	minGRU
	0.057694103	0.056333684	0.05807574	0.058672884	1454.5	minLSTM

Table 3.87. Batch size results at different RMSE values.

Batch Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.009107234	0.009033808	0.009073163	0.009214729	16774.47	minGRU
	0.009150673	0.009097929	0.009145605	0.009208485	22496.69	minLSTM
32	0.002931512	0.002829739	0.002810262	0.003154535	9698.26	minGRU
	0.006535562	0.006712274	0.006432746	0.006461668	10135.87	minLSTM
64	0.002927027	0.002837091	0.002813714	0.003130276	6908.46	minGRU
	0.005791066	0.005529679	0.005931393	0.005912127	4223.56	minLSTM
128	0.002915415	0.002809543	0.002800578	0.003136125	2183.34	minGRU
	0.004487355	0.004349989	0.004593029	0.004519046	2445.8	minLSTM
256	0.002920015	0.002823375	0.002802678	0.003133991	2770.96	minGRU
	0.002865048	0.002816365	0.002824442	0.002954336	1331.36	minLSTM
512	0.002951966	0.00284849	0.002828516	0.003178891	1444.41	minGRU
	0.002874259	0.002827105	0.002834852	0.002960819	1043.14	minLSTM
768	0.002949382	0.002858792	0.002836818	0.003152536	1691.08	minGRU
	0.002451774	0.002480465	0.002463499	0.002411357	1167.74	minLSTM
1024	0.00295096	0.002861053	0.002843394	0.003148434	1580.33	minGRU
	0.002485103	0.002519756	0.002497223	0.002438328	1454.5	minLSTM

Table 3.88. Batch size results at different MAPE values.

After analyzing Table 3.87 and Table 3.88, the batch size chosen was 512 as it was a good compromise between results and time.

Following the batch size tests, hidden sizes were computed:

Hidden Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	1.248159729	1.248242095	1.248164454	1.248072637	1748.47	minGRU
	0.783307159	0.783474249	0.783363865	0.783083363	1807.89	minLSTM
32	0.70691859	0.705918889	0.707250376	0.707586505	1501.21	minGRU
	0.785223324	0.786902821	0.785066167	0.783700984	1568.63	minLSTM
64	0.337322075	0.336752403	0.337536442	0.337677381	1602.7	minGRU
	0.241567552	0.240180077	0.242111599	0.24241098	1871.97	minLSTM
128	0.073185969	0.070780856	0.071312663	0.077464387	1299.78	minGRU
	0.056513202	0.054996461	0.056561445	0.0579817	1322	minLSTM
256	0.118838622	0.118993366	0.117536453	0.119986047	2348.42	minGRU
	0.097124156	0.096542805	0.097300111	0.097529554	2487.26	minLSTM
512	0.096366656	0.097587042	0.095724531	0.095788396	3849.64	minGRU
	0.138750962	0.136894106	0.139799663	0.139559118	4883.43	minLSTM
768	0.123536719	0.118325528	0.124892057	0.127392573	6464.55	minGRU
	0.147947	0.144960327	0.154383795	0.144496877	7647.14	minLSTM

Table 3.89. Hidden Size results at different RMSE values.

Hidden Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.056337372	0.056359703	0.056337304	0.056315109	1748.47	minGRU
	0.040516381	0.04048175	0.040503322	0.04056407	1807.89	minLSTM
32	0.035854174	0.035805732	0.035878973	0.035877816	1501.21	minGRU
	0.038911587	0.038938397	0.038901536	0.038894827	1568.63	minLSTM
64	0.014575779	0.014528875	0.01458918	0.014609283	1602.7	minGRU
	0.012371523	0.012302255	0.012377081	0.012435233	1871.97	minLSTM
128	0.002915439	0.002814565	0.002791146	0.003140605	1299.78	minGRU
	0.002443511	0.002472129	0.002450295	0.002408108	1322	minLSTM
256	0.005709609	0.005704088	0.005583578	0.005841161	2348.42	minGRU
	0.004564507	0.004581805	0.004552209	0.004559508	2487.26	minLSTM
512	0.004598974	0.004796335	0.004525403	0.004475185	3849.64	minGRU
	0.006306467	0.006096471	0.006501308	0.006321621	4883.43	minLSTM
768	0.006172849	0.005626322	0.00635776	0.006534466	6464.55	minGRU
	0.006738319	0.00649067	0.007366548	0.006357738	7647.14	minLSTM

Table 3.90. Hidden Size results at different MAPE values.

From Table 3.89 and Table 3.90, the hidden size value of 128 was chosen. Since the test for the hidden size of 768 took just under 4 hours and did not improve the results, the 1024 test was not performed.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

Number of Layers	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1	0.073185969	0.070780856	0.071312663	0.077464387	1299.78	minGRU
	0.056513202	0.054996461	0.056561445	0.0579817	1322	minLSTM
2	0.093167233	0.093505353	0.093397385	0.092598962	1883.19	minGRU
	0.159770746	0.15921142	0.160086639	0.160014178	3017	minLSTM
3	0.171287114	0.170733754	0.170658821	0.172468768	2086.84	minGRU
	0.189658392	0.189866305	0.18704938	0.192059491	3624.51	minLSTM
4	0.149801363	0.14948977	0.149791001	0.150123319	2051.67	minGRU
	0.180717408	0.180427334	0.180804321	0.180920567	4901.7	minLSTM
5	0.184444356	0.183607354	0.185023093	0.184702621	2041.96	minGRU
	0.145312264	0.144956982	0.145166798	0.14581301	5004	minLSTM
6	0.126353797	0.126105575	0.126169929	0.126785888	1868.83	minGRU
	0.152263177	0.151690944	0.152499754	0.152598832	5650.4	minLSTM
7	0.115237328	0.113475766	0.11555927	0.11667695	2099.35	minGRU
	0.128225371	0.12878358	0.128429593	0.127462941	8110.8	minLSTM

Table 3.91. RMSE values obtained in relation to the number of layers.

Number of Layers	Avg MAPE	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1	0.002915439	0.002814565	0.002791146	0.003140605	1299.78	minGRU
	0.002443511	0.002472129	0.002450295	0.002408108	1322	minLSTM
2	0.004739884	0.004837016	0.004750179	0.004632457	1883.19	minGRU
	0.007162198	0.007140759	0.00717974	0.007166095	3017	minLSTM
3	0.006579211	0.006493481	0.006553278	0.006690873	2086.84	minGRU
	0.010305408	0.010314113	0.010167155	0.010434957	3624.51	minLSTM
4	0.005580386	0.005697861	0.005601399	0.005441898	2051.67	minGRU
	0.009462679	0.009454018	0.009463581	0.009470438	4901.7	minLSTM
5	0.008382814	0.008330156	0.008458669	0.008359617	2041.96	minGRU
	0.007295246	0.007283484	0.007286295	0.007315959	5004	minLSTM
6	0.005974676	0.005997882	0.005972559	0.005953586	1868.83	minGRU
	0.007423237	0.007407517	0.007430871	0.007431323	5650.4	minLSTM
7	0.004772686	0.004760857	0.004707314	0.004849887	2099.35	minGRU
	0.00643494	0.006555702	0.006486641	0.006262476	8110.8	minLSTM

Table 3.92. MAPE values obtained in relation to the number of layers.

After testing all these layer numbers, 1 produced the best RMSE and MAPE values compared with the time spent.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)
RNN	0.16760154	0.16757949	0.167615978	0.167609152	2932.98
GRU	0.167600696	0.167578533	0.167614724	0.167608832	3247.75
LSTM	0.167602403	0.167579956	0.167616086	0.167611167	3847.92
BiGRU	0.167601892	0.167578367	0.167617257	0.167610051	4841.13
BiLSTM	0.167602979	0.167581156	0.16761727	0.16761051	4618.97
minGRUseq	0.167481311	0.167509813	0.167505267	0.167428852	4986.73
minGRUpar	0.167601262	0.16757831	0.16761665	0.167608825	3510.36
minLSTMseq	0.167525431	0.167547919	0.167552016	0.16747636	5634.12
minLSTMpar	0.167601023	0.167578411	0.167615015	0.167609643	3875.77
xLSTM	0.167482118	0.167510876	0.167506637	0.16742884	6282.86

Table 3.93. Comparison of models based on RMSE.

Model	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)
RNN	0.006187573	0.006189223	0.006195127	0.006178368	2932.98
GRU	0.00616791	0.006171501	0.006165652	0.006166576	3247.75
LSTM	0.006124811	0.006124418	0.006130399	0.006119615	3847.92
BiGRU	0.006138533	0.00616559	0.00611867	0.006131339	4841.13
BiLSTM	0.006118948	0.006112705	0.006118672	0.006125468	4618.97
minGRUseq	0.006142859	0.006135761	0.00614778	0.006145035	4986.73
minGRUpar	0.006150265	0.006159682	0.006124538	0.006166575	3510.36
minLSTMseq	0.006337801	0.00632673	0.006344738	0.006341936	5634.12
minLSTMpar	0.006146328	0.006153784	0.006147996	0.006137203	3875.77
xLSTM	0.00612915	0.006118111	0.006118399	0.006150941	6282.86

Table 3.94. Comparison of models based on MAPE.

3.5.2 Recurrent Layers with Additional FC Layers of 128 and 64 Neurons

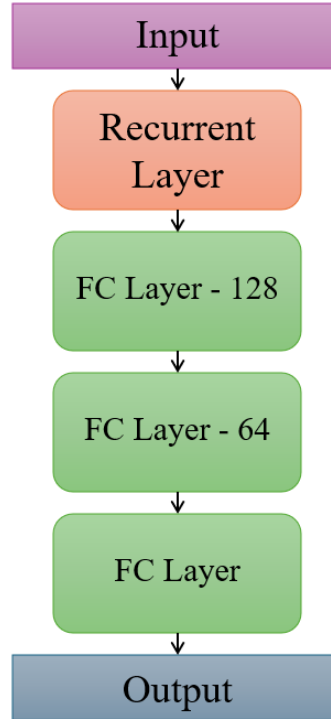


Figure 3.18. Recurrent layer with additional FC layers flowchart.

Starting with a LR of 1×10^{-1} and doing 7 tests of decreasing order:

Learning Rate	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1×10^{-1}	0.161037637	0.161109125	0.16110487	0.160898915	756	minGRU
	0.161027722	0.161110518	0.16111647	0.160856176	936.34	minLSTM
1×10^{-2}	0.162007199	0.161876485	0.162137805	0.162007307	495.38	minGRU
	0.162011907	0.161891918	0.16219165	0.161952153	607.57	minLSTM
1×10^{-3}	0.165927778	0.166243061	0.165700743	0.16583953	495.74	minGRU
	0.165965831	0.166297317	0.165850213	0.165749964	795.44	minLSTM
1×10^{-4}	0.157713737	0.157684221	0.15756139	0.157895599	727.66	minGRU
	0.157683877	0.157666739	0.157515716	0.157869178	797.45	minLSTM
1×10^{-5}	0.160252935	0.160218536	0.160167544	0.160372727	1530.84	minGRU
	0.160245029	0.160222833	0.160183988	0.160328265	2814.36	minLSTM
1×10^{-6}	0.159267774	0.159276619	0.159214724	0.15931198	2794.33	minGRU
	0.159246819	0.159269221	0.159195434	0.159275801	1690.16	minLSTM
1×10^{-7}	0.161424152	0.161217435	0.161362466	0.161692555	4336.56	minGRU
	0.161427644	0.161233543	0.161414743	0.161634647	3663.65	minLSTM

Table 3.95. LR results at different RMSE values.

Learning Rate	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1×10^{-1}	0.00620234	0.006260961	0.006341444	0.006004617	756	minGRU
	0.006109903	0.006165492	0.006073971	0.006090246	936.34	minLSTM
1×10^{-2}	0.006222406	0.006273356	0.006364577	0.006029284	495.38	minGRU
	0.006129993	0.006178331	0.006096823	0.006114825	607.57	minLSTM
1×10^{-3}	0.006365246	0.006432561	0.006491493	0.006171684	495.74	minGRU
	0.00627508	0.006339788	0.006230681	0.006254771	795.44	minLSTM
1×10^{-4}	0.006090068	0.006146698	0.006219307	0.005904198	727.66	minGRU
	0.005996714	0.006050209	0.005949719	0.005990213	797.45	minLSTM
1×10^{-5}	0.006182516	0.00623619	0.006315938	0.00599542	1530.84	minGRU
	0.006090086	0.006141392	0.006048727	0.006080139	2814.36	minLSTM
1×10^{-6}	0.006139163	0.006194243	0.006274453	0.005948794	2794.33	minGRU
	0.006046553	0.006098853	0.006006688	0.006034118	1690.16	minLSTM
1×10^{-7}	0.006215784	0.006261761	0.006351209	0.006034382	4336.56	minGRU
	0.006123919	0.006167074	0.006085248	0.006119434	3663.65	minLSTM

Table 3.96. LR results at different MAPE values.

From Table 3.95 and Table 3.96, the LR of 1×10^{-4} was chosen as it produced the best result. Next, the batch size was iterated through, starting from 16.

Batch Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	
16	0.075451961	0.074132978	0.075919821	0.076303085	20599.06	minGRU
	0.080131779	0.078376257	0.085353421	0.076665661	22836.39	minLSTM
32	0.065143282	0.063618011	0.06527987	0.066531964	14237.23	minGRU
	0.066036937	0.065288596	0.065296708	0.067525508	17396.32	minLSTM
64	0.065104861	0.063033079	0.065854208	0.066427294	6001.76	minGRU
	0.065877115	0.064712025	0.065710118	0.067209203	4151.04	minLSTM
128	0.062828583	0.061109586	0.06383103	0.063545133	2321.43	minGRU
	0.065214228	0.063470211	0.065011568	0.067160906	2546.7	minLSTM
256	0.063120867	0.061684574	0.063811962	0.063866065	1723.67	minGRU
	0.065378439	0.063992524	0.064872875	0.067269918	1999.05	minLSTM
512	0.063587489	0.061872269	0.064298207	0.064591991	1174.52	minGRU
	0.065335086	0.063619344	0.064739941	0.067645971	1409.22	minLSTM
768	0.064672091	0.063275221	0.065528833	0.06521222	1363.2	minGRU
	0.066067116	0.064726852	0.065926607	0.067547889	1080.36	minLSTM
1024	0.157713737	0.157684221	0.15756139	0.157895599	727.66	minGRU
	0.157683877	0.157666739	0.157515716	0.157869178	797.45	minLSTM

Table 3.97. Batch size results at different RMSE values.

Batch Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	
16	0.003279306	0.003254929	0.00334142	0.003241569	20599.06	minGRU
	0.00359631	0.003566761	0.004123036	0.003099133	22836.39	minLSTM
32	0.002208026	0.002229473	0.002211044	0.00218356	14237.23	minGRU
	0.002228112	0.002278114	0.00217759	0.002228631	17396.32	minLSTM
64	0.002197927	0.002191146	0.002221882	0.002180753	6001.76	minGRU
	0.002209277	0.002250571	0.002168353	0.002208907	4151.04	minLSTM
128	0.002003465	0.001959223	0.002133899	0.001917274	2321.43	minGRU
	0.002130462	0.002090056	0.002097469	0.00220386	2546.7	minLSTM
256	0.002038474	0.001998982	0.002171691	0.001944749	1723.67	minGRU
	0.002149993	0.002112935	0.002116227	0.002220817	1999.05	minLSTM
512	0.002046709	0.00200363	0.002173642	0.001962855	1174.52	minGRU
	0.002155348	0.00211235	0.002120583	0.002233111	1409.22	minLSTM
768	0.002071576	0.002033631	0.00220796	0.001973139	1363.2	minGRU
	0.002148275	0.002130424	0.002142976	0.002171426	1080.36	minLSTM
1024	0.006090068	0.006146698	0.006219307	0.005904198	727.66	minGRU
	0.005996714	0.006050209	0.005949719	0.005990213	797.45	minLSTM

Table 3.98. Batch size results at different MAPE values.

After analyzing Table 3.97 and Table 3.98, the batch size chosen was 512 as it is a good compromise between results and time.

Following the batch size tests, hidden sizes were computed:

Hidden Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	
16	0.162073415 0.162050115	0.161881349 0.161855326	0.161934418 0.161919942	0.162404478 0.162375078	874.52 1106.18	minGRU minLSTM
32	0.161000849 0.160993237	0.160833684 0.160832515	0.161163243 0.161159828	0.16100562 0.160987369	1043.12 1172.39	minGRU minLSTM
64	0.163788644 0.163989463	0.163804245 0.164037685	0.163617751 0.163776747	0.163943936 0.164153958	1223.66 1157.39	minGRU minLSTM
128	0.063891435 0.065021857	0.062308099 0.063456956	0.064556849 0.064714438	0.064809358 0.066894176	1477.68 1173.44	minGRU minLSTM
256	0.160464723 0.160633921	0.160285562 0.16042984	0.160524842 0.160659538	0.160583764 0.160812386	1364.48 1720.58	minGRU minLSTM
512	0.161135749 0.160403059	0.161249137 0.160508693	0.161132118 0.160423527	0.161025992 0.160276956	2668.13 2746.25	minGRU minLSTM
768	0.162791957 0.163377333	0.162713374 0.163365862	0.162959107 0.1635227	0.162703392 0.163243438	4832.77 6068.22	minGRU minLSTM

Table 3.99. Hidden Size results at different RMSE values.

Hidden Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	
16	0.006248177 0.00633942	0.006241974 0.006332449	0.006254687 0.006335927	0.00624787 0.006349882	874.52 1106.18	minGRU minLSTM
32	0.006219599 0.006265325	0.006199541 0.006253107	0.00623234 0.006287732	0.006226917 0.006255136	1043.12 1172.39	minGRU minLSTM
64	0.006508094 0.006529471	0.006478148 0.00652974	0.006515477 0.006522642	0.006530657 0.006536031	1223.66 1157.39	minGRU minLSTM
128	0.002068459 0.002117643	0.002026257 0.002094699	0.002201055 0.002109383	0.001978067 0.002148847	1477.68 1173.44	minGRU minLSTM
256	0.006228379 0.006265289	0.006232938 0.006252874	0.006230178 0.006266034	0.00622202 0.006276958	1364.48 1720.58	minGRU minLSTM
512	0.006286203 0.006170053	0.006280935 0.006169446	0.006314404 0.006171735	0.006263271 0.006168977	2668.13 2746.25	minGRU minLSTM
768	0.006207218 0.006202071	0.006238843 0.006143728	0.006190128 0.006191438	0.006192683 0.006271049	4832.77 6068.22	minGRU minLSTM

Table 3.100. Hidden Size results at different MAPE values.

From Table 3.99 and Table 3.100, the hidden size value of 256 was chosen. Since the test for hidden size of 768 took just over 3 hours and did not improve the results, 1024 test was not performed.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

Number of Layers	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	
1	0.160464723 0.160633921	0.160285562 0.16042984	0.160524842 0.160659538	0.160583764 0.160812386	1364.48 1720.58	minGRU minLSTM
2	0.163136216 0.162957642	0.163116312 0.162973318	0.163314859 0.163095055	0.162977477 0.162804552	1022.75 2192.54	minGRU minLSTM
3	0.165099871 0.164427559	0.165128653 0.164446202	0.165040783 0.164378837	0.165130176 0.164457639	1826.63 2532.02	minGRU minLSTM
4	0.162791914 0.162218381	0.162720841 0.162133638	0.162710412 0.162139627	0.162944488 0.162381876	1894.55 3271.96	minGRU minLSTM
5	0.159409743 0.159416066	0.159282133 0.159298334	0.15943564 0.159441726	0.159511456 0.159508137	1523.68 4825.65	minGRU minLSTM
6	0.160887553 0.160508153	0.160753099 0.160378515	0.160943856 0.160532587	0.160965706 0.160613355	1238.73 4526.8	minGRU minLSTM
7	0.163196852 0.163255221	0.163198916 0.163240623	0.163310188 0.163381347	0.163081453 0.163143694	1750.48 5190.57	minGRU minLSTM

Table 3.101. RMSE values obtained in relation to the number of layers.

Number of Layers	Avg MAPE	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	
1	0.006228379	0.006232938	0.006230178	0.00622202	1364.48	minGRU
	0.006265289	0.006252874	0.006266034	0.006276958	1720.58	minLSTM
2	0.006255848	0.006253898	0.00625866	0.006254985	1022.75	minGRU
	0.006341521	0.00633516	0.006353336	0.006336066	2192.54	minLSTM
3	0.006445177	0.006452806	0.006453057	0.00642967	1826.63	minGRU
	0.006354419	0.006372701	0.006353407	0.006337148	2532.02	minLSTM
4	0.006389957	0.00640175	0.006384928	0.006383194	1894.55	minGRU
	0.006327794	0.006303855	0.006340942	0.006338586	3271.96	minLSTM
5	0.006137206	0.006115716	0.006173515	0.006122388	1523.68	minGRU
	0.006105551	0.00607962	0.006102164	0.006102164	4825.65	minLSTM
6	0.006229916	0.006223274	0.006229229	0.006237244	1238.73	minGRU
	0.006171799	0.006171894	0.006151398	0.006192105	4526.8	minLSTM
7	0.006263792	0.006275756	0.006269938	0.006245681	1750.48	minGRU
	0.006289387	0.006290195	0.00628802	0.006289946	5190.57	minLSTM

Table 3.102. MAPE values obtained in relation to the number of layers.

After testing all these layer numbers, 1 produced the best RMSE and MAPE values compared with the time spent.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)
RNN	0.160234558	0.161834916	0.159101283	0.159767475	6085.7
GRU	0.119979392	0.119568497	0.120274708	0.12009497	9227.27
LSTM	0.131948251	0.131600907	0.132100522	0.132143325	12287.57
BiGRU	0.131274914	0.131216044	0.131335915	0.131272784	13326.11
BiLSTM	0.14057846	0.140933854	0.140088762	0.140712765	14709.27
minGRUseq	0.097651228	0.097671112	0.09765221	0.097630363	12098.01
minGRUpar	0.144554131	0.14418452	0.145253148	0.144224725	7362.85
minLSTMseq	0.140786053	0.140719472	0.14078021	0.140858476	7419.07
minLSTMpar	0.140643258	0.140148914	0.140872328	0.14090853	7362.87
xLSTM	0.168229073	0.168208938	0.168220822	0.168257458	6322.58

Table 3.103. Comparison of models based on RMSE.

Model	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)
RNN	0.005602643	0.005311066	0.005873494	0.005623368	6085.7
GRU	0.005062632	0.005020615	0.00508903	0.005078251	9227.27
LSTM	0.00627761	0.00622832	0.006302149	0.006302362	12287.57
BiGRU	0.005597891	0.005634435	0.005591616	0.005567622	13326.11
BiLSTM	0.005659341	0.005647083	0.005672946	0.005657995	14709.27
minGRUseq	0.004012112	0.004001077	0.004005053	0.004030207	12098.01
minGRUpar	0.005282584	0.005297011	0.005253762	0.00529698	7362.85
minLSTMseq	0.006293434	0.006316948	0.006283837	0.006279517	7419.07
minLSTMpar	0.005634376	0.005708674	0.005598185	0.005596268	7362.87
xLSTM	0.006204054	0.006184793	0.006209357	0.006218011	6322.58

Table 3.104. Comparison of models based on MAPE.

3.5.3 Recurrent Layers

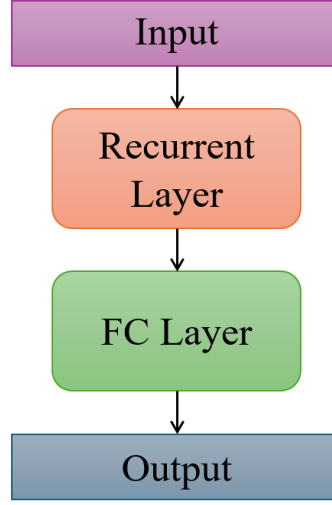


Figure 3.19. Recurrent layer flowchart.

Starting with a LR of 1×10^{-1} and doing 8 tests of decreasing order:

Learning Rate	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1×10^{-1}	0.165257392	0.165210556	0.165261235	0.165300385	933.82	minGRU
	0.165271776	0.165239684	0.165250808	0.165324837	1257.15	minLSTM
1×10^{-2}	0.15906735	0.158878494	0.159049389	0.159274166	983.36	minGRU
	0.159107275	0.158942336	0.159050873	0.159328615	1262.48	minLSTM
1×10^{-3}	0.164095079	0.164139261	0.164056966	0.164089009	878.56	minGRU
	0.164090181	0.164139209	0.164055276	0.164076059	1266.94	minLSTM
1×10^{-4}	0.161817954	0.161802511	0.161926872	0.161724477	2991.95	minGRU
	0.161711472	0.161687256	0.161816866	0.161630294	5261.17	minLSTM
1×10^{-5}	0.16033289	0.160392493	0.160462509	0.16014367	667.43	minGRU
	0.160230923	0.160282316	0.160354243	0.160056209	813.2	minLSTM
1×10^{-6}	0.16131954	0.161358255	0.16125027	0.161350094	673.32	minGRU
	0.161216282	0.161246149	0.161140556	0.161262141	818.09	minLSTM
1×10^{-7}	0.162518204	0.16251294	0.162620096	0.162421578	679	minGRU
	0.162402309	0.162386178	0.162494556	0.162326194	816.58	minLSTM
1×10^{-8}	0.161477629	0.161227463	0.161519656	0.161685769	676.89	minGRU
	0.161369207	0.161110036	0.161404515	0.16159307	859.44	minLSTM

Table 3.105. LR results at different RMSE values.

Learning Rate	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1×10^{-1}	0.006414851	0.006642587	0.006386603	0.006215365	933.82	minGRU
	0.006680584	0.006721728	0.006409081	0.006910943	1257.15	minLSTM
1×10^{-2}	0.006168663	0.006171805	0.006182623	0.00615156	983.36	minGRU
	0.006285184	0.006329903	0.006188852	0.006336798	1262.48	minLSTM
1×10^{-3}	0.006324145	0.006335908	0.006336993	0.006299534	878.56	minGRU
	0.006439486	0.006492496	0.006343128	0.006482834	1266.94	minLSTM
1×10^{-4}	0.006259061	0.006253358	0.006270348	0.006253477	2991.95	minGRU
	0.006293714	0.006293909	0.006305239	0.006281993	5261.17	minLSTM
1×10^{-5}	0.006193658	0.006190466	0.006205308	0.0061852	667.43	minGRU
	0.006228548	0.006231283	0.006240347	0.006214015	813.2	minLSTM
1×10^{-6}	0.006232349	0.006230018	0.006235514	0.006231516	673.32	minGRU
	0.006266926	0.00627041	0.006270131	0.006260238	818.09	minLSTM
1×10^{-7}	0.006262597	0.006257879	0.006275376	0.006254535	679	minGRU
	0.006297027	0.00629816	0.006309797	0.006283125	816.58	minLSTM
1×10^{-8}	0.006245966	0.006236449	0.006252033	0.006249416	676.89	minGRU
	0.006280422	0.00627679	0.006286645	0.006277831	859.44	minLSTM

Table 3.106. LR results at different MAPE values.

From Table 3.105 and Table 3.106, the LR of 1×10^{-2} was chosen as it produced the best result.

Next, the batch size was iterated through, starting from 64.

Batch Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
64	0.162269011	0.162505303	0.16237997	0.161921759	4168.33	minGRU
	0.162185995	0.162448612	0.162275825	0.161833549	6262.01	minLSTM
128	0.159443501	0.15964985	0.15936102	0.159319634	2143.57	minGRU
	0.159353392	0.159562753	0.1592756	0.159221823	2385.62	minLSTM
256	0.158618326	0.158703084	0.158812764	0.15833913	1360.77	minGRU
	0.154007235	0.153599413	0.154492812	0.15392948	1468.68	minLSTM
512	0.164963842	0.164967545	0.16499842	0.16492556	1151.21	minGRU
	0.160463393	0.159980977	0.16073629	0.160672912	1156.42	minLSTM
768	0.160139577	0.160447704	0.160053104	0.159917924	855.57	minGRU
	0.15552427	0.155339861	0.155698509	0.155534438	947.09	minLSTM
1024	0.15906735	0.158878494	0.159049389	0.159274166	983.36	minGRU
	0.159107275	0.158942336	0.159050873	0.159328615	1262.48	minLSTM

Table 3.107. Batch size results at different RMSE values.

Batch Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
64	0.006302082	0.006304923	0.006323376	0.006277946	4168.33	minGRU
	0.006295015	0.006275063	0.006318445	0.006291538	6262.01	minLSTM
128	0.006184169	0.006194117	0.006185813	0.006172576	2143.57	minGRU
	0.006191405	0.006170043	0.006214287	0.006189885	2385.62	minLSTM
256	0.006149417	0.006132426	0.006196686	0.006119139	1360.77	minGRU
	0.00548618	0.005636633	0.005412805	0.005409101	1468.68	minLSTM
512	0.00642125	0.006489646	0.006341329	0.006432776	1151.21	minGRU
	0.005745232	0.005869083	0.00568272	0.005683892	1156.42	minLSTM
768	0.006222703	0.006308592	0.006148603	0.006210915	855.57	minGRU
	0.005532876	0.005686925	0.00545015	0.005461553	947.09	minLSTM
1024	0.006168663	0.006171805	0.006182623	0.00615156	983.36	minGRU
	0.006285184	0.006329903	0.006188852	0.006336798	1262.48	minLSTM

Table 3.108. Batch size results at different MAPE values.

After analyzing Table 3.107 and Table 3.108, the batch size chosen was 768 as it was a good compromise between results and time.

Following the batch size tests, hidden sizes were computed:

Hidden Size	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
16	0.159544314	0.15951186	0.159565568	0.159555514	346.32	minGRU
	0.159537714	0.159502941	0.159561579	0.159548622	521.57	minLSTM
32	0.161704849	0.161614574	0.161576916	0.161923056	412.85	minGRU
	0.161704845	0.161614847	0.161577458	0.16192223	421.06	minLSTM
64	0.164017333	0.164149189	0.164050313	0.163852498	484.62	minGRU
	0.164015517	0.164143891	0.164048043	0.163854616	501.09	minLSTM
128	0.16267171	0.162745641	0.162570776	0.162698713	786.82	minGRU
	0.162676104	0.162749306	0.162576088	0.162702917	1070.88	minLSTM
256	0.161027566	0.161028468	0.16098134	0.16107289	899.91	minGRU
	0.161021897	0.160996684	0.160981989	0.161087018	905.15	minLSTM
512	0.162253802	0.162007983	0.162348742	0.162404682	1674.33	minGRU
	0.162214004	0.161998915	0.162292941	0.162350155	2190.06	minLSTM
768	0.161249601	0.161277982	0.161179313	0.161291508	3927.98	minGRU
	0.161227816	0.161274546	0.161116983	0.161291921	4421.07	minLSTM

Table 3.109. Hidden Size results at different RMSE values.

Hidden Size	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
16	0.006239706	0.006241082	0.006232492	0.006245544	346.32	minGRU
	0.006215205	0.006212458	0.006215637	0.006217519	521.57	minLSTM
32	0.006272994	0.006279309	0.006276812	0.006262862	412.85	minGRU
	0.006284008	0.006279027	0.006287946	0.00628505	421.06	minLSTM
64	0.006331149	0.006322723	0.006331691	0.006339034	484.62	minGRU
	0.006336628	0.006339263	0.006342529	0.006328091	501.09	minLSTM
128	0.006331935	0.006345729	0.00634134	0.006308736	786.82	minGRU
	0.006290878	0.006311768	0.006279868	0.006280998	1070.88	minLSTM
256	0.00608541	0.006041222	0.006100134	0.006114874	899.91	minGRU
	0.006095852	0.006105969	0.006099439	0.006082147	905.15	minLSTM
512	0.006198346	0.006257713	0.006171663	0.006165662	1674.33	minGRU
	0.006347474	0.006329457	0.00638899	0.006323976	2190.06	minLSTM
768	0.006166913	0.006216471	0.005994941	0.006289329	3927.98	minGRU
	0.006326984	0.00627174	0.006420107	0.006289104	4421.07	minLSTM

Table 3.110. Hidden Size results at different MAPE values.

From Table 3.109 and Table 3.110, the hidden size value of 16 was chosen. Since the test for hidden size of 768 took over 2 hours and did not improve the results, 1024 test was not performed.

Finally, the number of layers was chosen by iterating from 1 layer to 7 stacked layers:

Number of Layers	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)	Model
1	0.159544314	0.15951186	0.159565568	0.159555514	346.32	minGRU
	0.159537714	0.159502941	0.159561579	0.159548622	521.57	minLSTM
2	0.155772399	0.155746922	0.1557696	0.155800676	372.71	minGRU
	0.155775349	0.155746949	0.155757403	0.155821695	407.27	minLSTM
3	0.161738699	0.16185054	0.161754533	0.161611024	406.58	minGRU
	0.161765374	0.161878893	0.161782939	0.161634291	516.2	minLSTM
4	0.159177058	0.159139556	0.159168762	0.159222857	385.99	minGRU
	0.159214085	0.159159619	0.159215239	0.159267398	553.27	minLSTM
5	0.162652553	0.162500787	0.162628236	0.162828637	388.21	minGRU
	0.162651454	0.162499731	0.162626517	0.162828115	465.1	minLSTM
6	0.159891443	0.159905638	0.15997153	0.15979716	452.01	minGRU
	0.159884988	0.159897558	0.159967205	0.159790201	535.04	minLSTM
7	0.159491705	0.159436026	0.159517482	0.159521607	434.85	minGRU
	0.15949853	0.159441256	0.1595251	0.159529233	518.83	minLSTM

Table 3.111. RMSE values obtained in relation to the number of layers.

Number of Layers	Avg MAPE	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)	Model
1	0.006239706	0.006241082	0.006232492	0.006245544	346.32	minGRU
	0.006215205	0.006212458	0.006215637	0.006217519	521.57	minLSTM
2	0.006155822	0.00615971	0.00615997	0.006147788	372.71	minGRU
	0.006159721	0.006159768	0.00614294	0.006176455	407.27	minLSTM
3	0.006198916	0.006209582	0.006200612	0.006186553	406.58	minGRU
	0.00613736	0.006144667	0.006135146	0.006132267	516.2	minLSTM
4	0.006221997	0.006225006	0.006213325	0.00622766	385.99	minGRU
	0.006300095	0.006270774	0.006310229	0.00631928	553.27	minLSTM
5	0.006315898	0.006309461	0.006310821	0.006327411	388.21	minGRU
	0.006327071	0.006326124	0.006333042	0.006322048	465.1	minLSTM
6	0.006261102	0.006267893	0.006271471	0.006243941	452.01	minGRU
	0.00624227	0.006245491	0.006259996	0.006221324	535.04	minLSTM
7	0.006149591	0.006158724	0.006139532	0.006150518	434.85	minGRU
	0.006199992	0.006192349	0.006200903	0.006206722	518.83	minLSTM

Table 3.112. MAPE values obtained in relation to the number of layers.

After testing all these layer numbers, 2 produced the best RMSE and MAPE values compared with the time spent.

With the hyperparameters found, the test with all of the different models and all the data was conducted.

Model	Avg RMSE	RMSE at 1	RMSE at 25	RMSE at 50	Time (s)
RNN	0.167031946	0.167008629	0.167072974	0.167014234	1866.53
GRU	0.16702166	0.166992585	0.167061046	0.16701135	2009.22
LSTM	0.1670222	0.16699353	0.167061096	0.167011974	1957.82
BiGRU	0.167052657	0.167050581	0.167089629	0.167017759	2393.97
BiLSTM	0.167034407	0.167020524	0.167064864	0.167017833	2565.52
minGRUseq	0.166847611	0.16682499	0.166855168	0.166862677	2848.2
minGRUpar	0.167021945	0.166992322	0.167061825	0.167011688	1788.03
minLSTMseq	0.166848944	0.166824924	0.166859647	0.166862262	3081.15
minLSTMpar	0.167021494	0.166992203	0.167060958	0.16701132	1967.98
xLSTM	0.166847824	0.166825295	0.166855501	0.166862677	4586.87

Table 3.113. Comparison of models based on RMSE.

Model	Avg MAPE (%)	MAPE at 1	MAPE at 25	MAPE at 50	Time (s)
RNN	0.006033094	0.006004681	0.006024749	0.006069851	1866.53
GRU	0.006115008	0.006097886	0.006130292	0.006116846	2009.22
LSTM	0.006140577	0.006150929	0.006130337	0.006140464	1957.82
BiGRU	0.005975533	0.005907455	0.005972727	0.006046417	2393.97
BiLSTM	0.006027382	0.005970175	0.006065486	0.006046485	2565.52
minGRUseq	0.006130216	0.006128652	0.006124694	0.006137301	2848.2
minGRUpar	0.006099253	0.006103745	0.006094881	0.006099133	1788.03
minLSTMseq	0.006144076	0.006128696	0.006172087	0.006131444	3081.15
minLSTMpar	0.006118922	0.006121489	0.006118455	0.006116823	1967.98
xLSTM	0.006134155	0.006134605	0.006130558	0.006137301	4586.87

Table 3.114. Comparison of models based on MAPE.

3.6 Virtual Sensing

Since recurrent layers are not as useful during virtual sensing, only CNN were used alongside FC layers.

3.6.1 (Convolutional Layer + ReLU + MaxPool) x 1

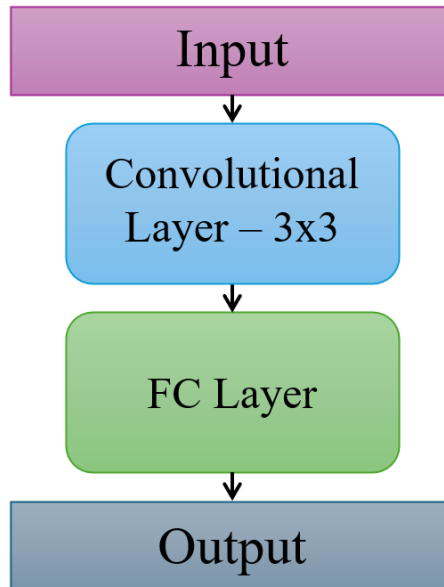


Figure 3.20. Flowchart of CNN with one convolutional layer.

LR	RMSE	MAPE (%)	Time (s)
1×10^{-1}	0.031408283	0.000897735	390.62
1×10^{-2}	0.029200599	0.000778278	402
1×10^{-3}	0.028954628	0.000749542	227.78
1×10^{-4}	0.030796259	0.000761768	173.93
1×10^{-5}	0.032404434	0.000766427	101.48
1×10^{-6}	0.027931359	0.000758681	125.17
1×10^{-7}	0.031822888	0.000774378	191.83
1×10^{-8}	0.03008611	0.000760616	101.51

Table 3.115. Learning Rate Testing

Batch Size	RMSE	MAPE (%)	Time (s)
16	0.020775285	0.000500838	1479.54
32	0.019610396	0.000489401	377.58
64	0.018120286	0.000487454	184.26
128	0.020292982	0.000501683	182.59
256	0.018609377	0.000489001	191.71
512	0.019055771	0.000488448	212.17
768	0.020277756	0.000486711	180.58
1024	0.026265273	0.000904328	44.3

Table 3.116. Batch Size Testing

3.6.2 Recurrent Layers with Additional FC Layers of 256, 128, and 64 Neurons

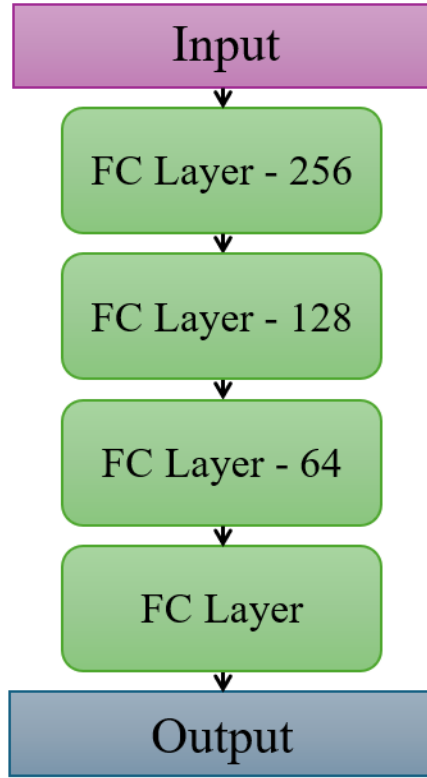


Figure 3.21. Flowchart of FC of 256, 128 and 64 neurons.

LR	RMSE	MAPE (%)	Time (s)
1×10^{-1}	0.026825504	0.000920302	109.85
1×10^{-2}	0.026455866	0.000910328	57.33
1×10^{-3}	0.026265273	0.000904328	44.3
1×10^{-4}	0.02665201	0.000917169	46.13
1×10^{-5}	0.026365186	0.000904628	35.72
1×10^{-6}	0.026377246	0.000909067	46.62
1×10^{-7}	0.026435644	0.00091309	38.24
1×10^{-8}	0.026283107	0.000901377	35.9

Table 3.117. Learning Rate Testing

Batch Size	RMSE	MAPE (%)	Time (s)
16	0.026835709	0.000924671	878.74
32	0.026590538	0.000915673	366.35
64	0.026519258	0.000913048	140.72
128	0.026369866	0.000911156	77.25
256	0.026314141	0.000909362	108.13
512	0.026761872	0.000918375	63.76
768	0.026432665	0.000907644	48.46
1024	0.026265273	0.000904328	44.3

Table 3.118. Batch Size Testing

The LR chosen was 1×10^{-3} and a batch size of 1024.

3.6.3 Recurrent Layers with Additional FC Layers of 128 and 64 Neurons

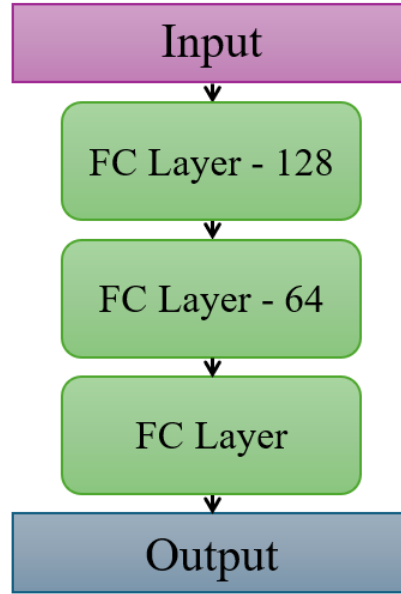


Figure 3.22. Flowchart of FC of 128 and 64 neurons.

LR	RMSE	MAPE (%)	Time (s)
1×10^{-1}	0.013115618	0.000394867	82.53
1×10^{-2}	0.012693357	0.000386408	68.5
1×10^{-3}	0.012487616	0.000385623	74.14
1×10^{-4}	0.012304049	0.000379213	60.05
1×10^{-5}	0.012164271	0.000375153	54.15
1×10^{-6}	0.012344102	0.00037885	33.28
1×10^{-7}	0.012176937	0.00037221	45.66
1×10^{-8}	0.012489267	0.000380688	94.98

Table 3.119. Learning Rate Testing

Batch Size	RMSE	MAPE (%)	Time (s)
16	0.01220961	0.000374274	1365.46
32	0.012300541	0.000375728	379.29
64	0.012031349	0.000370117	216.92
128	0.012501846	0.000380804	123.03
256	0.012188901	0.000373806	99.63
512	0.012137726	0.000370688	44.69
768	0.01219423	0.000372869	57.2
1024	0.012164271	0.000375153	54.15

Table 3.120. Batch Size Testing

After performing more than 250 epochs with a LR of 1×10^{-5} , it was changed to 1×10^{-4} since there was no improvement. The best batch size was found to be 512.

3.6.4 Recurrent Layers with Additional FC Layer of 64 Neurons

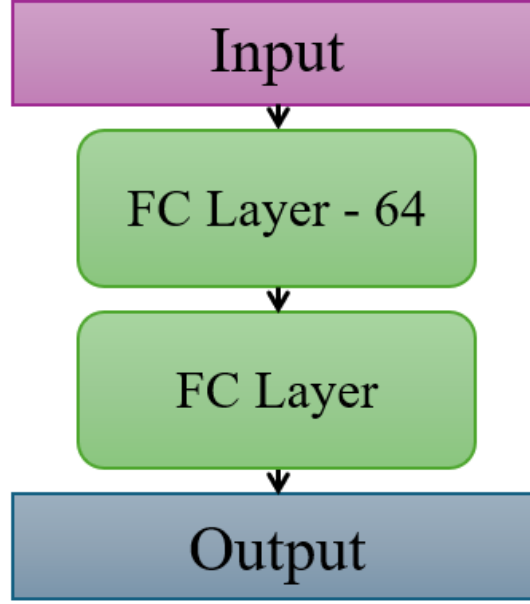


Figure 3.23. Flowchart of FC of 64 neurons.

LR	RMSE	MAPE (%)	Time (s)
1×10^{-1}	0.171866336	0.005986686	39.13
1×10^{-2}	0.014668353	0.000438582	107.95
1×10^{-3}	0.014106144	0.000426999	107.11
1×10^{-4}	0.013769051	0.000419875	47.35
1×10^{-5}	0.013650194	0.000414489	38.07
1×10^{-6}	0.014019074	0.000425121	35.05
1×10^{-7}	0.013701571	0.000416341	29.93
1×10^{-8}	0.013663169	0.000418076	30.12

Table 3.121. Learning Rate Testing

Batch Size	RMSE	MAPE (%)	Time (s)
16	0.013564154	0.000410177	947.84
32	0.013468881	0.000410025	262.33
64	0.013591314	0.000415961	117.94
128	0.013571255	0.00041283	61.38
256	0.01385236	0.000417733	45.61
512	0.013592106	0.000413668	39.74
768	0.013695895	0.000414259	29.76
1024	0.013650194	0.000414489	38.07

Table 3.122. Batch Size Testing

After performing more than 260 epochs with a LR of 1×10^{-5} , it was changed to 1×10^{-4} since there was no improvement.

The chosen batch size was 128.

In Figure 3.123, a comparison can be made between the tested models for virtual sensing.

Model	Test RMSE	Test MAPE (%)	Time (s)	Avg. Inf. Time (ms)
CNN x 1 (1st Current)	0.052142023	0.001466943	1180.27	0.0408
CNN x 1 (2nd Current)	0.127718148	0.004099074	1455.42	0.0368
CNN x 1 (3rd Current)	0.009614888	0.00029083	1116.82	0.0366
CNN x 1 (All Currents)	0.008979822	0.000279546	1835.62	0.0469
FC with 256, 128, 64 (1st Current)	0.014301125	0.000390966	659.63	0.0044
FC with 256, 128, 64 (2nd Current)	0.039147666	0.001078079	342.42	0.0033
FC with 256, 128, 64 (3rd Current)	0.011454653	0.000331311	551.9	0.0036
FC with 256, 128, 64 (All Currents)	0.007583554	0.000237417	1379.93	0.041
FC with 128, 64 (1st Current)	0.066103232	0.001732491	621.36	0.0016
FC with 128, 64 (2nd Current)	0.152520398	0.004585729	657.09	0.0017
FC with 128, 64 (3rd Current)	0.01360705	0.000397824	941.25	0.0015
FC with 128, 64 (All Currents)	0.012662531	0.000381386	1068.87	0.0016
FC with 64 (1st Current)	0.019177499	0.000523328	3194.87	0.0045
FC with 64 (2nd Current)	0.019294494	0.000518167	2481.67	0.0045
FC with 64 (3rd Current)	0.014966288	0.000431235	2331.77	0.0043
FC with 64 (All Currents)	0.01387692	0.000427972	2077.03	0.0047

Table 3.123. Comparison of models based on RMSE.

Chapter 4

Results

4.1 Condition Monitoring and Filtering Results

4.1.1 Sequence Length of 50 and Output of 1

(Convolution Layer) x 1

From Table 3.9, it can be seen that the best results were produced from minLSTMseq with an RMSE value of 0.0258. A close second was minGRUseq with an RMSE of 0.262. The training times were also faster than the other layers.

```
=====
Total params: 260,609
Trainable params: 260,609
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 3.24
=====
Input size (MB): 1.43
Forward/backward pass size (MB): 131.08
Params size (MB): 1.04
Estimated Total Size (MB): 133.56
=====
```

Figure 4.1. Complexity of minGRUseq with one convolutional layer.


```

=====
Total params: 343,041
Trainable params: 343,041
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 4.32
=====
Input size (MB): 1.43
Forward/backward pass size (MB): 131.08
Params size (MB): 1.37
Estimated Total Size (MB): 133.89
=====

```

Figure 4.2. Complexity of minLSTMseq with one convolutional layer.

In Figure 4.1 and Figure 4.2, only the complexity is shown since when printing the stack in Python, each cell is printed, and there is a cell for each sequence in the sequence length, meaning 50 separate cells.

(Convolution Layer) x 2

This layer stack proved to be especially effective with GRU, BiGRU, and BiLSTM being a full 0.01 less than with only 1 convolutional layer. Training times were comparable to other recurrent layer tests. Table 3.18 illustrates these values.

```

===== Training Model: GRU ===== | CNN Enabled: True
=====
Layer (type:depth-idx)      Output Shape      Param #
=====
RNNModel                    [512, 1]          --
├─CNN: 1-1                   [512, 128, 12]    --
│   └─Sequential: 2-1        [512, 128, 12]    --
│       └─Conv1d: 3-1         [512, 64, 50]      1,408
│           └─ReLU: 3-2       [512, 64, 50]      --
│               └─MaxPool1d: 3-3 [512, 64, 25]      --
│                   └─Conv1d: 3-4 [512, 128, 25]    24,704
│                       └─ReLU: 3-5 [512, 128, 25]    --
│                           └─MaxPool1d: 3-6 [512, 128, 12]    --
├─GRU: 1-2                   [512, 12, 256]    296,448
└─Linear: 1-3                [512, 1]          3,073
=====
Total params: 325,633
Trainable params: 325,633
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 2.18
=====
Input size (MB): 0.72
Forward/backward pass size (MB): 38.80
Params size (MB): 1.30
Estimated Total Size (MB): 40.82
=====

```

Figure 4.3. Complexity of GRU with two convolutional layers.

```

===== Training Model: LSTM ===== | CNN Enabled: True
=====
Layer (type:depth-idx)                Output Shape                Param #
=====
RNNModel                               [512, 1]                   --
├─CNN: 1-1                             [512, 128, 12]            --
│   └─Sequential: 2-1                 [512, 128, 12]            --
│       └─Conv1d: 3-1                  [512, 64, 50]             1,408
│           └─ReLU: 3-2                 [512, 64, 50]             --
│               └─MaxPool1d: 3-3         [512, 64, 25]             --
│                   └─Conv1d: 3-4         [512, 128, 25]            24,704
│                       └─ReLU: 3-5       [512, 128, 25]            --
│                           └─MaxPool1d: 3-6 [512, 128, 12]            --
├─LSTM: 1-2                             [512, 12, 256]            395,264
└─Linear: 1-3                           [512, 1]                  3,073
=====
Total params: 424,449
Trainable params: 424,449
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 2.78
=====
Input size (MB): 0.72
Forward/backward pass size (MB): 38.80
Params size (MB): 1.70
Estimated Total Size (MB): 41.22
=====

```

Figure 4.4. Complexity of LSTM with two convolutional layers.

(Convolution Layer) x 3

Table 3.27 illustrates BiGRU producing the lowest values but had one of the highest training times of this test.

```

===== Training Model: GRU ===== | CNN Enabled: True
=====
Layer (type:depth-idx)                Output Shape                Param #
=====
RNNModel                               [768, 1]                   --
├─CNN: 1-1                             [768, 128, 6]             --
│   └─Sequential: 2-1                 [768, 128, 6]             --
│       └─Conv1d: 3-1                  [768, 32, 50]             704
│           └─ReLU: 3-2                 [768, 32, 50]             --
│               └─MaxPool1d: 3-3         [768, 32, 25]             --
│                   └─Conv1d: 3-4         [768, 64, 25]             6,208
│                       └─ReLU: 3-5       [768, 64, 25]             --
│                           └─MaxPool1d: 3-6 [768, 64, 12]            --
│                               └─Conv1d: 3-7 [768, 128, 12]            24,704
│                                   └─ReLU: 3-8 [768, 128, 12]            --
│                                       └─MaxPool1d: 3-9 [768, 128, 6]            --
├─GRU: 1-2                             [768, 6, 256]             691,200
└─Linear: 1-3                           [768, 1]                  1,537
=====
Total params: 724,353
Trainable params: 724,353
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 3.56
=====
Input size (MB): 1.08
Forward/backward pass size (MB): 38.54
Params size (MB): 2.90
Estimated Total Size (MB): 42.51
=====

```

Figure 4.5. Complexity of GRU with two convolutional layers.

Recurrent Layer

With a simple vanilla layer, xLSTM can be seen from Table 3.36 to have produced the best result, but it also had the longest training time of all the layers in this test.

```
=====
Total params: 541,185
Trainable params: 541,185
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 12.07
=====
Input size (MB): 0.36
Forward/backward pass size (MB): 209.72
Params size (MB): 2.16
Estimated Total Size (MB): 212.24
=====
```

Figure 4.6. Complexity of xLSTM.

Recurrent Layer with Additional FC Layers of 128 and 64 Neurons

Table 3.45 shows xLSTM with an RMSE value of 0.0333, outperforming the other recurrent layers but having the second longest training time.

```
=====
Total params: 960,769
Trainable params: 960,769
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 1.95
=====
Input size (MB): 1.08
Forward/backward pass size (MB): 158.47
Params size (MB): 3.84
Estimated Total Size (MB): 163.39
=====
```

Figure 4.7. Complexity of xLSTM with only recurrent layer.

Recurrent Layer with Additional FC Layers of 256, 128, and 64 Neurons

Adding an additional FC layer of 256 neurons allowed the vanilla recurrent layer to produce the best results during this test with the fastest training times.

```

===== Training Model: RNN ===== | CNN Enabled: False
=====
Layer (type:depth-idx)              Output Shape              Param #
=====
RNNModel                             [256, 1]                  --
├─RNN: 1-1                           [256, 50, 256]           857,344
├─Sequential: 1-2                    [256, 1]                  --
│   └─Linear: 2-1                    [256, 256]               3,277,056
│       └─ReLU: 2-2                  [256, 256]               --
│           └─Linear: 2-3            [256, 128]               32,896
│               └─ReLU: 2-4          [256, 128]               --
│                   └─Linear: 2-5    [256, 64]                8,256
│                       └─ReLU: 2-6  [256, 64]               --
│                           └─Linear: 2-7 [256, 1]                65
=====
Total params: 4,175,617
Trainable params: 4,175,617
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 11.82
=====
Input size (MB): 0.36
Forward/backward pass size (MB): 27.13
Params size (MB): 16.70
Estimated Total Size (MB): 44.19
=====

```

Figure 4.8. Complexity of RNN with additional FC layers.

4.1.2 Sequence Length of 500 and Output of 50

(Convolution Layer) x 3

The following results produced similar graphs to all those displayed in sequence length of 50 and output of 1 section.

Table 3.63 illustrates that with a single convolutional layer and minLSTMseq produced the best RMSE value of 0.0232, but with an extremely long training time of 7097 seconds.

```

=====
Total params: 1,220,530
Trainable params: 1,220,530
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 1.61
=====
Input size (MB): 3.58
Forward/backward pass size (MB): 130.91
Params size (MB): 4.88
Estimated Total Size (MB): 139.38
=====

```

Figure 4.9. Complexity of minLSTMseq with three convolutional layers.

The filtering effect can be seen in Figure 4.10.

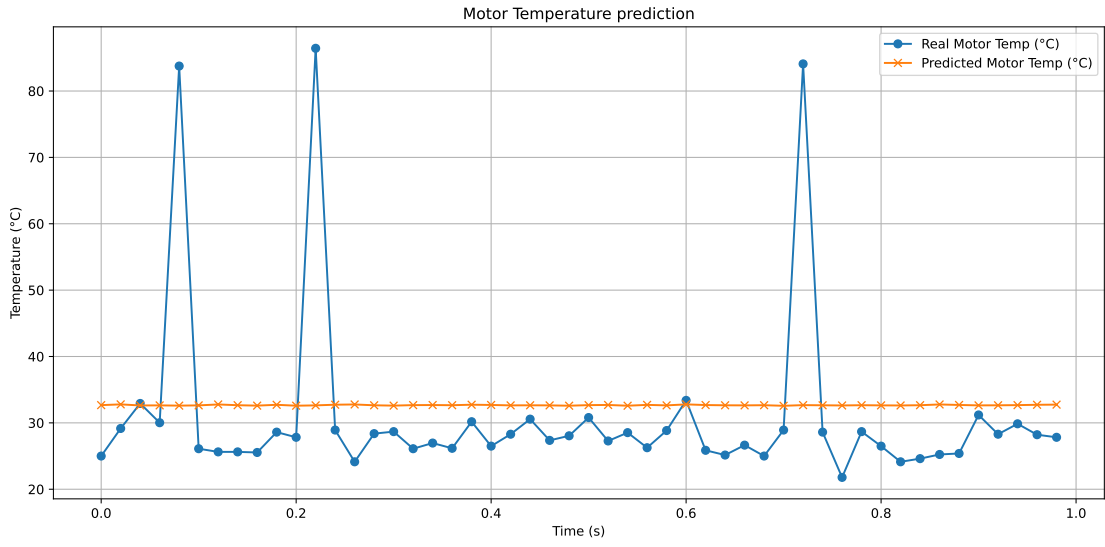


Figure 4.10. Predicted motor temperatures with three convolutional layers.

(Convolution Layer) x 2

Table 3.73 illustrates that with a single convolutional layer and minLSTMseq produced the best RMSE value of 0.033 with an extremely long training time of 14346 seconds.

```
=====
Total params: 2,021,426
Trainable params: 2,021,426
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 3.24
=====
Input size (MB): 3.58
Forward/backward pass size (MB): 196.71
Params size (MB): 8.09
Estimated Total Size (MB): 208.38
=====
```

Figure 4.11. Complexity of minLSTMseq with two convolutional layers.

(Convolution Layer) x 1

Table 3.83 illustrates that with a single convolutional layer and minLSTMseq produced the best RMSE value of 0.0428 and also with one of the shortest training times in this test.

```
=====
Total params: 3,531,186
Trainable params: 3,531,186
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 5.40
=====
Input size (MB): 3.58
Forward/backward pass size (MB): 196.71
Params size (MB): 14.12
Estimated Total Size (MB): 214.42
=====
```

Figure 4.12. Complexity of minLSTMseq with one convolutional layer.

4.1.3 Sequence Length of 250 and Output of 50

Recurrent Layer with Additional FC Layers of 256, 128, and 64 Neurons

With the addition of three FC layers, all of the recurrent layers produced RMSE values close to 0.167 with training times varying from 2933 to 6283 seconds.

Figure 4.13 illustrates the filtering effect that these tests produced.

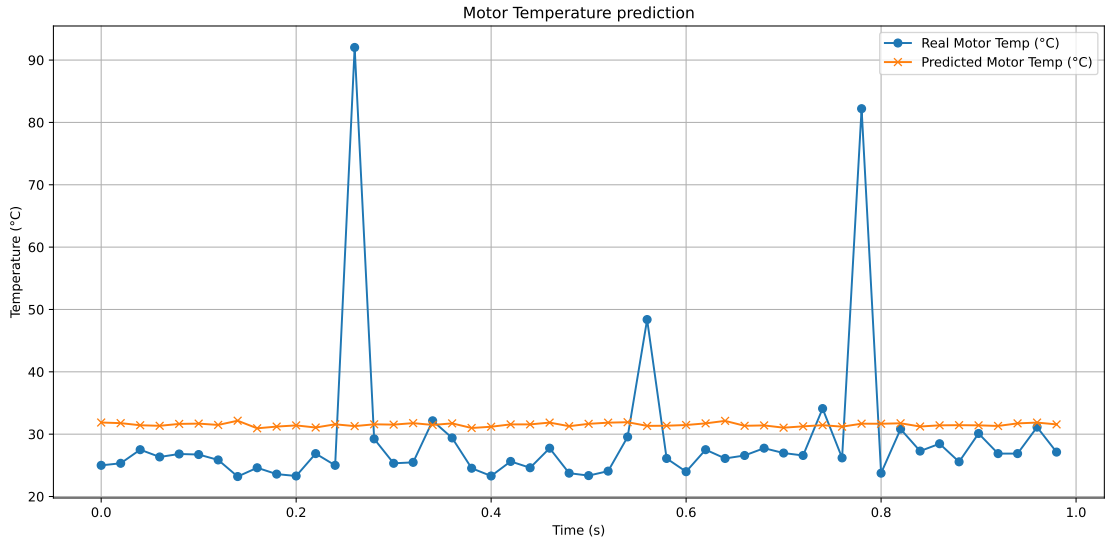


Figure 4.13. Predicted motor temperatures with sequence length of 250 and output of 50

Recurrent Layer with Additional FC Layers of 128 and 64 Neurons

minGRUseq produced the best RMSE value but also had one of the longest training times which can be seen from Table 3.103

```
=====
Total params: 8,407,154
Trainable params: 8,407,154
Non-trainable params: 0
Total mult-adds (Units.TERABYTES): 3.34
=====
Input size (MB): 1.79
Forward/backward pass size (MB): 131.57
Params size (MB): 33.63
Estimated Total Size (MB): 166.99
=====
```

Figure 4.14. Complexity of minGRUseq with additional FC layers.

Figure 4.15 illustrates the filtering effect that these tests produced.

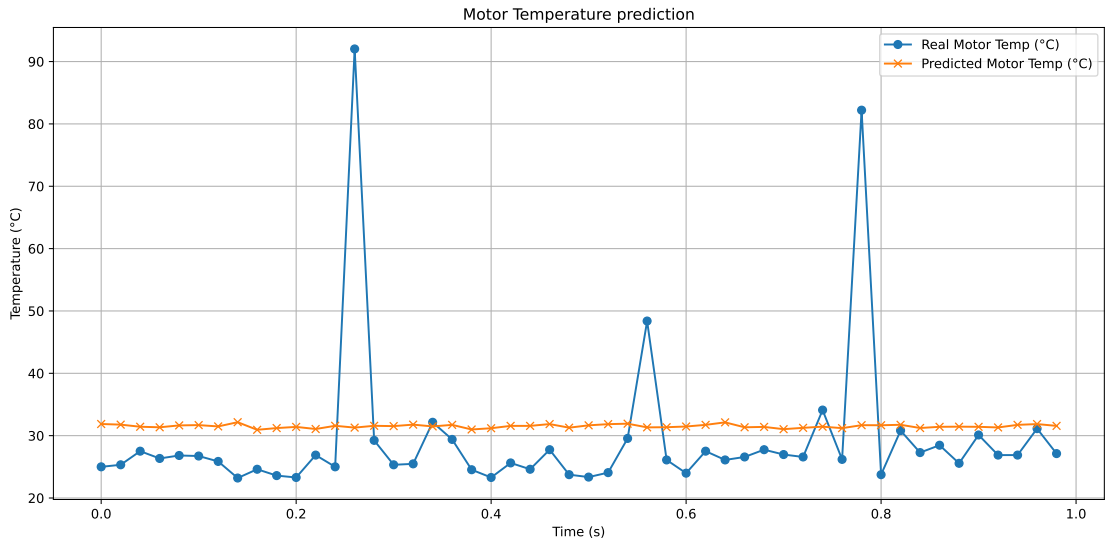


Figure 4.15. Predicted motor temperatures with sequence length of 250 and output of 50

Recurrent layer

Using just a recurrent layer without additional FC layers produced similar results to having the addition of three additional 256, 128, and 64 neuron layers, with only producing RMSE values of around 0.167.

4.2 Consideration of the Results Obtained

This section discusses the factors influencing the predictive performance of the proposed model. The limited accuracy observed in the tests can be attributed to the combined effects of nonlinear resistance behaviour, thermal-ventilation dynamics, and irregularities in the dataset.

The limited performance of the tests can be explained by the combined influence of nonlinear resistance behaviour, thermal-ventilation dynamics, and dataset irregularities.

First, the resistive contribution of the windings does not vary linearly with temperature. Most of the change in resistance occurs at higher values, near the zero-resistance region, which reduces the effectiveness of a linear approximation. This nonlinearity causes discrepancies between predicted and measured temperatures, particularly in the training phase of the NN since most of the data has little variation. The red line indicates the tendency of the temperatures to rise as the speed of the engine increases which is consistent with the physics involved.

In addition, the motor is subject to two competing thermal effects. In the initial operating phase (Zone 1) in Figure 4.16, the front portion of the rotor acts as a fan, generating airflow over the coils. This ventilation lowers the winding temperature and partially compensates for resistive heating. The cooling effect is not negligible and introduces variability into the system response. In the subsequent phase (Zone 2) in Figure 4.16, dissipative effects due to Joule heating become dominant. As current flows through the windings, resistive losses increase the temperature despite the ventilation effect. The superposition of these opposing mechanisms—forced convection cooling and resistive heating—results in a complex, nonlinear thermal response that is challenging to capture.

Finally, the experimental graphs also show visible spikes. These rises only occur when the resistance was set to $0\ \Omega$ (the max resistance), leading to much higher temperatures compared to the other tests. Such anomalies introduce discontinuities in the data, further complicating the model's ability to achieve consistent predictive accuracy.

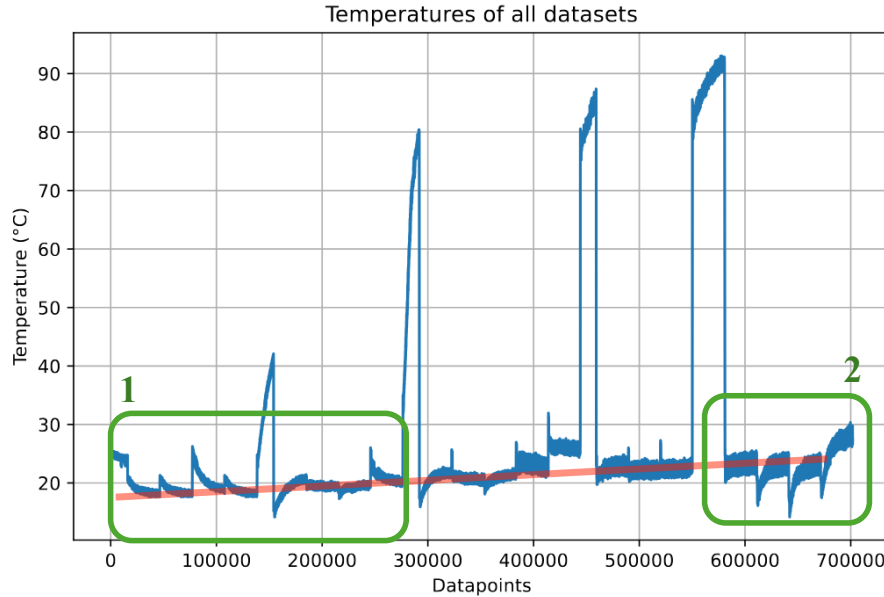


Figure 4.16. Combined temperatures of all datasets. The red line indicates the rise in temperatures as the speed increases. Zone 1 indicates the dataset in which ventilation effects are not negligible. Zone 2 indicates the datasets where the dissipative effects of Joule loss begin to increase.

4.3 Virtual Sensing Results

The first tests were done using only one convolutional layer with a sequence length of 50.

Other tests were performed using multiple convolutional layers but each proved to produce a filtering effect that can be seen in Figure 4.17.

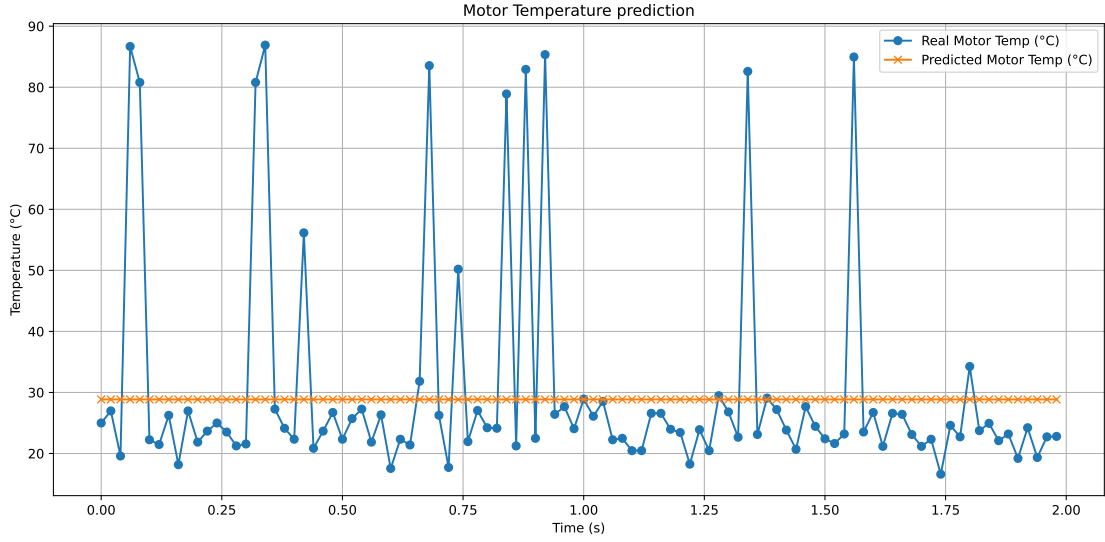


Figure 4.17. Flowchart of CNN with two convolutional layers.

Convolutional Layer x 1

In Figure 4.18 and Table 3.123, it can be seen that virtual sensing using only one convolutional layer is extremely promising producing values very close to real motor temperatures.

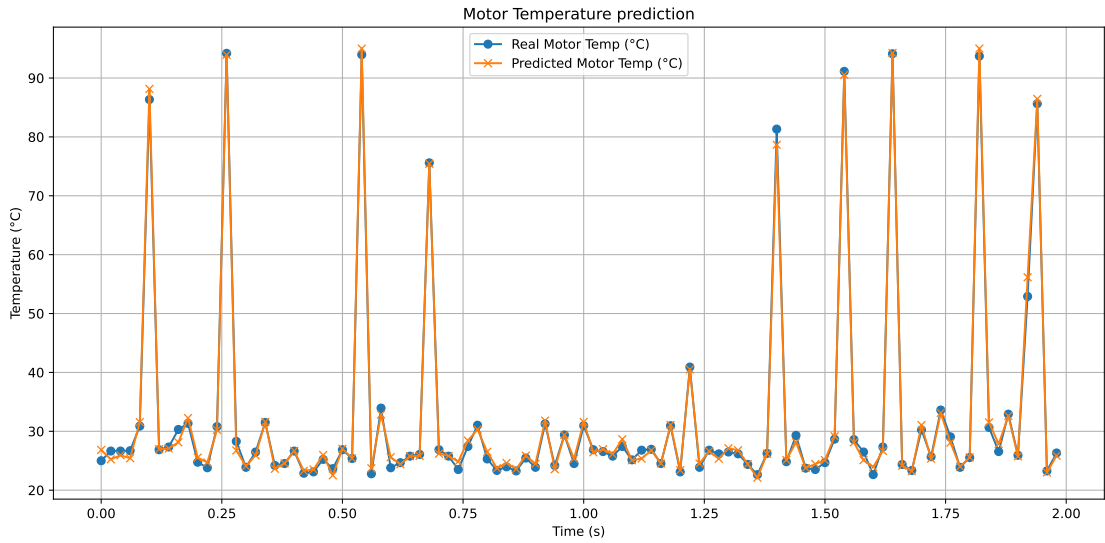


Figure 4.18. Virtual sensing using one convolutional layer.

FC layers with 256, 128 and 64 neurons

Figure 4.19 and Table 3.123 illustrate that virtual sensing using several FC layers yields very close to real motor temperatures. When testing FC layers with 256, 128 and 64 neuron layers, different tests were conducted using only one current as input instead of three as used for all the other tests. The final test performed with all three currents provided an RMSE of 0.01608 with a training time of 835 seconds, where as, when using only one produced an RMSE of 0.01467 with a training time of 394.9 seconds demonstrating that with less inputs, the system performed better and trained faster.

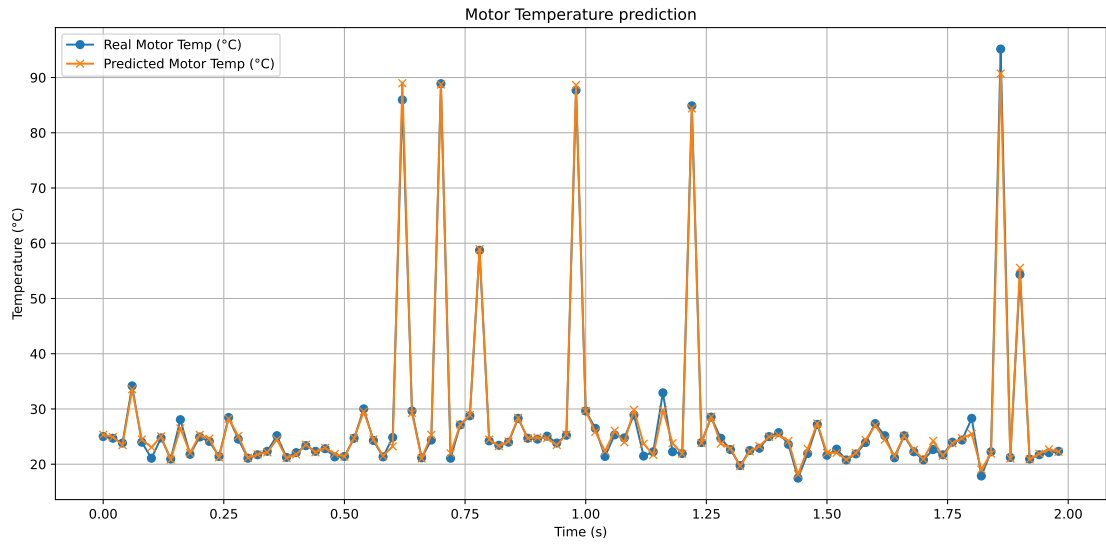


Figure 4.19. Virtual Sensing using FC layers with 256, 128 and 64 neurons.

FC layers with 128 and 64 neurons

In Figure 4.20 and Table 3.123, it can be seen that virtual sensing using several FC layers yields very close to real motor temperatures. When testing a FC with 128 and 64 neuron layers, different tests were conducted such as using only one current for input instead of three as used for all the other tests. The final test performed with all three currents provided an RMSE of 0.04606 with a training time of 2756 seconds, where as, when using only one produced an RMSE of 0.01614 with a training time of 727 seconds. Again showing that using only one current produces much better RMSE values and considerably faster training times.

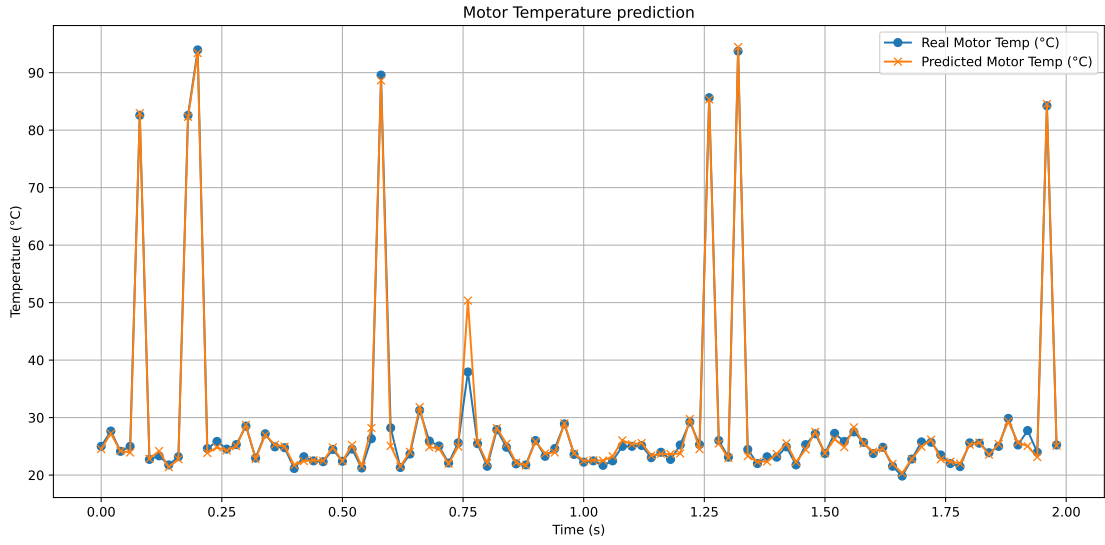


Figure 4.20. Virtual Sensing using FC layers with 128 and 64 neurons.

FC layer with 64 neurons

Figure 4.21 and Table 3.123 emphasize that virtual sensing using several FC layers yields very promising results. When testing a FC with only 64 neurons, different variations were tested such as using only one current as input instead of three as used for all the other tests. The final test performed with all three currents provided an RMSE of 0.0442 with a training time of 4725 seconds, where as, when using only one produced an RMSE of 0.01798 with a training time of 2323 seconds.

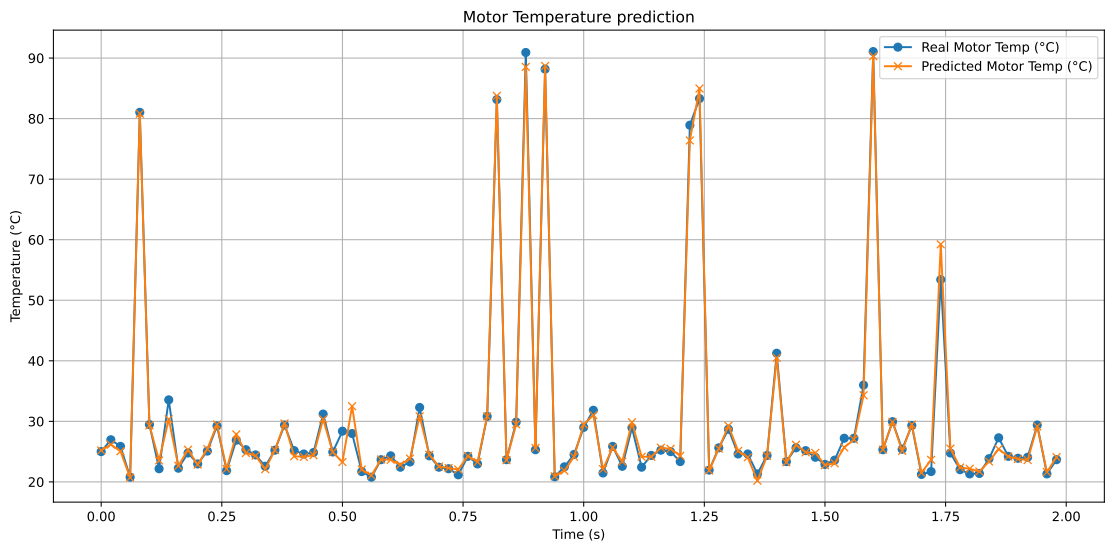


Figure 4.21. Virtual Sensing using FC layer with 64 neurons.

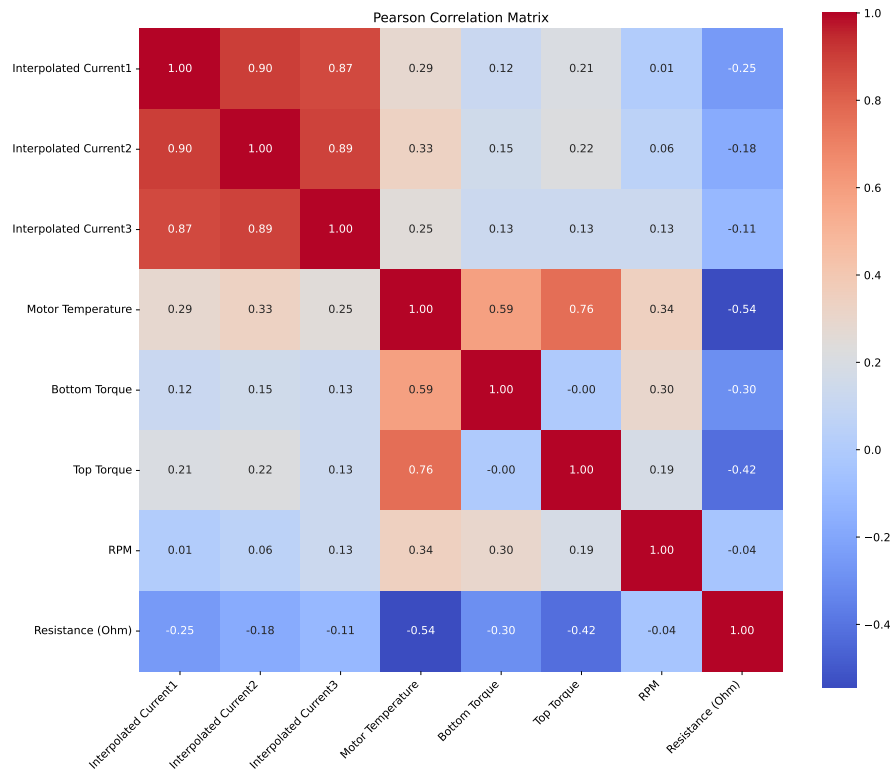


Figure 4.22. Pearson's Correlation Matrix

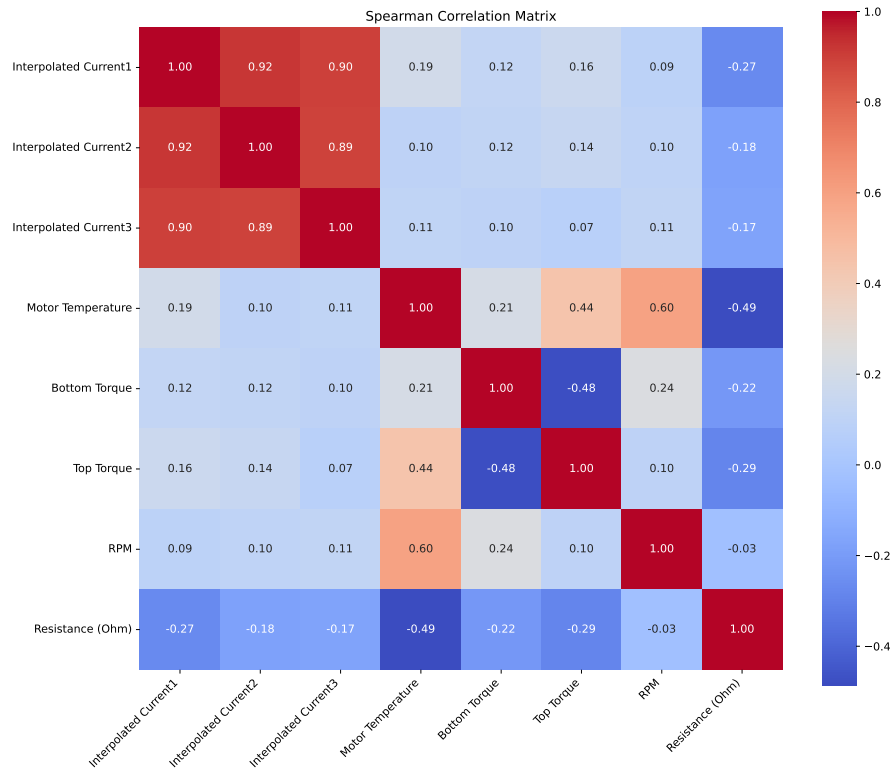


Figure 4.23. Spearman's Correlation Matrix

From Figure 4.22 and Figure 4.23, it is clear that all three currents are strongly correlated and because of this additional testing was done to analyze whether all three were needed. Table 3.123 illustrates the values obtained from performing tests on each individual current and all currents combined.

Table 3.123 shows the convolutional model (CNNx1) consistently achieved lower error metrics than the fully connected (FC) networks, particularly when all current signals were combined. Using all currents greatly improved predictive accuracy for every architecture compared to single-current inputs (e.g., CNNx1 RMSE decreased from 0.0521 for the 1st current to 0.0089 with all currents). Among FC models, the larger FC-256-128-64 configuration was the most competitive, reaching an RMSE of 0.0076 and MAPE of 0.00024% on combined currents, close to or slightly outperforming CNNx1 in MAPE but at similar training cost. Training times varied substantially, with CNNx1 requiring longer training than FC models, while inference times were lowest for the smaller FC networks (as low as 0.0015 ms) and highest for CNNx1 (0.04 ms). The 2nd current alone consistently yielded higher error values, possibly indicating a noisier sensor.

Chapter 5

Conclusions

This thesis investigated the feasibility of using NNs to predict motor temperatures in small-scale electric propulsion systems, aiming to reduce reliance on costly and fragile fiber optic sensors. A dedicated test bench with additional current sensors was employed to collect a comprehensive dataset suitable for machine learning applications, including motor currents, resistance, speeds, and reference temperatures.

The results demonstrated that lightweight recurrent architectures, particularly minLSTM and minGRU, consistently provided accurate temperature predictions while maintaining relatively efficient training times. Among the tested configurations, minLSTM with a sequence length of 500 achieved the lowest RMSE of 0.0232, accurately capturing motor heating dynamics in real time. Shorter sequences and shallower architectures offered faster convergence with only modest decreases in precision, highlighting the practical trade-off between accuracy and computational efficiency for potential onboard deployment.

In addition to recurrent networks with condition monitoring, this study explored virtual sensing using convolutional and FC layers. Even shallow networks with a single convolutional layer or modest FC architectures produced predictions neighbouring the real motor temperatures. Configurations with multiple FC layers (e.g., 256–128–64 neurons or 128–64 neurons) achieved low RMSE values comparable to the recurrent networks, demonstrating that lightweight architectures can provide accurate thermal monitoring. Experiments using only one current input instead of three decreased both RMSE values and training times, emphasizing the importance of comprehensive input features. Certain datasets with the highest resistance ($0\ \Omega$) produced temperature spikes, which the models captured with varying accuracy, reflecting the nonlinear thermal dynamics of the motor and highlighting the importance of careful data selection and pre-processing. These results confirm that virtual sensing is a viable approach for cost-effective and real-time temperature estimation.

The study further explored the effect of network complexity on prediction performance. Increasing the number of convolutional layers generally provided marginal improvements in RMSE but at the cost of longer training times. Similarly, adding additional FC layers sometimes improved accuracy, but these gains were inconsistent relative to the additional computational cost. These findings suggest that carefully designed lightweight architectures can achieve nearly optimal performance without excessive complexity, which is critical for UAVs and other resource-constrained applications.

Comparing the different recurrent layer types, minimal architectures such as minLSTM and minGRU consistently outperformed standard LSTM and GRU variants in balancing accuracy and efficiency. Bi-directional layers offered some improvements in prediction quality but incurred higher computational overhead. These results reinforce the potential of minimal recurrent architectures to replace physical temperature sensors in testing environments and possibly in real electric propulsion systems, reducing cost and improving durability.

In addition to recurrent networks, this thesis also investigated the use of lightweight models for virtual sensing using convolutional and FC layers trained directly on motor currents. Even shallow

networks with a single convolutional layer or modest FC architectures achieved low RMSE values comparable to the recurrent models. Combining all current signals consistently yielded the best performance (CNNx1 RMSE 0.0089, FC-256-128-64 RMSE 0.0076), while using individual currents increased error. These results confirm that convolutional and fully connected networks can provide accurate and fast temperature estimates from standard electrical measurements, further supporting the feasibility of replacing fragile fiber-optic sensors with low-cost machine-learning models.

Despite these promising outcomes, several limitations remain. Extended training times for long sequences may challenge real-time onboard deployment. The dataset was collected on small-scale motors under controlled conditions; additional testing on full-scale UAV motors and in varying environmental conditions is required to ensure generalizability. Furthermore, only electrical measurements were used as input features; incorporating environmental factors, such as ambient temperature or airflow, could further enhance prediction robustness. The observed temperature spikes in high-resistance ($0\ \Omega$) datasets also indicate that extreme operating conditions may challenge the models, necessitating careful handling of outlier data.

Future work should focus on optimizing sequence length and model depth for real-time embedded deployment, exploring hybrid architectures that combine CNN feature extraction with recurrent temporal modelling, and testing these models under dynamic operational conditions. Extending this approach to other components, such as batteries or actuators, could further broaden its applicability. Additionally, integrating these predictive models into intelligent monitoring systems for drones and Urban Air Mobility (UAM) concepts would enhance safety and operational reliability.

In conclusion, this thesis demonstrates that NNs, particularly lightweight RNNs and virtual sensing architectures, provide a reliable, efficient, and cost-effective alternative to fragile fiber optic sensors. By leveraging standard electrical measurements, these models enable safe, real-time thermal monitoring and open avenues for predictive maintenance and intelligent control in electric propulsion systems. Beyond UAVs, this approach has broader relevance in robotics, electric vehicles, and other domains where direct sensing is expensive or impractical, illustrating the transformative potential of machine learning for real-world monitoring applications.

Bibliography

- [1] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023, <https://D2L.ai>.
- [2] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, «A Novel Connectionist System for Unconstrained Handwriting Recognition», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. DOI: 10.1109/TPAMI.2009.23. [Online]. Available: <https://ieeexplore.ieee.org/document/4531750>.
- [3] O. V. I. Sutskever and Q. V. Le, «Sequence to Sequence Learning with Neural Networks», *arXiv preprint arXiv:1409.3215*, 2014. [Online]. Available: <https://arxiv.org/abs/1409.3215>.
- [4] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, «Learning to Diagnose with LSTM Recurrent Neural Networks», *arXiv preprint arXiv:1511.03677*, 2017. [Online]. Available: <https://arxiv.org/abs/1511.03677>.
- [5] GeeksForGeeks, *Activation Functions in Neural Networks*, <https://www.geeksforgeeks.org/machine-learning/activation-functions-neural-networks/>, Accessed: 2025-07-30, 2018.
- [6] Spectator, *There are 14 loss functions for regression tasks. You know several of them?*, <https://medium.com/@tpyrt1069098/there-are-14-loss-functions-for-regression-tasks-you-know-several-of-them-5ecd5e77a599>, Accessed: 2025-07-31, 2022.
- [7] GeeksForGeeks, *Stacked RNNs in NLP*, <https://www.geeksforgeeks.org/nlp/stacked-rnns-in-nlp/>, Accessed: 2025-08-15, 2025.
- [8] A. Kumar, *Sklearn Neural Network Example – MLPRegressor*, <https://vitalflux.com/sklearn-neural-network-regression-example-mlpregressor/>, Accessed: 2025-08-15, 2023.
- [9] M. Mishra, *The Learning Rate: A Hyperparameter That Matters*, <https://mohitmishra786687.medium.com/the-learning-rate-a-hyperparameter-that-matters-b2f3b68324ab>, Accessed: 2025-08-07, 2023.
- [10] T. Mucci, *Overfitting vs. underfitting: Finding the balance*, <https://www.ibm.com/think/topics/overfitting-vs-underfitting>, Accessed: 2025-08-15, 2024.
- [11] S. Hochreiter and J. Schmidhuber, «Long Short-Term Memory», *Neural Computation*, 1997. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/6795963>.
- [12] colah, *Understanding LSTM Networks*, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>, Accessed: 2025-08-15, 2015.
- [13] C. G. J. Chung and K. Cho, «Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling», *Neural and Evolutionary Computing*, 2014, ISSN: 0040-1625. DOI: <https://arxiv.org/pdf/1412.3555>. [Online]. Available: <https://arxiv.org/pdf/1412.3555>.
- [14] L. Feng, M. O. A. F. Tung, Y. Bengio, and H. Hajimirsadeghi, «Were RNNs All We Needed?», *arXiv.org*, 2024. [Online]. Available: <https://arxiv.org/abs/2410.01201>.
- [15] M. B. et al., «xLSTM: Extended Long Short-Term Memory», *arXiv.org*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.04517>.

- [16] H. Gholamalinezhad and H. Khosravi, «Pooling Methods in Deep Neural Networks, a Review», *Machine Learning*, DOI: <https://arxiv.org/pdf/2009.07485>. [Online]. Available: <https://arxiv.org/pdf/2009.07485>.
- [17] F. Amoretti, *Design and development of an experimental test bench for the study of small scale electric propulsion systems*, <https://webthesis.biblio.polito.it/34282/>, 2024.
- [18] Rototron, *Raspberry PI INA219 Tutorial*, <https://www.rototron.info/raspberry-pi-ina219-tutorial/>, Accessed: 2025-07-30, 2017.
- [19] M. Store. «Hall Effect Current Sensor Module for Arduino - 20A ACS712». Accessed: 2025-07-30. (2024), [Online]. Available: https://www.makerstore.com.au/product/electric-sensor-hall-ac712/?srsltid=AfmBOopD7-AXyB8nZBIgCqzAiR-ys5_19ygn2NPcq-UTDiHRTav8UXYM.
- [20] Smart Fibres. «SmartScan». Accessed: 2025-07-30. (2025), [Online]. Available: <https://www.smartfibres.com/products/smartsan>.