ACS6403 Industrial Training Programme
Lifecycle Management of Renewable Energy Storage System
ZT2_Final_Report

Contents

1.Project Information

1.1 Introduction

In the coming decades, there is an expectation for increasing energy use, which emphasises the urgent need for renewable energy sources that substantially reduce carbon emissions. Proton exchange membrane fuel cells (PEMFC), a type of hydrogen fuel cell with a high efficiency, quick response time, and clean energy source, have drawn attention. It can be used for a variety of purposes, including industrial use, portable electronics, and alternate ways to provide transportation power.[1] However, the durability of PEMFC is significantly affected by degradation. To address this issue, the present study collected two sets of data (static and quasi-dynamic) from laboratory experiments, and applied a moving average filter to reduce the noise in the datasets. A long-short term memory (LSTM) model was then utilised to predict the remaining useful life (RUL) and guide the health management of PEMFC lifespan. Furthermore, the degradation rate was calculated at two specific time points, which are 800 hours and 1000 hours of the PEMFC lifetime. The performance of the model and its results validity were assessed using various metrics, such as root mean square error, mean absolute percentage error, percentage error, and R-square values.

1.2 Project Background

The membrane electrode assembly (MEA) and bipolar plates are the two major components of a single PEMFC. Within fuel cells, bipolar plates are crucial for gas delivery (oxygen in the cathode and hydrogen in the anode), electrical current collection, mechanical support, and temperature regulation. The MEA has three distinct parts: the diffusion layer which is a porous segment of the electrode, the catalyst layer (the reaction zone), and the membrane itself. These sub-parts work together to enable the efficient transport of reactants, electrochemical reactions, and the production of electricity in fuel cells.[2]

The definition of degradations is Physical, chemical or other processes which lead or have led to failure. [3] The degradation of MEA and bipolar plates can be separated as mechanical degradation and chemical degradation. On the one hand, mechanical degradation is caused by non-uniform mechanical stresses. The physical dimensions of fuel cell components can experience changes depending on the temperature and relative humidity conditions. For instance, the components expand in hot temperatures and low relative humidity and constrict in low temperatures and high relative humidity conditions. This may cause uneven pressure between the anode and cathode then lead to mechanical degradation. Additionally, the stress difference may also occur at the contact surface and cause mechanical degradation. In practical applications, the PEMFC generally forms into stacks, which exacerbate the unbalanced mechanical stresses. On the other hand, the natural ageing of materials and external disturbances are the reasons for chemical degradation.

Specifically, materials ageing refers to the corrosion of bipolar plates, catalyst dissolution and polymer decomposition that happens in MEA.[4] Except for temperature and relative humidity conditions, the pH value and poisonous gas should also be considered as external disturbances. [5] In a single cell surface, the external conditions and reactants are not uniformly distributed. This heterogeneity can lead to varying current densities across different areas of the fuel cell system, hence resulting in different degradation speeds.

Prognostic and Health Management (PHM) systems can estimate the degradation status based on past operational profiles and current data. This enables condition-based maintenance (CBM) that minimises repair costs and extends the lifespan of the PEMFC.[6] In the PHM, Prognostic is the crucial process. According to ISO 13381-1(2015), the definition of prognostic involves estimating the time remaining until failure and the likelihood of one or more failure modes occurring current and future.[7] In other words, the prediction of remaining useful life (RUL) is the estimation of degradation status, hence monitoring the health of PEMFC. In this task, the RUL is defined as the corresponding time when the voltage drop reaches 5% minus 500 h.

In conclusion, the PEMFC stack is used for longer periods of time, and the degree of degradation tends to be more pronounced when there are varying or transient loads. The degradation results in a shorter overall lifespan for the fuel cell system. Therefore, monitoring and predicting the degradation is essential to improve the durability and the performance of the PEMFC.

1.3 Objectives & Aims

The aim of this project is to predict the RUL of the PEMFC, so that the lifespan and the performance of PEMFC could be improved. The objectives targeting this aim are to analyse the internal correlations between PEMFC indexes and select a feature as a health indicator. Then use it to predict the RUL with the data-driven model as accurately as possible.

1.4 Technical Requirements

1.4.1 Top-Level Requirements

Top-level requirements come from stakeholders, and stakeholders in this project include clean energy industry technology companies, which in turn interact, and are guided by, other stakeholders such as the manufacturers, customers, regulatory bodies and the government.

1.4.2 Functional Requirements

1. It is critical to analyse the static data provided in order to accurately predict voltage drops and calculate the RUL. The first functional requirement is to predict power drops due to cell degradation and estimate the RUL for the static operation mode based on a 3% voltage drop of initial voltage.

2. Similarly, estimating the RUL in the quasi-dynamic task requires a thorough understanding of the cell degradation process. The second functional requirement is to estimate RUL for the quasi-dynamic operation based on a 5% voltage drop of initial voltage.

3. The third functional requirement of the battery system is to utilise voltage as a health indicator to predict the value of health indicator at t=800h and t=1000h during both static and quasi-dynamic operation mode.

As a result, it is critical to approach the task methodically and analytically, using relevant data to create reliable systems that can accurately predict power drops and estimate RUL while also assessing health indicators.

1.4.3 Non-Functional Requirements

1. Performance: The system should be capable of analysing and processing large volumes of data within a reasonable time to ensure that the algorithm is trained on a diverse dataset. It should be designed to handle high volumes of data with minimum response times, to ensure that users can access results in a timely manner.

2. Accuracy: The algorithm should be highly accurate in predicting the RUL of specific systems to ensure that the system can be relied upon for making informed decisions.

3. Reliability: The system should be highly reliable to ensure that the algorithm produces consistent results across different datasets. It should be designed to handle an increasing amount of data as the number of PEMFC systems being monitored increases.

Appendix 1 contains a comprehensive matrix that defines all technical requirements in detail, including all relevant fields (ID; rationale; traced from; owner; verification method; verification lead; and verification level) for each requirement.

1.5 Literature Review

The prognostic can be divided into four steps based on the analysis chronological sequence, which are data processing, determination of health indicator and retirement condition, prediction model and RUL estimation, and results validation. Since the data is measured by sensors, the existence of disturbances and fault signals is ineluctable. Thus, data cleaning and processing are necessary before use. Different pre-processing techniques include loss filter, moving average filter, and Gaussian Kernel-based smoother. [4]

Additionally, the selection of appropriate Health Indicator (HI) for predicting the RUL of a system depends on the specific operating conditions, as efficient HI forms the fundamental basis of accurate RUL predictions. A typical HI under static and quasi-dynamic conditions is the stack voltage and power as summarised by Hua et.al.[8] These two HIs are easy to measure and exhibit a generally irreversible, monotonic decrease over time when the load current remains constant.[9] Prediction model selection is the main part of the prognostic. The strategy can be classified as model-driven, data-driven, and hybrid method. Model based methods involve development of empirical, semi-empirical or physical models and using statistics, filters, or machine learning for degradation prediction. Data driven models are based on machine learning algorithms and are one of the most widely used approaches.[4]

1.5.1 Model Driven Method

The prediction of RUL through model-driven approaches primarily depends on the fuel cell loading conditions, material properties, as well as degradation and failure mechanisms. It does not require large amounts of experimental data and can be adapted to different fuel cell models. Therefore, it is the most straightforward approach.[10] However, the model-based method has its complexity and difficulty in accurately establishing degradation models for PEMFCs. This is due to the fact that PEMFC degradation mechanisms operate on complex multi-time and multi-physical scales.[11] Jouin M has designed a particle filter framework to predict RUL. Three empirical voltage decay models have been tested: a linear model, an exponential model, and a log-linear model. Experimental results show that the log-linear model can predict RUL for 500 hours of fuel cell operation with 95% accuracy.[9]

Moreover, Zhang et al. proposed and tested a Multi-reservoir Echo State Network (ESN) with Mini Reservoir (MRM) degradation prediction model for PEMFC. The paper delves into the MRM structure, the use of the SG filter for noise reduction, the effect of the number of main reservoirs and main reservoir neurons on prediction accuracy, the effect of different training set lengths on model prediction accuracy, and compares the proposed MRM model to other models. The MRM structure, which includes a mini reservoir, was discovered to improve the model's prediction accuracy. Additionally, particle swarm optimization (PSO) was used to determine the optimal number of main reservoirs and main reservoir neurons, and it was found that for static conditions, the optimal number was 20,550 while for dynamic conditions, it was 10,800. Furthermore, the impact of different training set lengths on the model prediction accuracy was also explored, and it was found that the optimal prediction accuracy was achieved at 550 hours in dynamic conditions. The comparison with other models showed that the proposed MRM model had much better prediction accuracy. Therefore, the proposed MRM model is an effective option for degradation prediction of PEMFC.[12]

1.5.2 Data Driven Method

The technique that relies on monitored historical data to anticipate the remaining lifespan of PEMFC is referred to as the data-driven method. This approach is rapid and efficient since it does not require knowledge of fuel cell models or systems.[10]

In the field of predicting the remaining useful life (RUL) of machinery, Artificial Neural Networks (ANNs) are the commonly utilised data-driven approach. Y. Lei et al. summarised the previous research in this area, it has shown that Feed Forward Neural Networks (FFNNs) are employed in establishing the connection between health indicators and lifespan. ANNs can effectively capture complex non-linear relationships by training multi-layer networks, thus exhibiting strong potential in RUL prediction for intricate systems.[13]

Zheng et al. compared LSTM, traditional recurrent neural network (RNN), and backpropagation neural networks (BPNN). The results show that the root mean square error (RMSE) and mean absolute percentage error (MAPE) of the LSTM network are 0.0088 and 0.0101 respectively with 50% of the training dataset. The coefficient of determination (R^2) of LSTM is greater than 0.98. The result shows the comparison between LSTM, BPNN, and RNN. The prediction accuracy of the LSTM network is higher than that of the other two networks.[14]

Wang et. al proposed a stacked long-short term memory (LSTM) model combined with differential evolution (DE) to predict the RUL of the PEMFC reactor under stable current and dynamic current respectively. This method adds two dropout parameters to the LSTM model and stacks two LSTM models. Thus, stack LSTM is formed. One distinguishing characteristic of the S-LSTM model from other models is that its optimal hyperparameters are obtained through the differential evolution (DE) algorithm. The first step of the stack LSTM process is the feature selection. This method first selects several available features from the PEMFC system for prediction. The second step is utilising the stacked long-short term memory (S-LSTM) with two layers of LSTM to predict RUL, and the DE algorithm is used to confirm the optimal parameters of S-LSTM. The experimental results show that the final Score of particle filter (PF), LSTM, random forest (RF), and S-LSTM are 0.7765,0.7352, 0.8523, and 0.9633, respectively, at different percentages of the initial total voltage drop. The results show that the S-LSTM model is better than the other three methods in predicting the remaining life of PEMFC. [15]

Peng et. al proposed an LSTM model combined with convolutional neural networks (CNN). Finally, the prediction effect of CNN-LSTM and CNN-bidirectional (CNN-BiLSTM) is analysed and compared. Different from LSTM, CNN-LSTM uses a convolutional layer to extract different data features, and the output processed by the convolutional layer can better serve as the input of the LSTM layer. LSTM combined with CNN can not only extract the time characteristics of data but also extract both spatial and temporal features of data. The experimental results show that the RE values of CNN-LSTM and CNN-BiLSTM are 0.07% and 0.03%, respectively. Moreover, RMSE, MAPE, and other data show that CNN-LSTM and CNN-BiLSTM have high accuracy and good stability in predicting the RUL of PEMFC.[16]

Ilic et.al. brought linear regression into time series forecasting problems with the Explainable boosted linear regression (EBLR). EBLR is an iteration method that starts with the simple linear regression model, then the path with the biggest error is treated as a new variable and added to the base model at each iteration. This process enables the inclusion of non-linear features in the model. The experimental results demonstrate that EBLR is a simpler yet highly effective method compared to other algorithms. [17]

Support Vector Regression (SVR), which is a variant of Support Vector Machine (SVM), has become a widely used regression technique in state of health estimation systems. The primary goal of SVR is to discover a function that maintains flatness while deviating from observed response values by no more than a specified for each training point. The advantage of SVR is its ability to reduce large datasets into a smaller subspace known as support vectors, resulting in fewer mathematical operations. This computational efficiency is particularly advantageous when estimating the RUL. As summarised by M.A. Patil et al., the SVR is able to predict the exact RUL with 95% accuracy. [18]

In the case of forecasting using a Data-driven model, one limitation of the data-driven methods is that they rely heavily on historical data for training the models. This means that the accuracy of the predictions may be limited by the quality and quantity of available data. Another limitation is that the data-driven methods generally focus on predicting the RUL of PEMFC under static or quasi-dynamic operating conditions. In real-world settings, the operation condition can be highly different, and it is important to consider how these variations might affect the accuracy of RUL predictions.

1.5.3 Hybrid Method

A single hybrid model that combines or integrates multiple strategies. It can compensate for the shortcomings of a single approach and exploit the representation of different models to obtain the best performance.[10]

Renyou Xie et.al [19] suggests a fusion model to predict the RUL and short-term degradation of a PEMFC. The model is a combination of a particle filter (PF) and LSTM recurrent neural network. The particle filter derives the model parameter matrix in the training phase which is fed to the LSTM RNN model for training. The predicted values from the RNN are sent to the PF to get voltage for the next step. The above method is iterated until the threshold value is reached. Voltage is considered as the health indicator. The performance of the model was compared against a PF model. The fusion model was able to outperform the PF model in prediction of RUL and short-term degradation. However, the fusion model possesses a high uncertainty. The PF model showed better prediction results when 60 percent of data was used to train the model. The article concluded that the PF model works efficiently with less data whereas the fusion model requires lots of data to predict accurately.

Furthermore, Cheng Y et al. chose to use a least square vector machine (LSSVM) and a regularised example filter (RPF) approach to predict the PEMFC. The LSSVM was first used to indicate the PEMFC, and then the expected voltage was used as the basis for predicting the RUL through the RPF, where the RPF had better prediction performance than the PF. [20,21]

The performance of hybrid models in predicting PEMFC RUL is heavily dependent on the quality and quantity of training data, as well as the specific operating conditions of the fuel cell. Moreover, the practical implementation of these models should also take into account the associated costs and complexities. Therefore, when evaluating the efficacy of these models in predicting RUL, it's important to consider both their performance limitations and practical feasibility.

To summarise the prediction method, data-driven approaches have been discussed more frequently in recent years due to developments in computer science and the availability of large amounts of operational data.[22] They are gradually replacing model-based approaches in PHM research. The long short-term memory (LSTM) model compensates for the gradient disappearance, gradient explosion, and lack of long-term memory of RNNs. The floor cabinet neural network is capable of effectively utilising long-range sequence information. [10]

2.Technical Approach

2.1 Proposed System Design

The usage of machine learning (ML) techniques has increased dramatically with the rapid advancement of computer science in all fields. For PEMFC performance analysis, fault detection, and degradation prediction, some algorithms have been effectively used [23]. This section presents four of them, and all four of them will be used to predict battery RUL.

2.1.1 Linear Regression

Linear Regression (LR) is a statistical algorithm commonly used in data processing and analysis to estimate the relationship between two variables by fitting a linear function to a set of data points. This algorithm uses a linear equation to model the relationship between the dependent variable and one or more independent variables. A multiple linear regression model is formulated using the following equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon \qquad (1)$$

Where y is the predication value, $X_1,\ X_2 \ldots\ X_k$ is the input variables data, $\beta_1,\ \beta_2 \ldots\ \beta_k$ is the coefficient of each variable, β_0 is the intercept, and ε is the error term. Normally the error term is assumed to be 0 mean and constant variance. [24] One of the main advantages of linear regression is its simplicity and fast computational time. This feature makes it the first model to develop and provide a preliminary result.

Zhuo et.al.[25] proposed an output voltage prediction of PEMFC based on Linear regression.

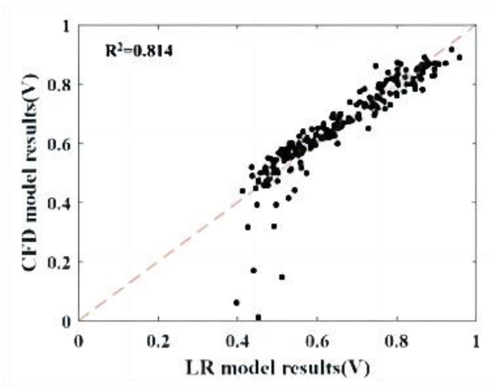


Figure 1. Linear Regression CFD Model Results [24]

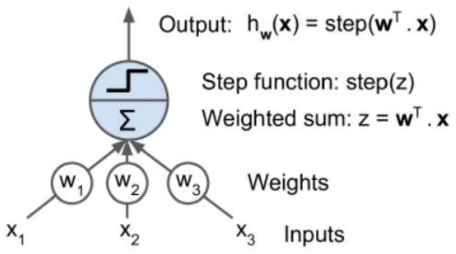
As shown in Figure 1, in the analysis of the prediction results of the output voltage. R-square is introduced to analyse the degree of fitting and accuracy of the model. As a result, LR has over 0.814 of R-square value. Contrary to this lower R^2, LR has a faster calculation speed and spent the least prediction time in the test to obtain the result.

In the research on the prediction of fuel cell stack voltage, Kheirandish et al. [26] also showed that LR has better prediction results, and the MSE of the prediction results is 0.05. However, linear regression techniques cannot automatically fit nonlinear variance but require explicit design to obtain more accurate results.

2.1.2 Artificial Neural Network

Artificial neural network is a type of feed forward neural network which is mainly used to deal with fixed-size input and output data. Typically, a neural network has four components; nodes, layers, activation function and weights. The nodes take up a weighted input and produce an outcome. The weights of the model are determined by the activation function. A stack of nodes makes up a layer. The layers of ANN contain interconnected nodes. Each node performs a simple computation on the inputs and passes the result on to the next layer. The input and output layers are always single layered. A simplest ANN perceptron structure is called threshold logic unit (TLU) or linear threshold unit (LTU). In this unit, each input is assigned a weight, then TLU applied a step function to the weight sum of the input and output the result as illustrated in Figure 2. The x are the features of the input dataset, the step function is also known as threshold active function. Heaviside step function is commonly used as shown in equation 2:

$$heaviside(z) = \begin{cases} 0 \ \ if \ z < 0 \\ 1 \ \ if \ z > 0 \end{cases} \qquad (2)$$



Output: $h_w(\mathbf{x}) = \text{step}(\mathbf{w}^T . \mathbf{x})$
Step function: step(z)
Weighted sum: $z = \mathbf{w}^T . \mathbf{x}$
w_1
w_2
w_3
Weights
x_1
x_2
x_3
Inputs

Figure 2: TLU structure [27]

A multi-layer ANN combines multiple TLUs to form the hidden layer so that each input goes into each TLU to predict the result. Additionally, a bias neuron is added to each layer except the output layer. Figure 3 shows the architecture of a multi-layer ANN.

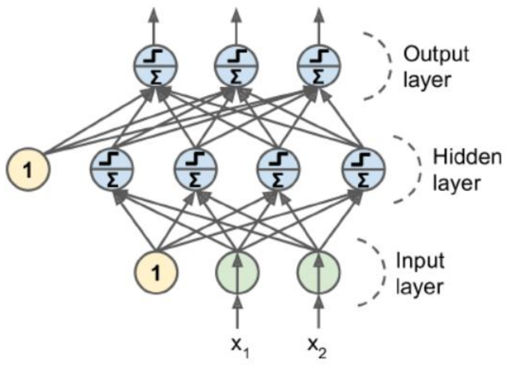


Figure 3: Multi-layer perceptron [27]

Once the outputs are predicted, the error is calculated by subtracting the actual value. Then the error is looped back to the model to generate weights of the nodes. [27] The idea of this process is to tune the weight to achieve the desired objective.

ANNs are very effective in handling fixed-size input and output data. However, one major limitation of ANNs is the challenge of capturing temporal dependencies in time series data. If the model is not well designed, it is difficult to discover the relationship between time and the input features. Hence result in large error and high computational time.

2.1.3 Support Vector Machine

SVM is a promising approach for estimating RUL since it is capable of processing multi-dimensional data and short training sets easily. A nonlinear relationship in a low-dimensional space can be mapped to a linear relationship in a high-dimensional space using SVM. In order to solve the regression problem, SVR is proposed. By minimising the $|\omega|$, SVR could be optimised using the following equation:

$$min \frac{1}{2} ||\omega||^2 + C(\sum_{i=1}^{l} \delta_i + \delta_i^* \qquad (3)$$

Taking the effect of prediction error into account, the relaxation factor δ_i is added to the optimization equation. C is the penalty factor. In nonlinear regression tasks, the kernel function is introduced to map data into high-dimensional space. [28] There are four commonly used kernel functions: linear function, polynomial function, radial based function, and two layers neural network function. [29]

Han et.al. built a SVR model to predict the output voltage of PEMFC. The result shows that SVR has good performance with a 0.98 R^2 value and a low root mean square error as shown in Figure 4. [30] This provides evidence of the capability of using the SVM model to predict the RUL of PEMFC.

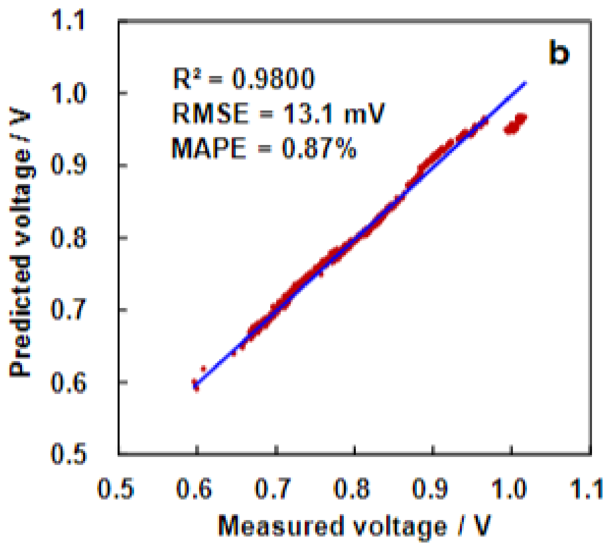


Figure 4: Prediction result of SVR [29]

The SVR model is a robust model to outliers and easy to update the decision model. However, it highly relies on the quality of input features and the choice of kernel function, which could be time consuming in feature engineering and tuning the parameters.

2.1.4 Long Short-Term Memory

The long short-term memory (LSTM) model is a special model based on a recurrent neural network (RNN). LSTM has the recursive characteristics of RNN, it also solves the problems of gradient disappearing and gradient explosion in the training process of RNN to a certain extent.

LSTM network consists of an input layer, output layer, and hidden layer. The most significant difference between LSTM and RNN is that the hidden layer of LSTM has unique memory capabilities. In a traditional RNN, it preserves all previous information, whether it is useful or not. The LSTM unit has memory ability, so it can choose to keep important information and discard useless information. It can not only reduce the burden of memory but also improve the accuracy of prediction. The LSTM cell contains three gates for adjusting the hidden state (h_t) and the cell state (C_t) respectively. These include the forget gate (f_t), the input

gate (i_t), and the output gate (o_t). They each have a different function. The input gate determines what new information will be stored into the cell state. The forget gate will decide which information needs to be discarded. The function of the output gate is to generate an output (hidden state) based on C_t.
x_t, $h_t$and C_t are the input, the hidden and the cell state of the current time step, respectively. x_{t-1}, h_{t-1} and C_{t-1} are the input of the previous time step, the hidden state of the previous time step and the cell state of the previous step, respectively. x_t, h_{t-1} and C_{t-1} are used as the input gates which process the input information and finally superimpose it with the memory cells of the forget gate to form a new cell state (C_t). The diagram and calculation formula among the variables is as follows: [30]

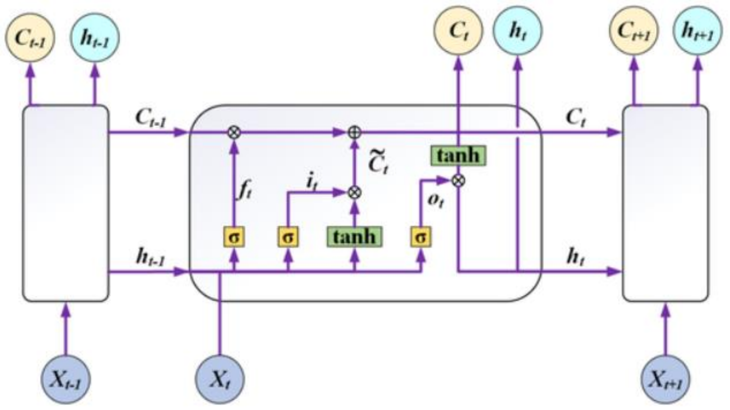


Figure 5: Basic LSTM Unit [30]

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (4)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (5)$$

$$\widetilde{C_t} = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (6)$$

$$C_t = f_tC_{t-1} + i_t\widetilde{C_t} \quad (7)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (8)$$

$$h_t = o_t tanh(C_t) \quad (9)$$

x_t, h_t, and C_t are connected input models, hidden layer output signals, and combined output vectors, respectively. W_{xi}, W_{xf}, W_{xc}, and W_{xo} are the weight matrix of x_t. W_{hi}, W_{hf}, W_{hc} and W_{ho} are the weight matrix of h_t. Tanh or sigmoid (σ) activation function are generally used. b_i, b_f, b_c, and b_o are the offset vectors.[30]

As previously mentioned, the LSTM approach addresses the issue of gradient disappearance in RNNs. However, its effectiveness heavily relies on the quality and quantity of training data, and it requires significant computational resources. Additionally, overfitting can be a concern when the LSTM model becomes too complex.

2.2 Data Collection and Preprocessing

2.2.1 Data Condensation

After reviewing the datasets, it was discovered that the static dataset contained 143,862 data points spanning from 0 to 1153 hours, while the quasi-dynamic dataset contained 65,536 data points from 0 to 504.8 hours. Due to the enormous size of these datasets, data point condensation was deemed necessary to preserve data primitivity and reduce computational time. To achieve this, data points were collected at intervals of 18 minutes for the static dataset and 6 minutes for the quasi-dynamic dataset. Since quasi-dynamic dataset only contains half of the time interval but more unstable tendency, more data points are collected with the purpose of training the model.

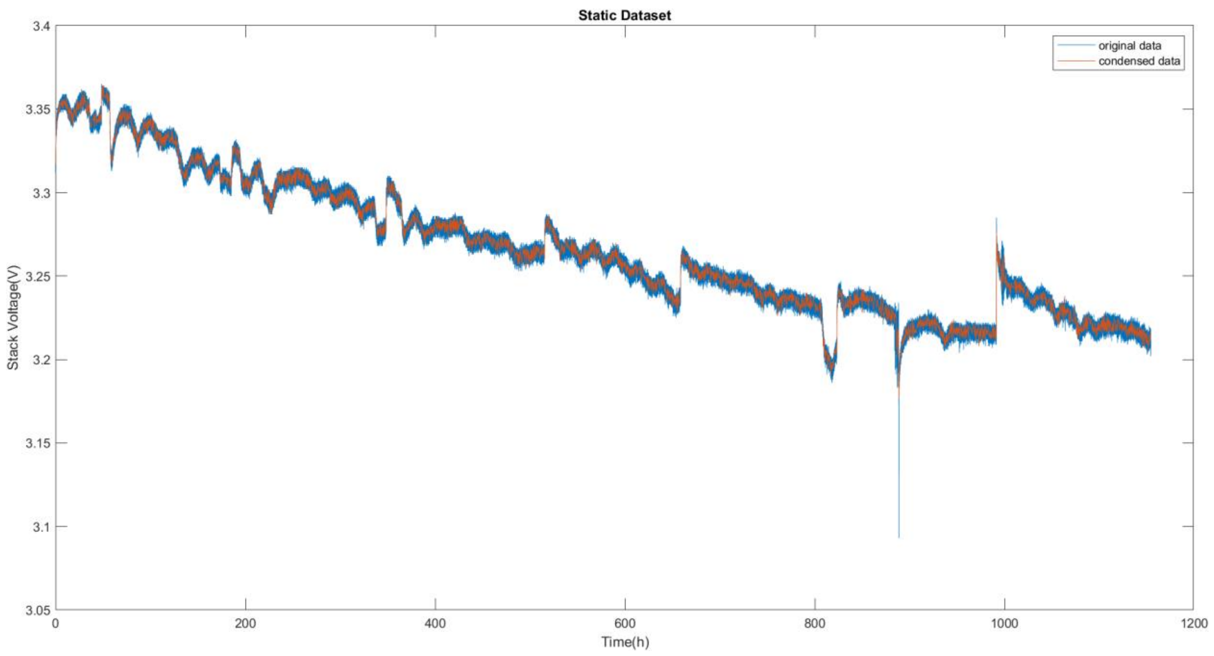


Figure 6: Data Condensation of static Datasets

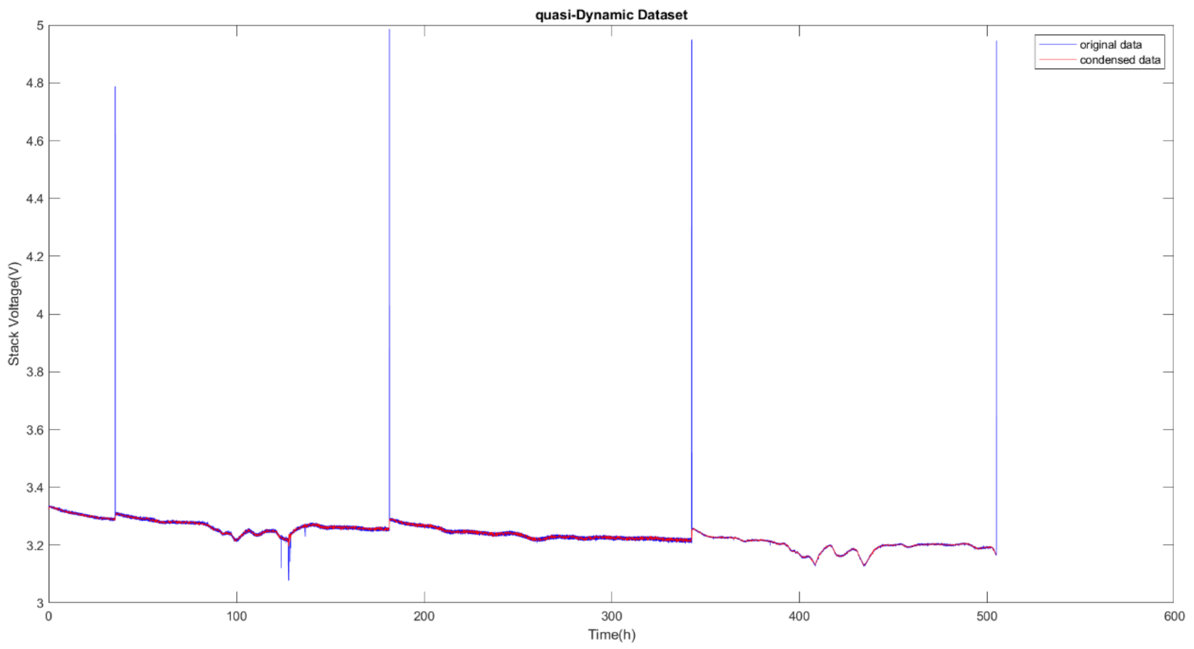


Figure 7: Data Condensation of Quasi-Dynamic Datasets

As illustrated in Figure 7, during the condensation process, outliers were also eliminated from the original data, particularly in the quasi-dynamic dataset. Four instances of extreme voltage peaks were identified in the quasi-dynamic dataset, which were attributed to abrupt changes in current. However, each peak was followed by a step response in the voltage. Therefore, the extreme peaks are treated as outliers and the step responses are chosen to represent the sudden change instead.

2.2.2 Moving Average Filter

A moving average filter (MAF) was employed to further reduce noise in the condensed datasets. MAF is a simple method to smooth an array of sampled data, making it an ideal filter for time-domain encoded signals. The idea of MAF is averaging the number of points from the input signal to produce a single output, as equation 10 explained:

$$y_i = \frac{1}{n}\sum_{j=0}^{n-1} x_{i+j} \quad (10)$$

Where x_i is the input signal, y_i is output signal, n is the number of averaging points. When n is even, one-sided averaging is performed over the range from 0 to n-1; whereas when n is odd, symmetrical averaging is carried out from -(n-1)/2 to (n-1)/2. [31] It is a commonly used technique for removing noise from time-series data while retaining the underlying trend. The number of averaging points, also known as window size, is a crucial parameter that affects the MAF's performance. Specifically, larger window sizes offer greater noise reduction but may sacrifice the response details. Conversely, smaller window sizes capture sharper responses but may incorporate more noise. [32] Therefore, selecting an optimal window size should be based on the sample characteristics and a careful consideration of the balance between noise and trend preservation. Z. Hua et al. utilised MAF in their own data pre-processing and were able to effectively remove peak and noise from both static and quasi-dynamic datasets. [33] The data after MAF is illustrated in Figure 8 & Figure 9.

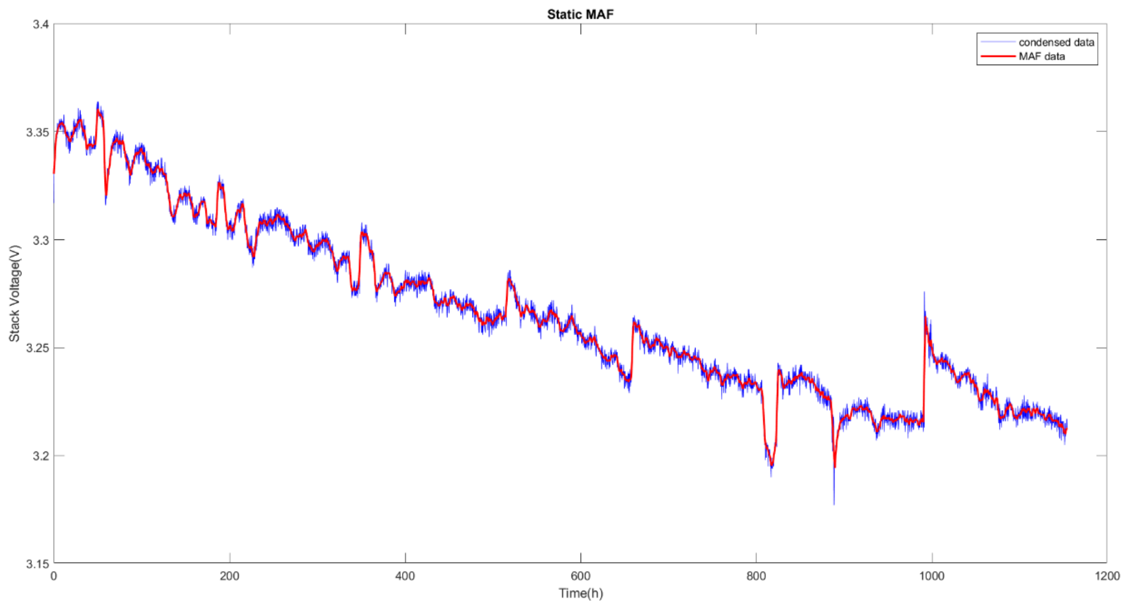


Figure 8: MAF of Static Datasets

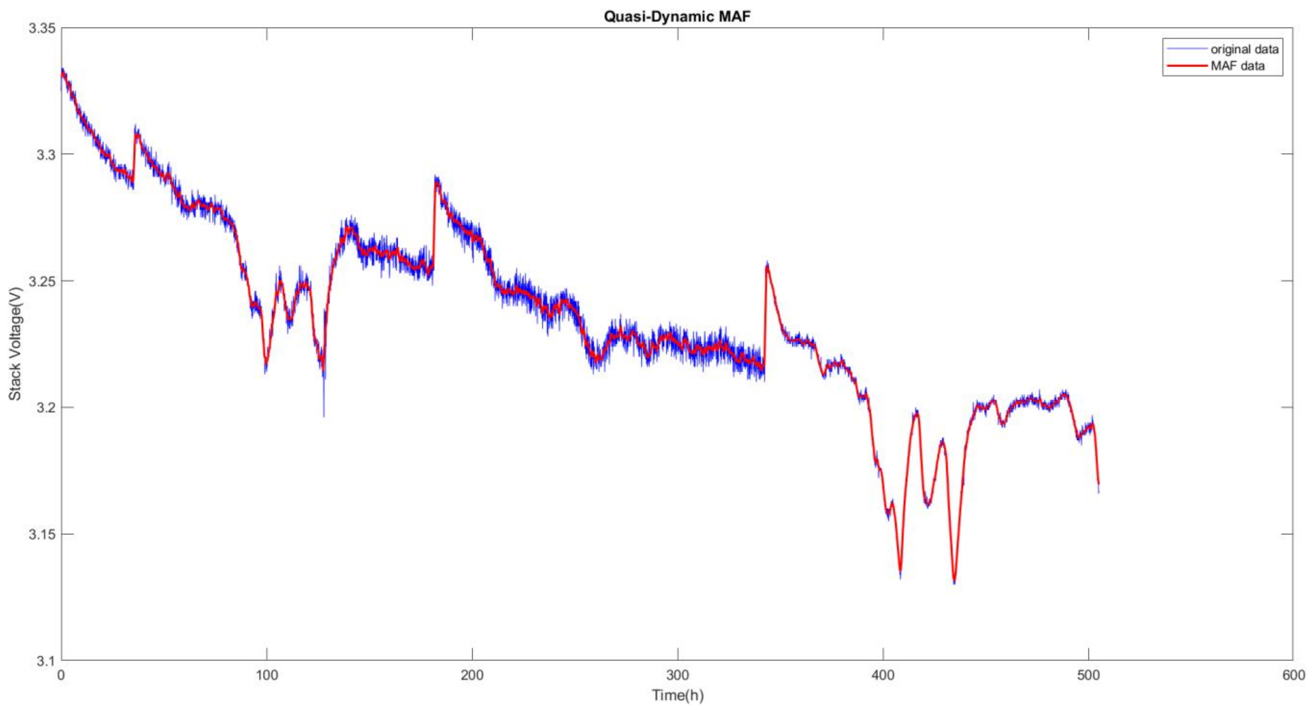


Figure 9: MAF of quasi-dynamic datasets

2.3 Feature Selection

2.3.1 Variable Meaning

There are 25 variables in static dataset and dynamic dataset. The meaning of each variable in the dataset can be find in Appendix1.

U1-U5 are the most relevant variables associated with Utot, and their definitions are**:** 1: The voltage across the anode side of the membrane electrode assembly (MEA), where the electrochemical oxidation of hydrogen fuel takes place in a fuel cell. 2: The voltage applied across the fuel cell's proton exchange membrane (PEM), which divides the anode and cathode compartments. 3: The voltage measured across the MEA's cathode side, where oxygen from the air is reduced electrochemically. 4: The voltage across the circuit or load that is linked to the fuel cell in order to draw current and produce power. 5: The bipolar plates, which are conductive plates that separate adjacent fuel cell units and allow for electrical connections between them, are used to measure the voltage [34].

A PEMFC system is made up of a stack that transforms chemical energy into electrical energy and is surrounded by several ancillaries that supply the system with reactants, collect power, or regulate working parameters [35]. The stack is an assembly of cells which are themselves made of different components. The total voltage (Utot) provided by the stack is the sum of the voltage provided by the cells that are connected in series(U1-U5).

2.3.2 Correlation Analysis

All data can be dimension-reduced based on an understanding of the significance of the variables by examining the correlation between the variables before being utilized as input for training a model. Dimension reduction results in a more representative data set, which might lower calculation requirements and result in less overfitting during model training.

The dimensions can be lowered by figuring out the relative relationship between the variables. An equation for the current density, for instance, has the variables J and I: J stands for current density, I for current, and A for surface area of the fuel cell. J = I/A. This equation relates current to current density, a measure of the amount of current per unit area of the fuel cell.

In studying the impact of variables on battery performance, Santarelli [34] found that increasing the fuel cell temperature can improve its behavior by increasing membrane conductivity and exchange current density. Additionally, increasing reactant operating pressure shifts the maximum power curve to higher current densities and enhances performance, particularly at high currents. However, rising humidity may cause cathode flooding and reduced efficiency when the temperature is low and the current density is high as opposed to having a favorable effect on fuel cell performance when the temperature is high.

PCA (Principal Components Analysis) method is usually used to reduce the dimensionality of data. PCA is principal component analysis, which is an unsupervised data dimensionality reduction method. The benefits of the PCA method are: finding hidden patterns in the data, reducing the dimensionality by reducing noise and redundancy in the data, and finding correlation variables.

In this project, Utot was selected as the indicator to reflect the decline in battery performance, so more important variables can be found by finding the correlation between other variables and Utot. After PCA processing on the static dataset, it was found that using 8 variables can present more than 80% of the data. After the correlation analysis of the variables, the variables with high correlation with Utot are selected as U3:0.997, Time:-0.96, DoutAIR:0.8, PoutAIR:-0.31, DWAT:0.23, HrAIRFC:-0.22, DoutH2:0.17, ToutWAT:0.102. Among them, a positive value represents a positive correlation with the change of Utot, and a negative value represents a negative correlation with the change of Utot. The larger absolute value of the value, the higher correlation. The above eight variables are input as model training data in static operation mode.

In the dynamic operation mode, 9 variables obtained by PCA can present 86% of the data of the quasi-dynamic Dataset. Time, TinH2, ToutH2, TinAIR, ToutAIR, DoutAIR, TinWAT, PoutAIR, I, HrAIRFC, these nine variables are selected as quasi-dynamic operation mode input. The correlations between these variables and Utot are U2:0.995, Time:-0.908, Tinh2:-0.648, TinAIR:-0.510, ToutH2:-0.436, TinWAT:0.279, PoutAIR:-0.227, I:-0.201, HrAIRFC:-0.147.

2.4 Prognostic framework based on LSTM

To develop the Long short term memory model, the KERAS API was used. A total of two LSTM layer were used with 190 and 120 nodes respectively. An LSTM model has three input parameters, the batch size, the time step, and the number of features. A dropout layer was introduced in between the first and the second layer. The second layer was connected to a dense layer with one neuron. The prediction of the voltage values based on other features can be categorized as a regression task. Hence, the “Adam” optimizer and the “MSE” loss function was found to be appropriate. The optimizer has a fixed learning rate of 0.001 and a L2 regularization value of 0.02. An early stopping function was added to stop the training process if the MSE value of the validation data remains constant for 5 iterations. The various parameters of the model are explained below in detail.

2.4.1 The LSTM Model

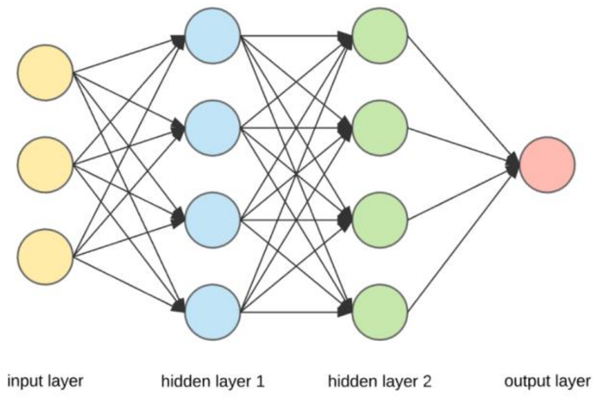


Figure 10: A basic neural network model [36]

The above figure represents a basic neural network model. The circles represent the nodes or the neurons of the network which usually contains the memory cell, input gate and the output gate. Nodes when stacked up together make a layer. Generally, a model consists of three basic layers, an input layer, a hidden layer, and an output layer. The size of each layer depends on the application of the neural network. The lines represent the connection of between two nodes and the relationship between two consecutive layers. Based on the type of information produced by a node, the strength of the connection is determined in terms of the weights.

To capture the complex relationship between the various features of the fuel cell and the voltage two LSTM layers are used. The first layer consists of 190 nodes and acts as the input layer which accepts 8 features and 10 as the time step. Hence, the model can learn the voltage trend for every 2 hours and make predictions based on them. The input layer passes a sequence to the next LSTM layer. This feature allows the model to learn the complex relationship of the various input features with voltage. The second layer of the model has 120 nodes and returns a sequence as output. The third layer is a dense layer. This layer has one node and is connected to the 120 nodes of the previous layer. The purpose of this node is to convert the sequence received from the previous layer into single values.

2.4.2 Hyperparameter Tuning

Overfitting occurs when the model performs well on the training dataset but poorly on unseen data. As a result, the model fails to generalize and provide reliable predictions on new set of data. There are several reasons for overfitting, such as complex model, insufficient training data and lack of regularization. To steer clear of overfitting, various regularizations have been added to the model.

A dropout layer randomly drops a fraction of the neurons in a layer, setting their output value to 0.

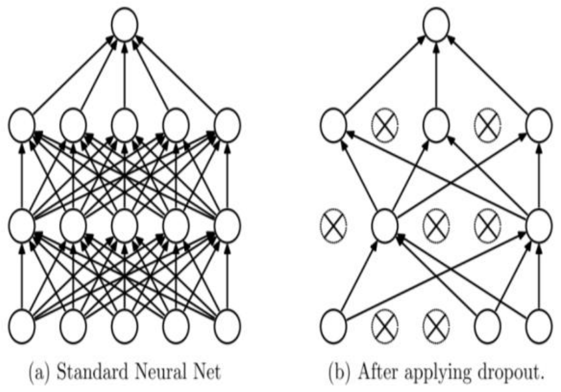


Figure 11: A basic neural network without (left) and with dropout(right) [37]

Figure 11(b) illustrates a neural network with dropout layers. The circle with the "X" mark indicate the neurons which are dropped during the training period. By dropping neurons, the model becomes flexible and less dependent on specific neurons for prediction. Hence, the model will be forced to learn complex relationships. Dropout rates usually range from 20 percent to 50 percent. A higher percentage can cause information loss, slow convergence, or inefficiencies.

To determine the optimal dropout rate, the performance of the LSTM model was evaluated with three values; 0.8, 0.2 and 0.1. The RMSE and the R^2 score was chosen as the evaluation matrix. A high dropout rate of 80 percent resulted in RMSE value of 0.074 and an R^2 score of 0.752 which indicates information loss during training and an inefficient model. Lowering the dropout rate to 0.2 resulted in an RMSE score of 0.075 and a R^2 score of 0.808. A 2 percents drop in the R^2 score was observed with a dropout value of 0.1 resulting in a RMSE score of 0.0752 and a R^2score of 0.781. Based on the above results, a value of 0.2 was found to be efficient.

Learning rate is a hyperparameter which determines the size of the step by which the model updates the weights during the training process. It controls the learning speed from the gradients computed during backpropagation. The figure below explains the relationship between learning rates, loss function and epoch.

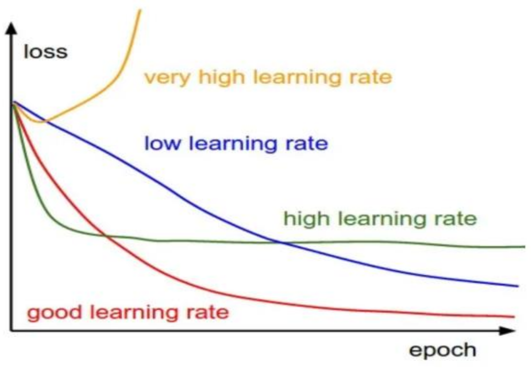


Figure 12: The relationship between learning rate, loss function and Epoch [38]

A high learning rate will incur very high loss values which will hinder the model from converging into meaningful predictions. On the other hand, a low learning rate take a very long time to train, hence convergence to a solution will take time.

To find the optimal learning rate, the RMSE and R^2 score of the model with different learning rate values determined and compared. The table below represent them:

Table 1: the RMSE and R^2 score of the model with different learning rate

Learning Rate	RMSE	R^2
0.1	0.19027	-3*10^{11}
0.01	0.07897	0.752
0.001	0.08643	0.791
0.0001	0.07553	0.808

The model was tested with 2 LSTM layers one with 190 and another with 120 nodes respectively. A dropout layer with 0.2 as the dropout value is introduced in between the two LSTM layers. The L2 regularization was valued at 0.02. The model was initially tested with a learning rate of 0.1 which received an RMSE score of 0.190 and an R^2 score of -3*10^{11}. To improve the performance of the model, the learning rate was decreased to a value of 0.01 and the scores were better with an RMSE of 0.0789 and an R^2 score of 0.751. The model was found to provide with best results when a learning rate 0.0001 was applied. The model scored an RMSE value of 0.0755 and an R^2 score of 0.808. Any rate beyond this value was found to be resulting in long training time. The Loss vs Epoch graph for the model is illustrated below:

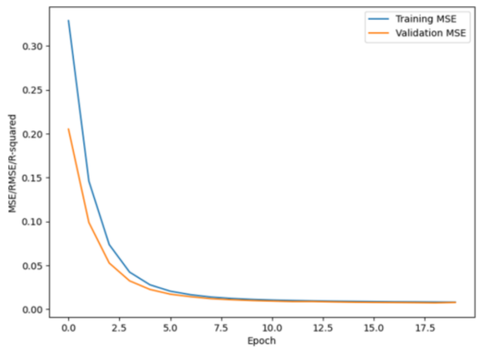


Figure 13: MSE Vs Epoch relationship

Ridge regularization, also termed as weight decay, is a common regularization technique which prevents overfitting by adding a penalty term to the loss function. This hyperparameter discourages large weights in the model. It is calculated by taking the sum of the squared values of all the weights in the network multiplied by a regularization parameter often denoted as λ.

To determine the optimal value of alpha, for L2 regularization, the RMSE and R^2 score of the model with varying alpha values was monitored. On using a value of 0.1 the RMSE and R^2 value were found to be 0.099 and 0.776. To improve the performance of the model, the alpha value was increased to a value of 0.01. RMSE and the R^2value for the corresponding alpha value was found to be 0.786 and 0.084. A value of 0.02 produced an RMSE value of 0.076 and a R^2 score of 0.808. Hence, an alpha value of 0.02 was found to be appropriate.

2.4.3 The ADAM Optimizer

Optimizers are methods used to tune the parameters of a model during training phase in order to minimize the loss function and improve the model's performance. During the model's training, the optimizer finds the optimal weight and biases in order to minimize the difference between the predicted values and the actual values.

The LSTM model uses an ADAM optimizer and MSE as the loss function. To update the parameters, the optimizer initializes a variable for each parameter. The gradient of the loss function with respect to the parameters is calculated during every training iteration. A moving average of the gradient is recoded over time. The learning rate for each parameter is adjusted based on the magnitude of the gradient. A large gradient will have smaller learning rates and smaller learning rates has larger learning rates. However, the LSTM model was found to be producing high scores with a fixed learning rate. After the learning rates are updated, the parameters are accordingly changed in the direction of negative gradient [39].

3.Result Analysis

3.1 Configuration of Computer

The computer used in this study is equipped with a 12th Gen Intel(R) Core (TM) i7-12700H 2.30 GHz processor and has 16GB of RAM. The Python programming language was utilized to develop and train the LSTM model. Additionally, the system was equipped with a NVIDIA GPU to accelerate the computations involved in training the LSTM model.

3.2 Evaluation Metrics

In order to evaluate the prediction performance of the LSTM model, the selected evaluation metrics are used 1) Mean Squared Error (MSE): Measures the average of squared deviations between predicted and observed values, where lower value indicates better performance. 2) Root Mean Square Error (RMSE): Provides measure of the overall model prediction accuracy, where lower value indicates better performance. 3) Mean Absolute Percent Error (MAPE): Computes the magnitude of the errors relative to the actual values. 4)Mean Absolute Error (MAE): It determines the average magnitude of the variations between predicted and actual values, irrespective of their directions. It is computed as the mean of the absolute differences between predicted and actual values across all instances in the test set, with equal weight assigned to each difference.[40] 5) Determination coefficient (R^2): Determines how well the model fits the data. R^2 represents the goodness-of-fit of a regression model. It measures how well the independent variables in the model account for the variability observed in the dependent variable. The value ranges from 0 to 1 where 1 indicates a perfect fit.[41]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (11)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \quad (12)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| * 100 \quad (13)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \qquad (14)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \qquad (15)$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of data points. $\bar{y}_i$ is the average value of the output values observed across a given set of samples. Another important metric which can be considered for the evaluation of the model is the Percentage Error in Failure Time ($\%Er_{FT}$). It measures the percentage difference between the predicted failure (t_{RUL}^{pre}) time and the actual failure time (t_{RUL}^{act}).

$$\%Er_{FT} = \frac{t_{RUL}^{act} - t_{RUL}^{pre}}{t_{RUL}^{act}} * 100 \quad (16)$$

The RMSE is a readily comprehensible measure that denotes the mean size of the discrepancies in the identical units as the dependent variable, thereby making it instinctive to comprehend and contrast across diverse models or datasets. The utilization of the square differences approach guarantees that the summation of positive and negative errors does not result in cancellation during the averaging process, thereby yielding a more comprehensive evaluation of the overall precision of predictions. [42]

The MAPE offers a uniform metric for evaluating precision that is applicable to various magnitudes and dimensions. This feature makes it advantageous in evaluating the efficacy of models across datasets that exhibit varying scales or degrees of magnitude. The relative performance of a model is of particular use, as opposed to its absolute accuracy. This facilitates a more comprehensive evaluation of the consequences of errors. [43]

R^2 is a standardized metric that ranges between 0 and 1, making it easy to interpret. A higher R^2 value indicates a better fit of the model to the data. [44]

The Percentage Error of Failure Time serves as a precise indicator of the predictive capacity of a given model with respect to estimating the remaining useful life (RUL) of a given system. The prognostics and health management (PHM) field considers RUL prediction to be a critical task as it aids in averting unforeseen system failures by furnishing an approximation of the residual life of a system. The quantification of the precision of the LSTM model's Remaining Useful Life (RUL) prediction is a crucial aspect in guaranteeing the dependability and security of the system.

3.3 Static Operation Model

3.3.1 LSTM Prediction Result Analysis

The experimental data used for the static task consisted of 1153 hours of continuous testing on the PEMFC. For training our model, we used data ranging from 0 to 500 hours, while data from 500 to 1153 hours were employed for prediction. Figure 14 illustrates the comparison between the stack voltage prediction results obtained from the LSTM and the real stack voltage data.

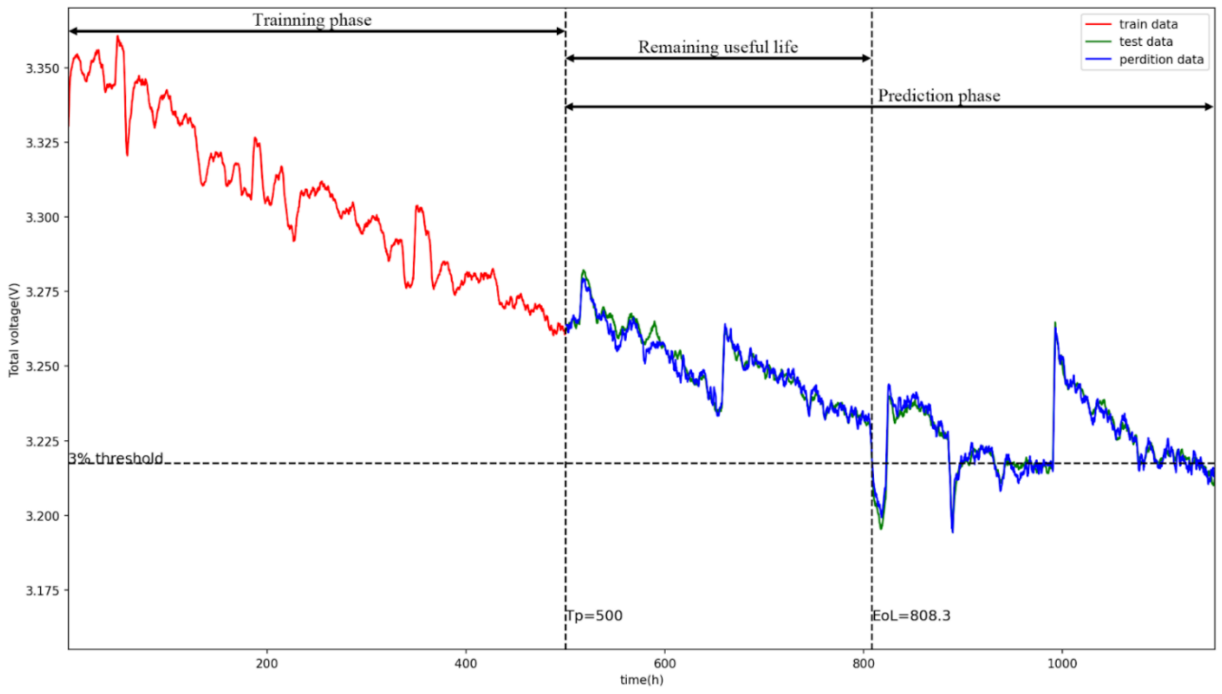


Figure 14: Prediction results of LSTM RNN for static operation mode at TP=500

Table 2: Static LSTM prediction results

Algorithm	RMSE	MAPE(%)	R^2	ErFt(%)
LSTM	0.00248	0.05896	0.98223	0.09721

Based on Table 2, it is evident that the RMSE of the model's predicted results in the static data set is 0.00248. This value represents a significantly smaller magnitude compared to the overall variation of 0.16 observed throughout the entire experimental process of Utot. Thus, this outcome serves as compelling evidence to support the high prediction accuracy of our model.

R^2 equals to=0.98 indicating that approximately 98% of the variance in the voltage predictions from our model can be explained by the independent variables or features used in the model. This high R-squared value suggests that our model is performing exceptionally well in capturing and predicting the patterns and trends in the voltage data. It indicates a strong relationship between the input features and the target variable (stack voltage) within our model. Overall, an R-squared value of 0.98 is considered very high and indicates a robust and accurate prediction model.

As this study focuses on the RUL of PEMFC, the predictions can be evaluated by %Erft. As a result, the error between the predicted RUL and the actual RUL is 0.097219%. This is an underestimation result (positive %Erft) which shows the predicted RUL is smaller than the actual RUL. Underestimation would allow us to detect failure time in advance but with a small time difference. A low %Erft value could provide evidence of it. A large %Erft the value indicates the detected failure time predicted by the model is much sooner than the real RUL. This would lead to energy wasted and misjudging the maintenance time. Conversely, overestimation is a more dangerous situation. It will delay the fault alarm which may lead to the potential safety issue.

3.3.2 Comparison of Prediction Results of Different Algorithm

Table3 shows the results of LSTM, ANN, Linear Regression, and SVM for the PEMFC stack of voltage degradation prediction. The four criteria are used for evaluating the predictions: MSE, RMSE, MAE, and R^2.

Table 3: Comparison results of different models

Model	MSE	RMSE	MAE	R^2
LSTM	$6*10^{-6}$	0.002	0.002	0.982
Neural Network	0.004	0.060	0.044	0.942
Linear Regression	0.010	0.100	0.083	0.842
SVM	0.010	0.100	0.088	0.841

Table 3 clearly shows that the linear regression and SVM models give less satisfied results compared with the LSTM model. The analysis of MSE, RMSE, MAE and R^2 proves that the linear regression and SVM models exhibit a relatively in their predictions. This is because linear regression and SVM are more suitable for analysing linear data. [45] Since the relationship between input feature data (except U3) and predicted total voltage is not linear, linear regression and SVM have difficulties learning the relationship between the input and the output. Hence, this results in less accurate predictions. However, the results are acceptable for this complex dataset. The reason is that feature U3, which has a strong linear relationship with the total voltage, is involved in the input. ANN is the relatively better performer of the three models used for comparison because it can analyse and learn from non-linear data, so the results of ANN's R^2 and LSTM models are very close. However, the error in the prediction results given by the LSTM model is still much smaller than the results obtained by the ANN model. This is because the LSTM introduces storage units and gating mechanisms into traditional neural networks targeting time series problems. This makes LSTM more suitable to process time series data and capture long-term dependencies between variables.

In summary, the LSTM model effectively predicts Stack voltage of PEMFC under static dataset and the result suggests that the LSTM model outperforms the other models in terms of accuracy and precision.

3.4 Quasi-Dynamic Operation Model

3.4.1 LSTM Prediction Result Analysis

Figure 15 presents the results of comparison between stack voltage prediction value and Real value in the 505h training dataset under quasi-dynamic operating environment.

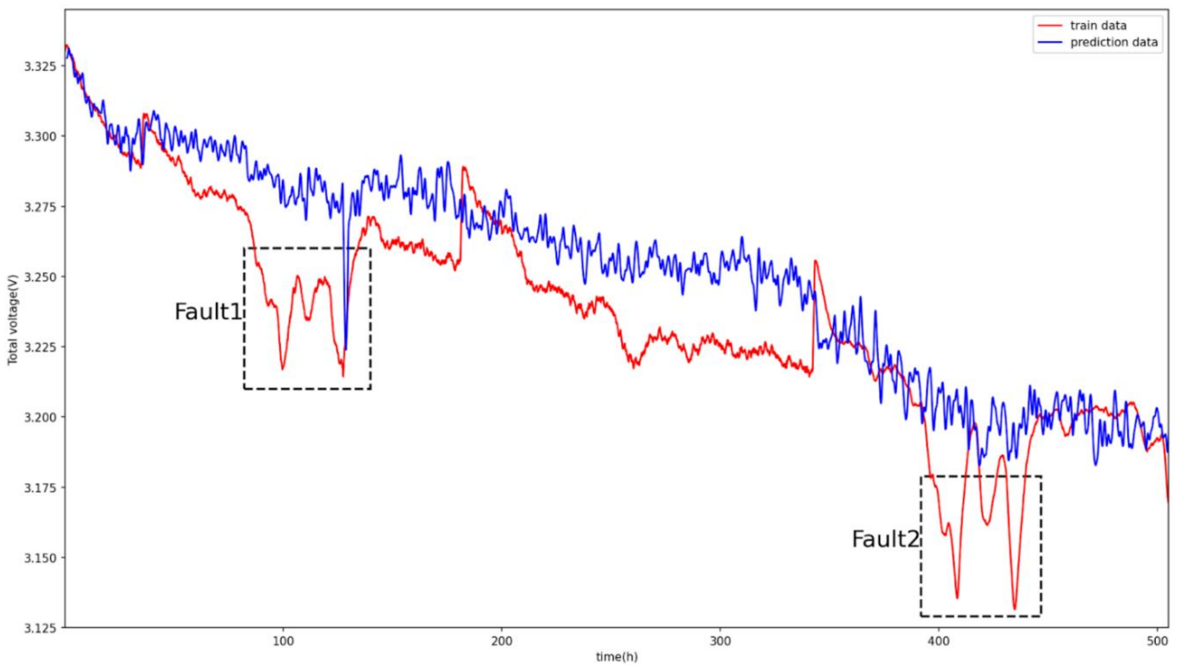


Figure 15: Prediction results of the training set in quasi-dynamic data

Table 4: Evaluation results of dynamic prediction models in the training dataset

Model	RMSE	MAPE(%)	R^2	MAE
LSTM	0.07553	1.00562	0.80812	0.01866

The results show that in the quasi-dynamic task, the LSTM model does not perform as well as it does in the static dataset. In the evaluation criteria, RMSE and MAE are used equally to evaluate the model's performance in predicting errors., MAE is a metric commonly used to measure the average magnitude of errors in a prediction model. Unlike RMSE which may emphasize larger errors more strongly, MAE treats all errors equally, considering their absolute values without considering their direction (positive or negative). Therefore, the RMSE (0.07) of the model prediction results is more significant than the MAE (0.02), which indicates that the LSTM model cannot predict abnormal voltage fluctuations (such as Fault1 and Fault2) well in the prediction results of the quasi-dynamic data set.

In this prediction result, RMSE equals 0.0753 is a larger prediction error than the range between Utotmax and Utotmin (0.2) after MAF. This is because compared with the static dataset, our model in quasi-dynamic prediction lacks U3 which has a higher correlation with the total voltage as an input, and there are outliers such as Fault1 and Fault2, resulting in a higher value of RMSE.

In the research on R^2, R^2=0.8 indicates that approximately 80% of the variance in the voltage predictions from our model can be explained by the independent variables or features used in the model. This is considered a good prediction result.

Combining RMSE and R^2proves that our model can have good prediction results for non-abnormal voltages in the quasi-dynamic dataset, but the prediction effect for abnormal voltages is poor. In voltage prediction, we need to pay attention to normal voltage drop prediction accuracy. Even if the voltage drops to a particularly low value for some outliers, it will quickly revert to its normal value. As a result, if the outlier is below the battery failure threshold, the battery is not thought to have failed. In general, the model can accurately predict the non-outlier voltage and the misprediction of some outliers will not affect the evaluation of battery failure time, so we believe that the model also has better prediction performance in the quasi-dynamic case.

3.4.2 Comparison of Prediction Results of Different Algorithm

Table 5 shows the evaluation results of LSTM, ANN, linear regression and SVM in the voltage prediction task, the criteria used are MSE, RMSE, MAE, and R^2.

Table 5: quasi-dynamic LSTM prediction results

Model	MSE	RMSE	MAE	R^2
LSTM	0.006	0.076	0.019	0.808
ANN	0.008	0.089	0.066	-4.847
Linear Regression	$3*10^4$	0.018	0.012	0.773
SVM	0.001	0.037	0.029	-0.012

The determination coefficients for the ANN and the SVM yielded negative values. This indicates inadequate fitting of the models to the data and insufficient capture of the voltage data variations, though other criteria are acceptable. Conversely, the regression model produced an R^2 score of 77.3%, which is in proximity to the R^2 value of the LSTM model (80.8%). This suggests that the regression model has a significant capacity to compensate for a significant portion of the variations observed in the data.

It is surprising that MSE, RMSE, and MAE of linear regression have smaller values than LSTM. The reason behind it could be that the whole trend of the total voltage is linear, as observed in Figure 2. The regression line passes most of the data points while LSTM does not. Nevertheless, the degree of fitting is still lower than LSTM. Linear regression is a simple line that ignores the natural voltage fluctuations. This fluctuation also influences the predicted voltage, which further leads to an inaccurate RUL calculation. Given the need for reliable prediction results, LSTM has better performance than the others.

Therefore, the LSTM model exhibits superior performance, indicating its ability to capture complex relationships and dependencies within the dataset.

3.5 Completion of Requirements and Sensitivity Analysis

3.5.1 Static Operation Model

Since the threshold for the static data set is 3%, the failure voltage is calculated from the initial values to be 3.21749 V. After 500h the predicted voltage is first reduced to the threshold at 808.3h, so the RUL for the static data set is 308.3 hours. In the predicted results, the voltages at t=800 and 1000 hours are 3.233V and 3.253V respectively. This is completely coincident with the true values at these two time points in the static data set.

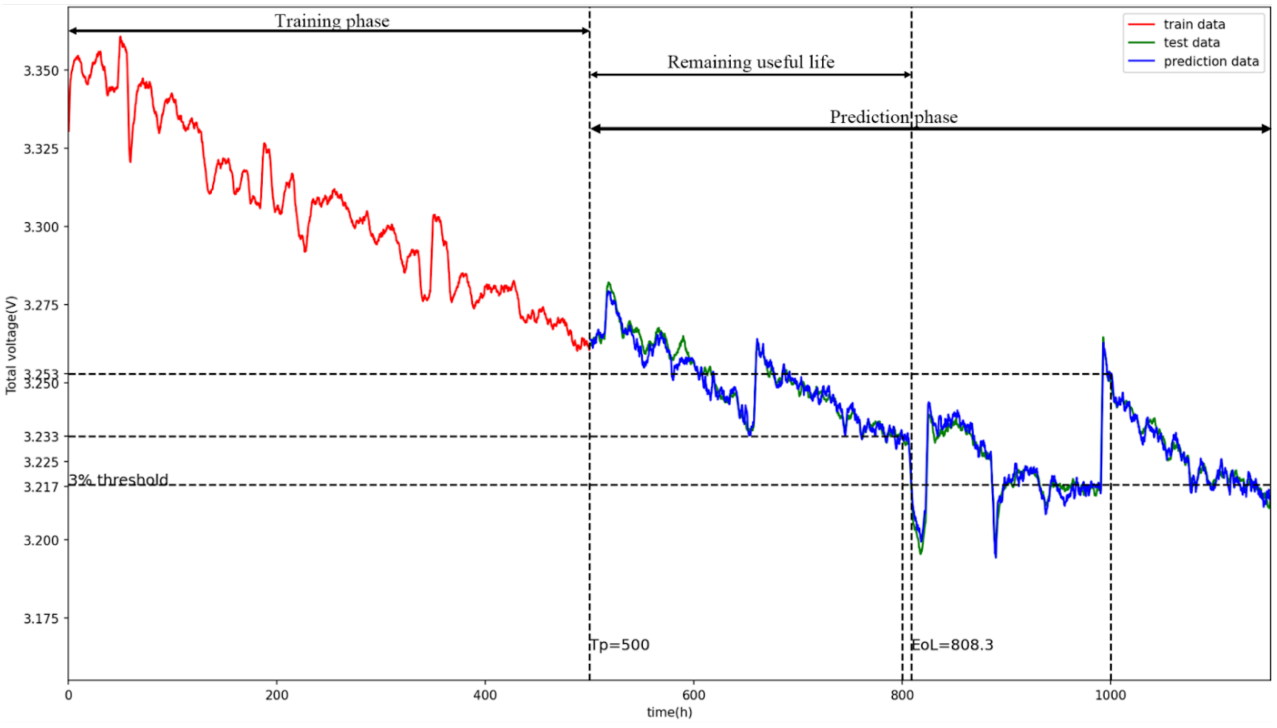


Figure 16: Predicted voltage values at 800 hours, 1000 hours and failure time in Static dataset

Since the threshold for the static data set is 3%, the failure voltage is calculated from the initial values to be 3.21749 V. After 500h the predicted voltage is first reduced to the threshold at 808.3h, so the RUL for the static data set is 308.3 hours. In the predicted results, the voltages at t=800 and 1000 hours are 3.233V and 3.253V respectively. This is completely coincident with the true values at these two time points in the static dataset.

3.5.2 Quasi-Dynamic Operation Model

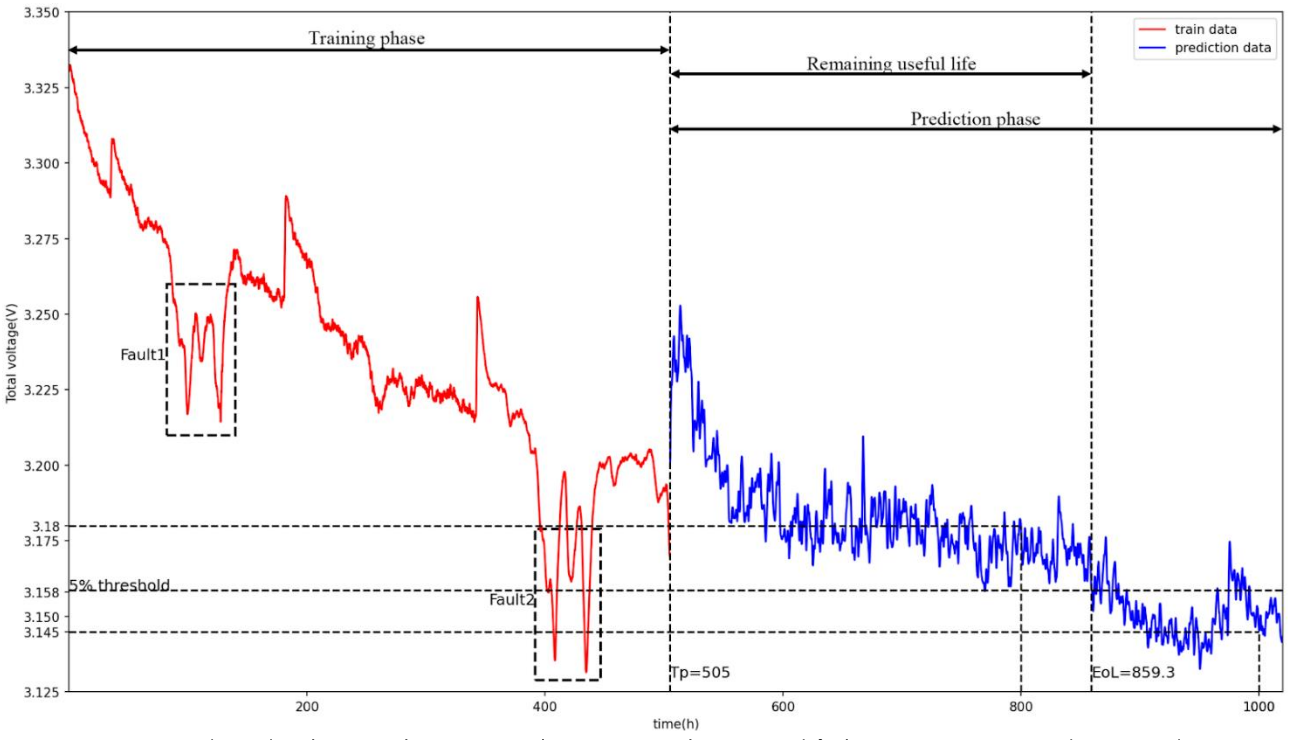


Figure 17: Predicted voltage values at 800 hours, 1000 hours and failure time in quasi-dynamic dataset

Figure 17 shows the predicted stack voltages at 800 and 1000 hours are 3.18 V and 3.145 V, respectively. When the threshold value is 5%, the threshold voltage calculated from the initial voltage is 3.158V. After 505 hours, the voltage first arrives at the point of failure at 859.3 hours, so the RUL at quasi-dynamic dataset is 354.3 hours.

4.Conclusion

In this project, the prediction of the remaining useful life of PEMFC using LSTM is studied under both static and quasi-dynamic datasets. Data condensation and a moving average filter are initially applied to both datasets. The data pre-processing aims to reduce the data size and remove noise, which could significantly lower the computational time while keeping the characteristics of the original data. The features are also reduced by correlation analysis, principal analysis, and clustering. The LSTM is adopted to predict the RUL, as well as linear regression, ANN, and the SVM for comparison. In static operation mode, the first 500 h of data are used for training, and the rest of the 653 h of data are for testing. The result analysis shows the LSTM has better performance than the others, with low RMSE, MAPE, and %Erft values and a high R^2score. The results prove the ability of LSTM in time series prediction. Thus, it is brought to the quasi-dynamic dataset, which contains more fluctuations and lacks the true voltage value. The performance of LSTM is less satisfactory compared with the static dataset. The evaluation result shows a bigger error due to the unstable operation condition and signal faults in the dataset, which happen frequently in real life. Surprisingly, the performance of linear regression is better than expected. After analysis, the team found that the overall trend of the voltage drop is in a linear relationship with time. This led us to combine the LSTM with linear regression in future work.

The next steps of this work consist of, first of all, in the dynamic data set, the data can be expanded. For example, collecting more data on PEMFC under different condition, helps to improve the generalization ability and prediction accuracy of the model. Larger datasets can help us better train models.

Second, the current model exhibits potential for further improvement, with ample room for enhancements. When incorporating the U3 feature as an input in the static model, notable improvements in prediction accuracy are observed. However, the original dynamic prediction model did not utilize U3, despite its strong correlation with Utot, because the dynamic dataset lacks U3 values. Hence, in the dynamic prediction model, we aim to integrate U3 as an input to forecast future Utot values. Therefore, a data imputation process is necessary to address this gap.

In our research analyzing the dataset and comparing algorithmic prediction results, we have discovered that linear regression can yield favorable outcomes for voltage prediction. This can be attributed to the presence of a linear relationship between the voltage drop trend and time within the original dataset. The inherent characteristics of linear regression allow it to effectively capture the overall trend in variable changes. Leveraging these two characteristics, we employ linear regression to establish a straight-line projection for predicting the value of U3, particularly focusing on its value after 500 hours. Subsequently, this predicted U3 value serves as an input for LSTM, enhancing the accuracy of Utot prediction. Our model demonstrates the ability to effectively capture atypical Utot variations, ensuring robust performance even in the presence of abnormal changes.

Acquiring a straight line that aligns closely with the actual downward trend through linear regression holds significant importance as it directly influences the ultimate prediction outcome of Utot. Given the presence of fluctuations in the quasi-dynamic data, it is crucial for the linear regression approach to be less susceptible to these outliers. To mitigate the impact of outliers, we have incorporated the Huber loss function, which reduces their weight in the regression process. This adaptation allows the linear regression results to more accurately conform to the overall trend, resulting in improved fitting capabilities.

The objective of utilizing linear fitting with the Huber loss function is to determine an optimal straight line that effectively aligns with the dataset. In contrast to traditional methods such as least squares or mean square error, the Huber loss function offers the advantage of mitigating the influence of outliers. Additionally, it allows for manual adjustment of the threshold parameter (δ), which controls the permissible range of data volatility. When selecting the optimal value for δ, it becomes essential to strike a judicious balance between the problem's contextual requirements, outlier tolerance, and the desired level of fitting. [46] The utilization of linear regression based on the Huber loss function serves as an effective approach to mitigating the impact of outliers while providing a more accurate depiction of the overall data trend.

Consequently, the integration of linear regression with the Huber loss function proves particularly valuable in handling future unknown data points of U3. By incorporating U3 as an input within the model, the prediction accuracy can be significantly enhanced, thereby facilitating improved performance.

Reference

[1] Cai, Y. et al. (2020) ‘A proton exchange membrane fuel cell-compound thermoelectric system: Bidirectional modeling and energy conversion potentials’, Energy conversion and management, 207, p. 112517. doi: 10.1016/j.enconman.2020.112517.

[2] Gao, F., Blunier, B. and Miraoui, A. (2012) Proton exchange membrane fuel cells modeling [electronic resource]. London : Hoboken, N.J.: ISTE ; Wiley.

[3] European Standard EN 13306, British Standards Institution; 2001.

[4] Hua, Z. et al. (2022) ‘A review on lifetime prediction of proton exchange membrane fuel cells system’, Journal of power sources, 529, p. 231256. doi: 10.1016/j.jpowsour.2022.231256.

[5] Futter, G. A., Latz, A. and Jahnke, T. (2019) ‘Physical modeling of chemical membrane degradation in polymer electrolyte membrane fuel cells: Influence of pressure, relative humidity and cell voltage’, Journal of power sources, 410-411, pp. 78–90. doi: 10.1016/j.jpowsour.2018.10.085.

[6] Sutharssan, T. et al. (2017) ‘A review on prognostics and health monitoring of proton exchange membrane fuel cell’, Renewable & sustainable energy reviews, 75, pp. 440–450. doi: 10.1016/j.rser.2016.11.009.

[7] International Standard, "Condition monitoring and diagnostics of machines - prognostics - part1: general guidelines," 3rd ed., International Organization for Standardization (ISO), no. ISO13381-1, 2015.

[8] Hua, Z. et al. (2020). ‘Health Indicators for PEMFC Systems Life Prediction Under Both Static and Dynamic Operating Conditions’ IEEE Xplore. doi: https://doi.org/10.1109/IECON43393.2020.9254916.

[9] Jouin, M. et al. (2014) ‘Prognostics of PEM fuel cell in a particle filtering framework’, International journal of hydrogen energy, 39(1), pp. 481–494. doi: 10.1016/j.ijhydene.2013.10.054.

[10] J. Liu, Q. Li, W. Chen, Y. Yan, Y. Qiu, and T. Cao, “Remaining useful life prediction of PEMFC based on long short-term memory recurrent neural networks,” International journal of hydrogen energy, vol. 44, no. 11, pp. 5470–5480, 2019, doi: 10.1016/j.ijhydene.2018.10.042.

[11] H. Liu, J. Chen, D. Hissel, and H. Su, “Remaining useful life estimation for proton exchange membrane fuel cells using a hybrid method,” Applied energy, vol. 237, pp. 910–919, 2019, doi: 10.1016/j.apenergy.2019.01.023.

[12] S. Zhang, T. Chen, F. Xiao, and R. Zhang, “Degradation prediction model of PEMFC based on multi-reservoir echo state network with mini reservoir,” International journal of hydrogen energy, vol. 47, no. 94, pp. 40026–40040, 2022, doi: 10.1016/j.ijhydene.2022.09.160.

[13] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, “Machinery health prognostics: A systematic review from data acquisition to RUL prediction,” Mechanical systems and signal processing, vol. 104, pp. 799–834, 2018, doi: 10.1016/j.ymssp.2017.11.016.

[14]L. Zheng, Y. Hou, T. Zhang, and X. Pan, “Performance prediction of fuel cells using long short-term memory recurrent neural network,” International journal of energy research, vol. 45, no. 6, pp. 9141–9161, 2021, doi: 10.1002/er.6443.

[15] F.-K. Wang, X.-B. Cheng, and K.-C. Hsiao, “Stacked long short-term memory model for proton exchange membrane fuel cell systems degradation,” Journal of power sources, vol. 448, p. 227591, 2020, doi: 10.1016/j.jpowsour.2019.227591.

[16] Y. Peng, T. Chen, F. Xiao, and S. Zhang, “Remaining useful lifetime prediction methods of proton exchange membrane fuel cell based on convolutional neural network-long short-term memory and convolutional neural network-bidirectional long short-term memory,” Fuel cells (Weinheim an der Bergstrasse, Germany), vol. 23, no. 1, pp. 75–87, 2023, doi: 10.1002/fuce.202200106.

[17]I. Ilic, B. Görgülü, M. Cevik, and M. G. Baydoğan, “Explainable boosted linear regression for time series forecasting,” Pattern recognition, vol. 120, p. 108144, 2021, doi: 10.1016/j.patcog.2021.108144.

[18] M. A. Patil et al., “A novel multistage Support Vector Machine based approach for Li ion battery remaining useful life estimation,” Applied energy, vol. 159, pp. 285–297, 2015, doi: 10.1016/j.apenergy.2015.08.119.

[19] R. Xie, R. Ma, S. Pu, L. Xu, D. Zhao, and Y. Huangfu, “Prognostic for fuel cell based on particle filter and recurrent neural network fusion structure,” Energy and AI, vol. 2, p. 100017, 2020, doi: 10.1016/j.egyai.2020.100017.

[20] Y. Cheng, N. Zerhouni, and C. Lu, “A hybrid remaining useful life prognostic method for proton exchange membrane fuel cell,” International journal of hydrogen energy, vol. 43, no. 27, pp. 12314–12327, 2018, doi: 10.1016/j.ijhydene.2018.04.160.

[21] Y. Cheng, N. Zerhouni, and C. Lu, “A Prognostic Framework for PEMFC Based on Least Squares Support Vector Regression-Particle Filter,” IEEE Xplore, Dec. 01, 2017. https://ieeexplore.ieee.org/abstract/document/8331036 (accessed Mar. 13, 2023).

[22] J. Jin, Y. Chen, C. Xie, W. Zhu, and F. Wu, “Remaining useful life prediction of PEMFC based on cycle reservoir with jump model,” International journal of hydrogen energy, vol. 46, no. 80, pp. 40001–40013, 2021, doi: 10.1016/j.ijhydene.2021.09.233.

[23] Wang, Yun, et al. "Fundamentals, materials, and machine learning of polymer electrolyte membrane fuel cell technology." Energy and AI 1 (2020): 100014.

[24] X. Yan and X. Su, Linear regression analysis [electronic resource] : theory and computing. Singapore ; Hackensack, NJ: World Scientific, 2009.

[25] Z. Zhang, F. Bai, H. -B. Quan, R. -J. Yin and W. -Q. Tao, "PEMFC Output Voltage Prediction Based on Different Machine Learning Regression Models," 2022 5th International Conference on Energy, Electrical and Power Engineering (CEEPE), Chongqing, China, 2022, pp. 401-406, doi: 10.1109/CEEPE55110.2022.9783124.

[26] Kheirandish, Azadeh, et al. "Dynamic modelling of PEM fuel cell of power electric bicycle system." International Journal of Hydrogen Energy 41.22 (2016): 9585-9594.

[27] A. Géron, Hands-on machine learning with Scikit-Learn and TensorFlow [electronic resource] : concepts, tools, and techniques to build intelligent systems. Sebastopol, California: O'Reilly Media, Inc., 2017

[28] H.-Z. Huang, H.-K. Wang, Y.-F. Li, L. Zhang, and Z. Liu, “Support vector machine based estimation of remaining useful life: current research status and future trends,” Journal of mechanical science and technology, vol. 29, no. 1, pp. 151–163, 2015, doi: 10.1007/s12206-014-1222-z.

[29] J. Zhang and W. Zhang, Intelligent fault diagnosis and prognosis for equipment, Beijing: National Defense Industry Press (2012)

[30] I.-S. Han and C.-B. Chung, “Performance prediction and analysis of a PEM fuel cell operating on pure oxygen using data-driven models: A comparison of artificial neural network and support vector machine,”

International journal of hydrogen energy, vol. 41, no. 24, pp. 10202–10211, 2016, doi: 10.1016/j.ijhydene.2016.04.247.

[31] Smith, Steven W. and Smith, Steven W, Digital signal processing : a practical guide for engineers and scientists. Amsterdam ; Boston: Newnes, 2003.

[32] Pengfei Luo, Min Zhang, Yile Liu, Dahai Han, and Qing Li, “A moving average filter based method of performance improvement for ultraviolet communication system,” in 2012 8th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP), 2012, pp. 1–4. doi: 10.1109/CSNDSP.2012.6292672.

[33] Z. Hua, Z. Zheng, M.-C. Péra, and F. Gao, “Remaining useful life prediction of PEMFC systems based on the multi-input echo state network,” Applied energy, vol. 265, p. 114791, 2020, doi: 10.1016/j.apenergy.2020.114791.

[34] Santarelli, M. G., and M. F. Torchio. "Experimental analysis of the effects of the operating variables on the performance of a single PEMFC." Energy conversion and management 48.1 (2007): 40-51.

[35] Jouin, Marine, et al. "Degradations analysis and aging modeling for health assessment and prognostics of PEMFC." Reliability Engineering & System Safety 148 (2016): 78-95.

[36] “Neural Network: Architecture, Components & Top Algorithms,” upGrad blog, May 06, 2020. https://www.upgrad.com/blog/neural-network-architecture-components-algorithms/

[37] A. G, “A review of Dropout as applied to RNNs,” Medium, Jun. 24, 2018. https://adriangcoder.medium.com/a-review-of-dropout-as-applied-to-rnns-72e79ecd5b7b (accessed May 19, 2023).

[38] A. Rakhecha, “Understanding Learning Rate,” Medium, Jul. 07, 2019. https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de

[39] N. Kumawat, “Adam Optimizer: In-depth explanation,” InsideAIML, 2020. https://insideaiml.com/blog/Adam-Optimizer:-In-depth-explanation-1051

[40] D. G. da Silva, M. T. B. Geller, M. S. dos S. Moura, and A. A. de M. Meneses, “Performance evaluation of LSTM neural networks for consumption prediction,” e-Prime, vol. 2, p. 100030, 2022, doi: 10.1016/j.prime.2022.100030.

[41] W. Huo, W. Li, Z. Zhang, C. Sun, F. Zhou, and G. Gong, “Performance prediction of proton-exchange membrane fuel cell based on convolutional neural network and random forest feature selection,” Energy conversion and management, vol. 243, p. 114367, 2021, doi: 10.1016/j.enconman.2021.114367.

[42] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” International journal of forecasting, vol. 22, no. 4, pp. 679–688, 2006, doi: 10.1016/j.ijforecast.2006.03.001.

[43] J. S. Armstrong and F. Collopy, “Error measures for generalizing about forecasting methods: Empirical comparisons,” International journal of forecasting, vol. 8, no. 1, pp. 69–80, 1992, doi: 10.1016/0169-2070(92)90008-W.

[44] Montgomery, D.C., Peck, E.A. and Vining, G.G., 2021. Introduction to linear regression analysis. John Wiley & Sons.

[45]T. Burr, “Pattern Recognition and Machine Learning,” Journal of the American Statistical Association, vol. 103, no. 482. Taylor & Francis, Alexandria, pp. 886–887, 2008. doi: 10.1198/jasa.2008.s236.

[46] Y. Liu, P. Zeng, and L. Lin, “Degrees of freedom for regularized regression with Huber loss and linear constraints,” Statistical papers (Berlin, Germany), vol. 62, no. 5, pp. 2383–2405, 2021, doi: 10.1007/s00362-020-01192-2.

Appendix

Appendix1.Parameter Definition

No.	Variables	Notation
1	Ageing Time (h)	Time
2	Single Cell #1 voltage (V)	U1
3	Single Cell #2 voltage (V)	U2
4	Single Cell #3 voltage (V)	U3
5	Single Cell #4 voltage (V)	U4
6	Single Cell #5 voltage (V)	U5
7	Stack voltage (V)	Utot
8	Current Density(A/cm^2)	J
9	Current(A)	I
10	Inlet temperatures of H2(℃)	TinH2
11	Outlet temperatures of H2(℃)	ToutH2
12	Inlet temperatures of AIR(℃)	TinAIR
13	Outlet temperatures of AIR(℃)	ToutAIR
14	Inlet temperatures of cooling water(℃)	TinWAT
15	Outlet temperatures of cooling water(℃)	ToutWAT
16	Inlet Pressure of H2(mbar)	PinH2
17	Outlet Pressure of H2(mbar)	PoutH2
18	Inlet Pressure of AIR (mbar)	PinAIR
19	Outlet Pressure of AIR (mbar)	PoutH2
20	Inlet flow rate of H2 (l/min)	DinH2
21	Outlet flow rate of H2 (l/min)	DoutH2
22	Inlet flow rate of AIR (l/min)	DinAIR
23	Inlet flow rate of AIR (l/min)	DoutAIR
24	Flow rate of cooling water (l/min)	DWAT
25	Inlet Hygrometry (Air) estimated (%)	HrAIRFC

Appendix2.Static Operation Mode Prediction Code

```
#Run this-1
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM,Dropout,Flatten
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from keras.callbacks import EarlyStopping
from keras.regularizers import l2
from keras.optimizers import Adam

#Run this-2/change the path
data = pd.read_csv('/content/MAF_realtime.csv')
X = data.iloc[:, [2,3,4,5,6,7,8]].values
y = data.iloc[:, [1]].values
t=data.iloc[:,[9]].values

# Normalize the features static
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
y= scaler.fit_transform(y)

# Create input and output sequences static
window_size = 10
X_seq_stat = []
y_seq_stat = []
for i in range(window_size, len(X)):
   X_seq_stat.append(X[i-window_size:i, :])
   y_seq_stat.append(y[i])

X_seq_stat = np.array(X_seq_stat)
y_seq_stat = np.array(y_seq_stat)

#static split
X_train, X_test, y_train, y_test = train_test_split(X_seq_stat, y_seq_stat,
test_size=0.57, random_state=0, shuffle=True)#0.57
X_train.shape, X_test.shape,y_train.shape,y_test.shape

#Run this-6 for static change the quasi_features.shape[1] to X.shape[1]
model = Sequential()
model.add(LSTM(units=190,return_sequences=True,input_shape=(10,7),kernel_reg
ularizer=l2(0.02)))
model.add(Dropout(0.2))
model.add(LSTM(units=120,return_sequences=True))
model.add(Dense(units=1))
optimizer = Adam(learning_rate=0.0001)
model.compile(optimizer, loss='mse')

early_stop = EarlyStopping(monitor='val_loss', patience=5) #Run this-7

#Run this-8
from sklearn.metrics import mean_squared_error, r2_score
history = model.fit(X_train, y_train, epochs=20, batch_size=15,
validation_data=(X_test, y_test), callbacks=[early_stop], verbose=1)
```

```
# predict the test set outputs
y_pred = model.predict(X_test)
# reshape y_test and y_pred to 2D arrays
y_pred_2d=y_pred[:,-1, :]
y_pred_2d= np.reshape(y_pred_2d,(y_pred_2d.shape[0],1))
y_test_2d=y_test

# calculate the mean squared error
mse = np.mean((y_pred_2d - y_test_2d)**2)

# calculate the MSE, RMSE, and R-squared
rmse = np.sqrt(mse)
r2 = r2_score(y_pred_2d,y_test_2d)

# plot convergence in terms of MSE, RMSE, and R-squared vs epochs for both
training and validation data
plt.figure(figsize=(8, 6))
plt.plot(history.history["loss"], label='Training MSE')
plt.plot(history.history["val_loss"], label= 'Validation MSE')
plt.xlabel("Epoch")
plt.ylabel("MSE/RMSE/R-squared")
plt.legend(loc='upper right')
print(r2)
print(rmse)

# Static prediction
y_all = model.predict(X_seq_stat)
y_all_rescaled=y_all[:,-1,:]
y_all_rescaled= np.reshape(y_all_rescaled,(y_all_rescaled.shape[0],1))

#Static rescaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(3.19425, 3.360625))
# fit and transform the data
y_all_rescaled = scaler.fit_transform(y_all_rescaled)
# rescaled_y = pd.DataFrame(data_scaled, columns=data.columns)
y=scaler.fit_transform(y)
y_all_rescaled.shape,y.shape

# Commented out IPython magic to ensure Python compatibility.
#original
# %matplotlib notebook
# %matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(100, 50))
ax = fig.add_subplot(111)

ax.plot(y,label='true value')
ax.plot(y_all_rescaled,label='predicted')
plt.legend()

# Add labels and title
ax.set_xlabel('Time')
ax.set_ylabel('Values')
ax.set_title('Predicted vs. Actual Values')
plt.show()
```

Appendix3.Quasi-Dynamic Operation Mode Prediction Code

```
#Run this-1
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM,Dropout,Flatten
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from keras.callbacks import EarlyStopping
from keras.regularizers import l2
from keras.optimizers import Adam

#Run this-2/change the path
data_dynamic= pd.read_csv('/content/quasi_1000_MAF_data.csv')
quasi_features= data_dynamic.iloc[0:5015,[0,2,3,4,5,6,7,8,9]].values
quasi_features_1000= data_dynamic.iloc[:,[0,2,3,4,5,6,7,8,9]].values
time_quasi=data_dynamic.iloc[0:5015,[0]].values
time_quasi_1000=data_dynamic.iloc[:,[0]].values
totalvolt_quasi_data= data_dynamic.iloc[0:5015,[1]].values

# Normalize the features dynamic #Run this-3
scaler = MinMaxScaler()
quasi_features = scaler.fit_transform(quasi_features)
totalvolt_quasi_data= scaler.fit_transform(totalvolt_quasi_data)

# Dynamic scaling #Run this-4
window_size = 10
X_seq = []
y_seq = []
for i in range(window_size, len(quasi_features)):
   X_seq.append(quasi_features[i-window_size:i, :])
   y_seq.append(totalvolt_quasi_data[i])

X_seq = np.array(X_seq)
y_seq = np.array(y_seq)

#Dynamic Test #Run this-5
X_train, X_test, y_train, y_test = train_test_split(X_seq, y_seq,
test_size=0.2, random_state=32, shuffle=True)
X_train.shape, X_test.shape,y_train.shape,y_test.shape

#Run this-6 for static change the quasi_features.shape[1] to X.shape[1]
model = Sequential()
model.add(LSTM(units=190,return_sequences=True,input_shape=(10,quasi_feature
s.shape[1]),kernel_regularizer=l2(0.02)))
model.add(Dropout(0.2))
model.add(LSTM(units=120,return_sequences=True))
model.add(Dense(units=1))
optimizer = Adam(learning_rate=0.0001)
model.compile(optimizer, loss='mse')

early_stop = EarlyStopping(monitor='val_loss', patience=5) #Run this-7

#Run this-8
from sklearn.metrics import mean_squared_error, r2_score
```

```
history = model.fit(X_train, y_train, epochs=20, batch_size=15,
validation_data=(X_test, y_test), callbacks=[early_stop], verbose=1)
# predict the test set outputs
y_pred = model.predict(X_test)
# reshape y_test and y_pred to 2D arrays
y_pred_2d=y_pred[:,-1, :]
y_pred_2d= np.reshape(y_pred_2d,(y_pred_2d.shape[0],1))
y_test_2d=y_test

# calculate the mean squared error
mse = np.mean((y_pred_2d - y_test_2d)**2)

# calculate the MSE, RMSE, and R-squared
rmse = np.sqrt(mse)
r2 = r2_score(y_pred_2d,y_test_2d)

plt.figure(figsize=(8, 6))
plt.plot(history.history["loss"], label='Training MSE')
plt.plot(history.history["val_loss"], label= 'Validation MSE')
plt.xlabel("Epoch")
plt.ylabel("MSE/RMSE/R-squared")
plt.legend(loc='upper right')
print(r2)
print(rmse)

#Time based scaling
scaler = MinMaxScaler()
time_quasi_scaled=scaler.fit_transform(time_quasi_1000)

#Run this-9 features scaling
scaler = MinMaxScaler()
dynamic_scaled_1000=scaler.fit_transform(quasi_features_1000)

# Dynamic scaling time
time_ind=np.broadcast_to(time_quasi_scaled, (len(time_quasi_scaled), 9))
window_size = 10
time_seq = []
for i in range(window_size, len(time_ind)):
   time_seq.append(time_ind[i-window_size:i, :])

time_seq = np.array(time_seq)

#Dynamic total scaling #Run this-10
window_size = 10
dynamic_1000_seq = []
for i in range(window_size, len(dynamic_scaled_1000)):
   dynamic_1000_seq.append(dynamic_scaled_1000[i-window_size:i, :])

dynamic_1000_seq = np.array(dynamic_1000_seq)

#Total quasi prediction #Run this-11
total_dynamic = model.predict(dynamic_1000_seq)
total_dynamic_rescaled=total_dynamic[:,-1,:]
total_dynamic_rescaled=
np.reshape(total_dynamic_rescaled,(total_dynamic_rescaled.shape[0],1))

#Time quasi prediction
total_dynamic_time = model.predict(time_seq)
```

```
total_dynamic_time_rescaled=total_dynamic_time[:,-1,:]
total_dynamic_time_rescaled=
np.reshape(total_dynamic_time_rescaled,(total_dynamic_time_rescaled.shape[0]
,1))

#quasi data prediction #Run this-12
y_quasi = model.predict(X_seq)
y_quasi_rescaled=y_quasi[:,-1, :]
y_quasi_rescaled= np.reshape(y_quasi_rescaled,(y_quasi_rescaled.shape[0],1))
y_quasi_rescaled.shape

#quasi data scaling #Run this-13
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(3.1315, 3.3324))
# fit and transform the data
y_quasi_rescaled = scaler.fit_transform(y_quasi_rescaled)
total_dynamic_rescaled = scaler.fit_transform(total_dynamic_rescaled)
total_dynamic_time_rescaled=scaler.fit_transform(total_dynamic_time_rescaled
)
totalvolt_quasi_data=scaler.fit_transform(totalvolt_quasi_data)
y_quasi_rescaled.shape,totalvolt_quasi_data.shape,total_dynamic_rescaled.sha
pe,total_dynamic_time_rescaled.shape

# Commented out IPython magic to ensure Python compatibility.
#dynamic #Run this-14
# %matplotlib notebook
# %matplotlib inline
import matplotlib.pyplot as plt


fig = plt.figure(figsize=(100, 50))
ax = fig.add_subplot(111)

# Create scatter plot of predicted vs. target values
ax.plot(totalvolt_quasi_data,label='true value')
ax.plot(y_quasi_rescaled,label='predicted')
ax.plot(total_dynamic_time_rescaled,label='predicted')
ax.plot(total_dynamic_rescaled,label='predicted')
plt.legend()

# Add labels and title
ax.set_xlabel('Time')
ax.set_ylabel('Values')
ax.set_title('Predicted vs. Actual Values')
plt.show()
```

Appendix4. Static MAF Code

```
clear;close all
%% data merging
A1 = readtable('E:\Sheffield\Spring\ITP\data\FC1_Ageing_part1');
A2 = readtable('E:\Sheffield\Spring\ITP\data\FC1_Ageing_part2');
A3 = readtable('E:\Sheffield\Spring\ITP\data\FC1_Ageing_part3');
A = union(A1,A2,'stable');
data = union(A,A3,'stable');
%% data condence for every 20 min
nRows = size(data,1);
data_c = table();
data_c = [data_c;data(1,:)];
for i = 1: nRows
   if data.Time(i) - data_c.Time(end) >= 0.3
     data_c = [data_c;data(i,:)];
   end
end
%%
figure,plot(data.Time,data.Utot);
hold on
plot(data_c.Time,data_c.Utot);
xlabel('Time(h)')
ylabel('Stack Voltage(V)')
title('Static Dataset')
legend('original data','condensed data')
%% moving average filter
windowSize = 8;
output1 = data_c;
for i=1:width(data_c)
   %output(:,i) = filter(ones(1,windowSize)/windowSize,1,data_c{:,i});
   output1{:,i} = movmean(data_c{:,i}, windowSize);
   %output = movmean(data_c.Utot, windowSize);
end
%% select feature
result = table;
result.Time = output1.Time;
result.Utot = output1.Utot;
result.U3 = output1.U3;
result.DoutAIR = output1.DoutAIR;
result.PoutAIR = output1.PoutAIR;
result.HrAIRFC = output1.HrAIRFC;
result.DoutH2 = output1.DoutH2;
result.ToutWAT = output1.ToutWAT;
result.DWAT = output1.DWAT;
%%
% writetable(result,'origin_condenced_dataset.csv');
%%
figure, plot(data_c.Time,data_c.Utot,'b');
hold on
plot(data_c.Time,output1.Utot,'r','LineWidth',1.5);
xlabel('Time(h)')
ylabel('Stack Voltage(V)')
title('Static MAF')
legend('condensed data','MAF data')
```

Appendix5. Quasi-Dynamic MAF Code

```
%% load quasi-dynamic data
clear all; close all
A1 = readtable('E:\Sheffield\Spring\ITP\data\FC2_Ageing_part1');
A1 = removevars(A1,["U1","U2","U3","U4","U5","Utot"]);
A2 = readtable('E:\Sheffield\Spring\ITP\data\FC2_Ageing_part2');
data = union(A1,A2,'stable');
%% data condence for every 9 min
nRows = size(data,1);
data_c = table();
data_c = [data_c;data(1,:)];
for i = 1:nRows
    if data.Time(i) - data_c.Time(end) >= 0.1
      data_c = [data_c;data(i,:)];
    end
%    data_c = [data_c;data(i,:)];

end
%%
figure
plot(data.Time,data.TinH2,'b');
hold on
plot(data_c.Time,data_c.TinH2,'r');
xlabel('Time(h)')
ylabel('Stack Voltage(V)')
title('quasi-Dynamic Dataset')
legend('original data','condensed data')
%%
variables = table;
variables.Time = data_c.Time;
variables.TinH2 = data_c.TinH2;
variables.TinAIR = data_c.TinAIR;
variables.ToutH2 = data_c.ToutH2;
variables.TinWAT = data_c.TinWAT;
variables.I = data_c.I;
variables.PoutAIR = data_c.PoutAIR;
variables.HrAIRFC = data_c.HrAIRFC;
variables.ToutAIR = data_c.ToutAIR;
%% moving average filter
windowSize = 10;
output = variables;
for i=1:width(variables)
    %output(:,i) = filter(ones(1,windowSize)/windowSize,1,data_c{:,i});
    output{:,i} = movmean(variables{:,i}, windowSize);
    %output = movmean(data_c.Utot, windowSize);
end
%%
% writetable(output,'quasi_1000_MAF_data.csv');
%%
figure
plot(variables.Time,variables.ToutH2,'b');
hold on
plot(variables.Time,output.ToutH2,'r','LineWidth',1.5);
xlabel('Time(h)')
ylabel('Stack Voltage(V)')
title('Quasi-Dynamic MAF')
legend('original data','MAF data')
```