Battery Life Cycle Prediction using Feedforward Neural Network

ME674 - Term Project Submission

by

JOEL JOSEPH A

(210103060)

under the guidance of

Dr. Sukhomay Pal

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI-781039, ASSAM

TABLE OF CONTENTS

1. Introduction

Predicting the remaining useful life (RUL) of lithium-ion batteries is crucial for the efficient management of electric vehicle (EV) batteries, where early detection of degradation can significantly enhance cost savings and operational reliability. This project employs a feedforward neural network to predict the RUL of 14 NMC-LCO 18650 batteries with a nominal capacity of 2.8 Ah, which were cycled over 1000 times at 25°C using a CC-CV charge rate of C/2 and a discharge rate of 1.5C. By modeling the battery degradation patterns, the neural network aims to provide accurate predictions of the RUL based on cycle-by-cycle features.

2. Dataset Overview

The dataset for this project contains cycle summary data for 14 batteries, collected from experiments conducted by the Hawaii Natural Energy Institute. Each battery was cycled at a constant temperature, charge rate, and discharge rate, producing features summarizing the voltage and current behavior over each cycle. These features are essential for training the model to predict battery degradation and remaining useful life.

2.1 Variables in the Dataset

- **Cycle Index**: Number of charge-discharge cycles completed.
- **F1**: Discharge Time (seconds).
- **F2**: Time at 4.15V (seconds).
- **F3**: Time Constant Current (seconds).
- **F4**: Decrement from 3.6V to 3.4V (seconds).
- **F5**: Maximum Voltage during Discharge (volts).
- **F6**: Minimum Voltage during Charge (volts).
- **F7**: Charging Time (seconds).
- **Total Time**: Total time per cycle (seconds).
- **RUL**: Target variable representing the remaining useful life (in cycles) of the battery.

3. Methodology

3.1 Neural Network Design

A feedforward neural network model was designed with an input layer, three hidden layers, and an output layer. After experimentation, the optimal configuration was found to have 3,2,1 hidden neurons for the first, second and third hidden layers respectively and a learning rate of 10^{-4}. The ReLU activation function was used in the hidden layer to allow for non-linear transformations, while the output layer had a linear activation function to enable the prediction of continuous RUL values.

3.2 Activation Function

The ReLU activation function was selected for the hidden layer to introduce non-linearity, allowing the model to capture more complex relationships between features. ReLU was chosen for its efficiency in handling vanishing gradients and accelerating convergence.

3.3 Gradient Clipping and Weight Initialization

To address issues of gradient instability, gradient clipping was applied with a maximum gradient norm to prevent exploding gradients. He initialization was used to initialize the weights, aiding in effective convergence by scaling weights according to layer size.

4. Model Optimization

To ensure the feedforward neural network model accurately predicts the remaining useful life (RUL) of lithium-ion batteries, a careful optimization of hyperparameters was essential. The two main hyperparameters optimized were the **number of hidden neurons** and the **learning rate**. This section discusses the methods used to select these parameters, as well as the techniques implemented to improve model stability and performance.

4.1 Hyperparameter Optimization: Hidden Neurons and Learning Rate

The network's hidden layer complexity (number of neurons) and learning rate were tuned to achieve a balance between model capacity and convergence rate:

- **Hidden Neurons**: The hidden layer size controls the model's ability to capture complex relationships in the data. Too few neurons may result in underfitting, while too many may lead to overfitting and high computational cost. For this project, the number of hidden neurons where fixed as 3,2,1 respectively for the first, second and third hidden layers respectively.
- **Learning Rate**: Learning rate affects how quickly the model converges to a solution. A low learning rate can lead to slow convergence, while a high learning rate can cause the model to diverge. A learning rate of 10^{-4} was used in the project.

Each combination of these hyperparameters was evaluated by training the model and recording the **Mean Squared Error (MSE)**. This iterative process identified the optimal configuration that minimized the MSE, resulting in a configuration of **3,2,1** hidden neurons for the first, second and third hidden layers respectively and a learning rate of $\mathbf{10^{-4}}$

4.2 Activation Functions and Weight Initialization

- **ReLU Activation**: The **Rectified Linear Unit (ReLU)** activation function was chosen for the hidden layer due to its computational efficiency and its ability to mitigate vanishing gradients, a common issue in deeper networks. ReLU’s non-linearity enables the network to model complex interactions among features, which is critical for accurately predicting battery life.
- **Linear Output Activation**: Since RUL is a continuous target variable, a linear activation function was used in the output layer. This allowed the model to predict a range of values directly without transformations, facilitating smooth error backpropagation.
- **He Initialization**: Weights were initialized using **He initialization**, which scales the weights by the square root of 2/input size. This technique is beneficial when using **ReLU** activations as it helps maintain a balance in variance through layers, aiding convergence.

4.3 Gradient Clipping

During backpropagation, gradients can become excessively large, leading to unstable weight updates that may cause the model to diverge. To address this, **gradient clipping** was applied. Specifically, the gradient norm was capped to a maximum value (1.0) to prevent exploding gradients while still allowing gradient descent to proceed effectively. Gradient clipping provided the network with stability, reducing the risk of high MSE values caused by abrupt jumps in weight updates.

4.4 Training and Evaluation

The network was trained over 1000 iterations, with the MSE calculated at each iteration to monitor convergence. The configuration achieved the lowest MSE of **336.61**, indicating that the model accurately minimized error across cycles.

5. Results and Error Metrics

The model's performance was evaluated using several **error metrics**, which measure the accuracy and reliability of the predictions for the Remaining Useful Life (RUL) of the lithium-ion batteries. These metrics help determine how closely the predicted RUL values match the actual RUL values and give us a sense of the model's generalization ability.

5.1. Model Training and MSE Progression

The neural network was trained over 1000 iterations using the configuration: **3,2,1** hidden neurons for the **first, second** and **third** hidden layers respectively and a learning rate of **10^{-4}**.. The training process involves the model adjusting its weights through backpropagation to minimize the **Mean Squared Error (MSE)**, a widely-used metric in regression problems.

The **MSE** is the average of the squared differences between the predicted and actual values, representing the average magnitude of errors. A lower MSE indicates better model performance, as it suggests that the model's predictions are closer to the true values.

The model's MSE gradually decreased over the course of the training process, indicating that the model was learning and converging towards a more accurate prediction. At the final iteration (1000th iteration), the MSE stabilized at **0.08118921858774839**, which is considered a good result for this type of regression problem.

5.2. Root Mean Squared Error (RMSE)

In addition to MAE and MSE, **Root Mean Squared Error (RMSE)** is another useful metric that gives the error in the same units as the target variable (RUL in this case). It is the square root of the MSE, providing a more interpretable value by bringing the error back to the original scale.

- **RMSE**: 0.285

RMSE is often used because it penalizes large errors more than MAE due to the squaring operation in its calculation. In this case, the RMSE of **0.285** provides a sense of how large the typical prediction error is, with larger errors in predictions contributing more heavily to this value.

5.3. Model Performance Visualizations

To provide a more intuitive understanding of the model's performance, several visualizations were plotted:

- **Prediction vs. Actual RUL**: A scatter plot comparing the predicted RUL values against the true RUL values for the test set. Ideally, the points should cluster along a 45-degree line (perfect prediction).
- **MSE Over Iterations**: A line plot showing how the MSE evolved over the course of training. This plot allows us to visually confirm that the model converged smoothly and steadily.
- **Residuals Plot**: A plot of residuals (errors between predicted and actual values) helps assess whether there are systematic patterns in the errors that the model failed to capture.

5.4. Interpretation of Results

Based on the error metrics, the model performs well with a strong fit and relatively low errors in terms of both **MSE (0.08088386855577023) a**nd **RMSE (0.284)**. However, there may still be some room for improvement, particularly by exploring more advanced architectures or incorporating additional features that might explain the remaining variance in RUL predictions.

Despite the optimization of hyperparameters (such as the learning rate and hidden neurons), the presence of some outliers or noise in the dataset could explain the less-than-perfect MAE and RMSE values. Further improvements could involve refining the dataset by removing noise, adding more features, or utilizing more advanced architectures such as **Recurrent Neural Networks (RNNs)**, which could capture temporal dependencies in battery degradation over cycles.

Metric	Value	Explanation
Mean Squared Error (MSE)	0.08088386855577023	Average squared differences between predicted and actual RUL. Lower MSE indicates better model performance.
Root Mean Squared Error (RMSE)	0.284	Square root of MSE, providing the error in the same units as the target variable (RUL).

6. Conclusion

The Feedforward Neural Network (FFNN) model successfully predicted the Remaining Useful Life (RUL) of lithium-ion batteries based on various operational features, demonstrating strong predictive accuracy. The optimized model achieved the following performance:

- **Mean Squared Error (MSE)**: 0.08088386855577023

The configuration in which the model was trained is **3,2,1** hidden neurons for the **first, second** and **third** hidden layers respectively and a learning rate of **10^{-4}**, which provided the best balance between performance and training stability.

While the model showed strong performance, further improvements could be made by incorporating additional features, experimenting with advanced neural network architectures like Recurrent Neural Networks (RNNs), or using regularization techniques to prevent overfitting.

Overall, this model is suitable for real-world applications, including battery management systems and electric vehicle maintenance, where predicting battery degradation and planning for replacements is crucial for efficiency and cost savings.